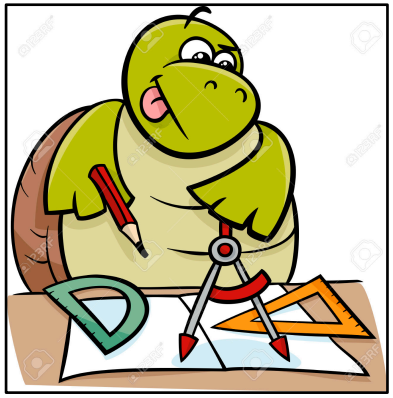
Decomposition Theorems in UDGs and Applications in Param. Complexity

1. Excluded Grid Minors and Efficient Polynomial-Time Approximation Schemes. Fomin, Lokshtanov, Saurabh. J. ACM 2018.

2. Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs. Fomin, Lokshtanov, Panolan, Saurabh, Zehavi. Discrete & Computational Geometry 2019.

3. Contraction Decomposition in Unit Disk Graphs and Algorithmic Applications in Parameterized Complexity. Panolan, Saurabh, Zehavi. SODA 2019.

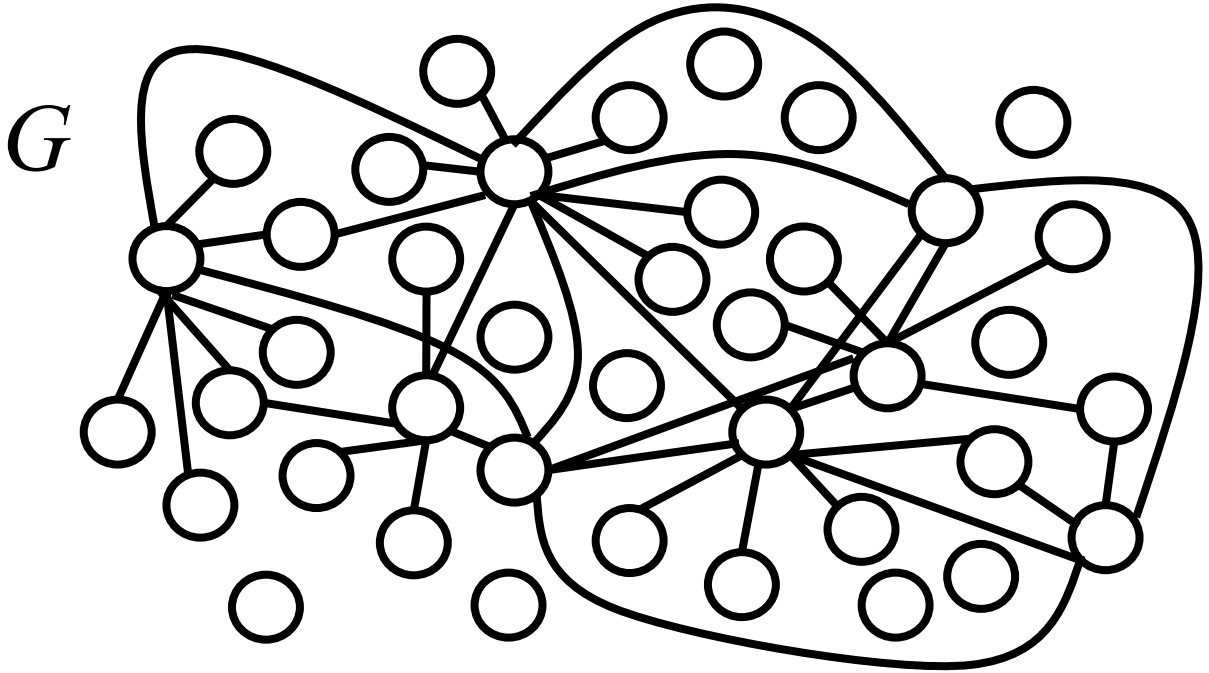


Parameterized Complexity

Vertex Cover.

Input: Graph G; non-negative integer k.

Question: Does G have a vertex cover of size at most k?



$n = m = 40$

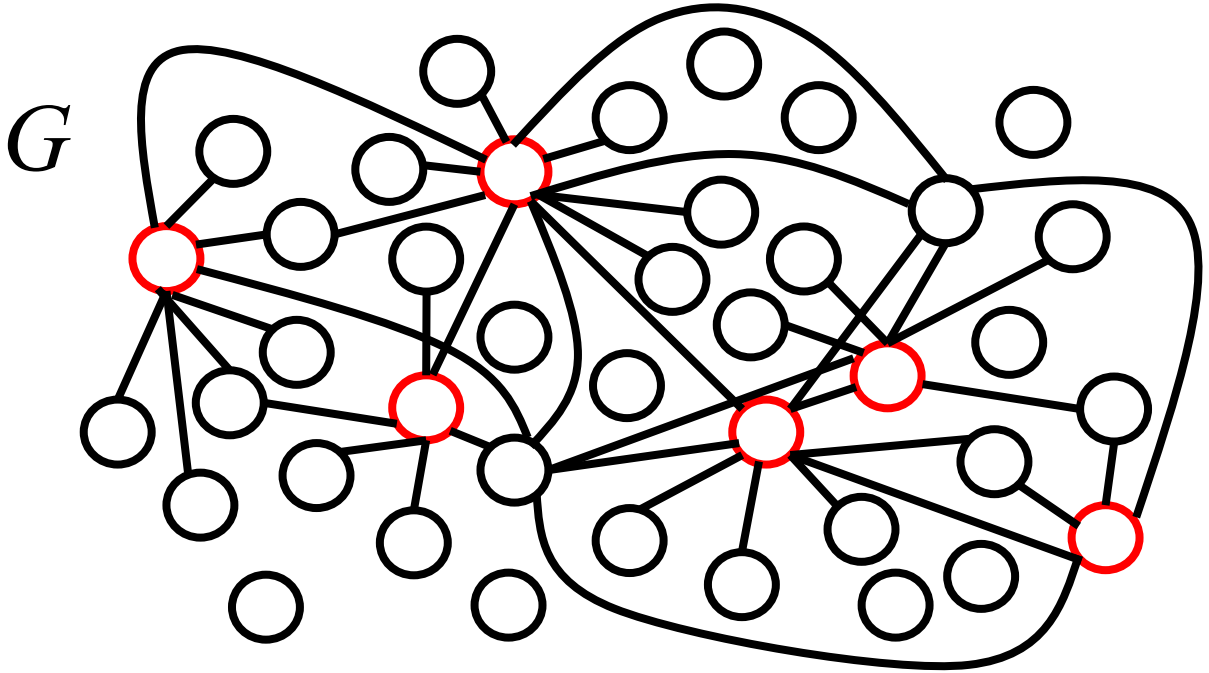
$k = 5$

Parameterized Complexity

Vertex Cover.

Input: Graph G; non-negative integer k.

Question: Does G have a vertex cover of size at most k?



No

$n = m = 40$

$k = 5$

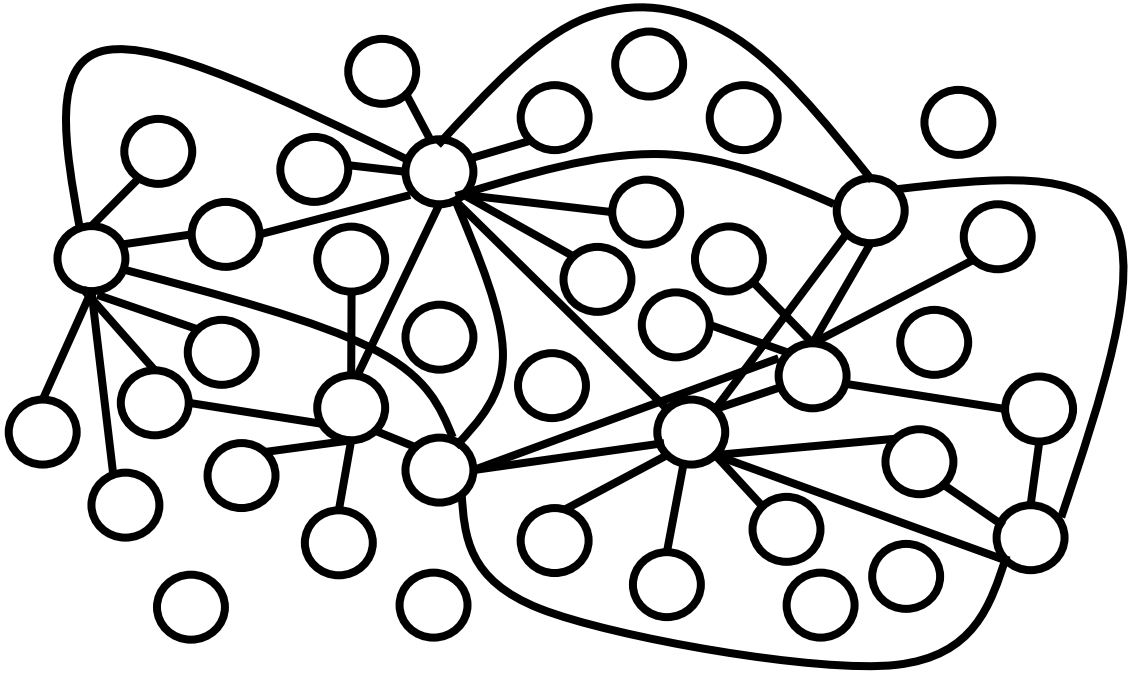
Parameterized Complexity

Brute-Force. $\binom{n}{k}(n+m) = \binom{40}{5}(40+40)$

$= 658{,}008 \cdot 80$

$= 52{,}640{,}640$

G



$n = m = 40$

$k = 5$

Parameterized Complexity

Brute-Force. $\binom{n}{k}(n+m) = 658{,}008 \cdot 80 = 52{,}640{,}640$

Magic 1. $2^k(n+m) = 32 \cdot 80 = 2{,}560$

Parameterized Complexity

Brute-Force. $\binom{n}{k}(n+m) = 658{,}008 \cdot 80 = 52{,}640{,}640$

Magic 1. $2^k(n+m) = 32 \cdot 80 = 2{,}560$

Magic 2. $(n+m+k^3)$ = 205 → Equivalent instance:
$n' = 2k = 10$; $m' = k^2 = 25$.

Parameterized Complexity

Brute-Force. $\binom{n}{k}(n+m) = 658{,}008 \cdot 80 = 52{,}640{,}640$

Magic 1. $2^k(n+m) = 32 \cdot 80 = 2{,}560$

Magic 2. $(n+m+k^3)$ = 205 → Equivalent instance:
$n' = 2k = 10$; $m' = k^2 = 25$.

With Brute-Force. 205 + $\binom{10}{5}(10+25)$ = 9,025

With Magic 1. 205 + $2^5(10+25)$ = 1,245

Parameterized Complexity

Brute-Force. $\binom{n}{k}(n+m) = 658{,}008 \cdot 80 = 52{,}640{,}640$

Magic 1. $2^k(n+m) = 32 \cdot 80 = 2{,}560$

Magic 2. $(n+m+k^3)$ = 205 → Equivalent instance:
$n' = 2k = 10$; $m' = k^2 = 25$.

With Brute-Force. 205 + $\binom{10}{5}(10+25)$ = 9,025

With Magic 1. 205 + $2^5(10+25)$ = 1,245

What if *k* = 10, *n* = 100 and *m* = 300?

Parameterized Complexity

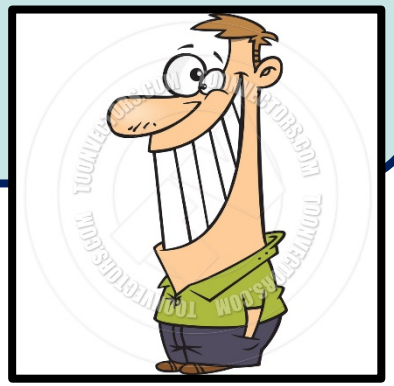
Brute-Force. $\binom{n}{k}(n+m) \sim 7 \cdot 10^{15}$

Magic 1. $2^k(n+m) = 409,600$

Magic 2. $(n + m + k^3)$ = 1,400 → Equivalent instance:
n' = $2k$ = **20**; m'=k^2 = **100**.

With Brute-Force. 1,400 + $\binom{20}{10} \cdot 120$ = **22,172,120**

With Magic 1. 1,400 + $2^{10} \cdot 120$ = **124,280**

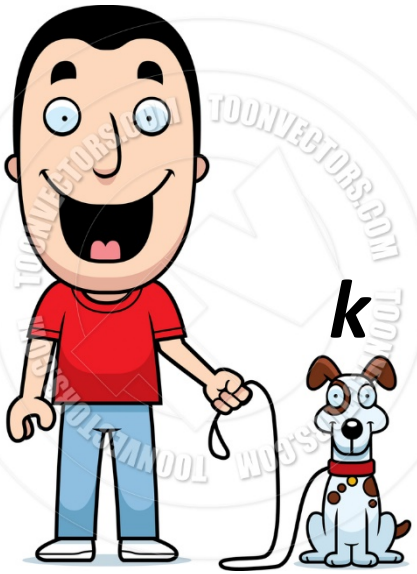


What if *k* = 10, *n* = 100 and *m* = 300?

Parameterized Complexity

Parameterized problem.

Each instance is associated with a parameter k.



Fixed-Parameter Tractability

$k << n$

$f(k) \cdot n^{O(1)} = O^*(f(k))$

$\binom{n}{k}$ … or worse!

Fixed-Parameter Tractability

$k << n$

$f(k) \cdot n^{O(1)} = O^*(f(k))$ $\binom{n}{k}$ *… or worse!*

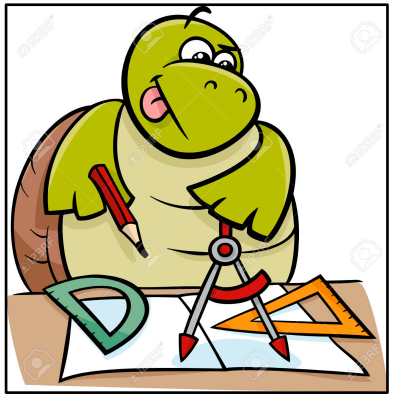
Unfortunately, not all problems are FPT!

Decomposition Theorems in UDGs and Applications in Param. Complexity

1. Excluded Grid Minors and Efficient Polynomial-Time Approximation Schemes. Fomin, Lokshtanov, Saurabh. J. ACM 2018.

2. Finding, Hitting and Packing Cycles in Subexponential Time on Unit Disk Graphs. Fomin, Lokshtanov, Panolan, Saurabh, Zehavi. Discrete & Computational Geometry 2019.

3. Contraction Decomposition in Unit Disk Graphs and Algorithmic Applications in Parameterized Complexity. Panolan, Saurabh, Zehavi. SODA 2019.

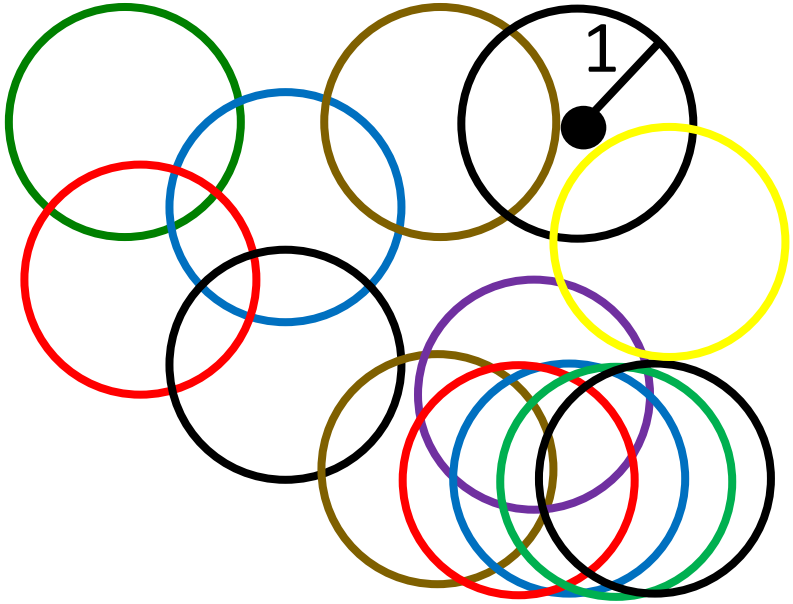


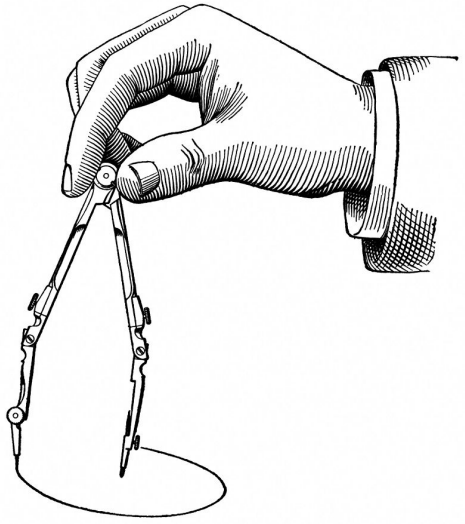
Outline

- **Introduction:**
  - UDGs: Definition
  - Decomposition I: Branch on Cliques
- Decomposition II: General Decomposition
- Decomposition III: Contraction Decomposition

Introduction: Unit Disk Graphs

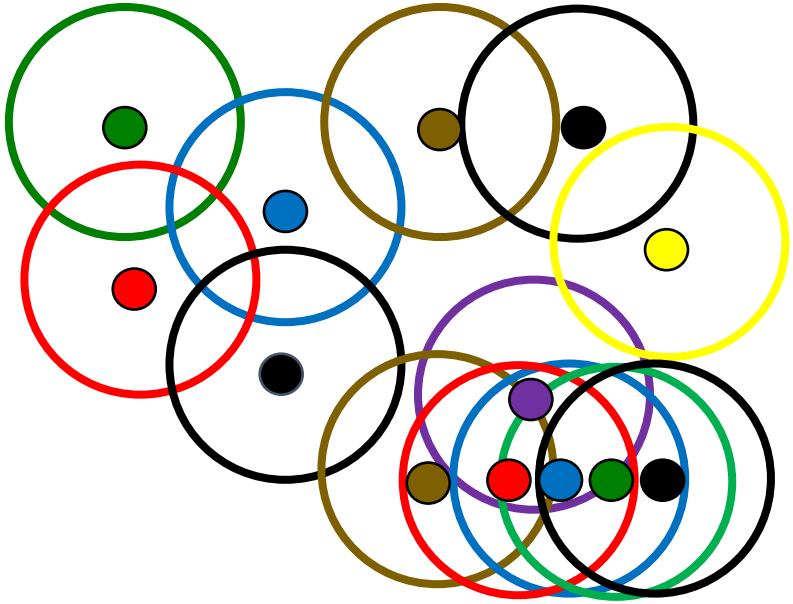
Unit Disk Graph.
Intersection graph of unit circles in the plane.





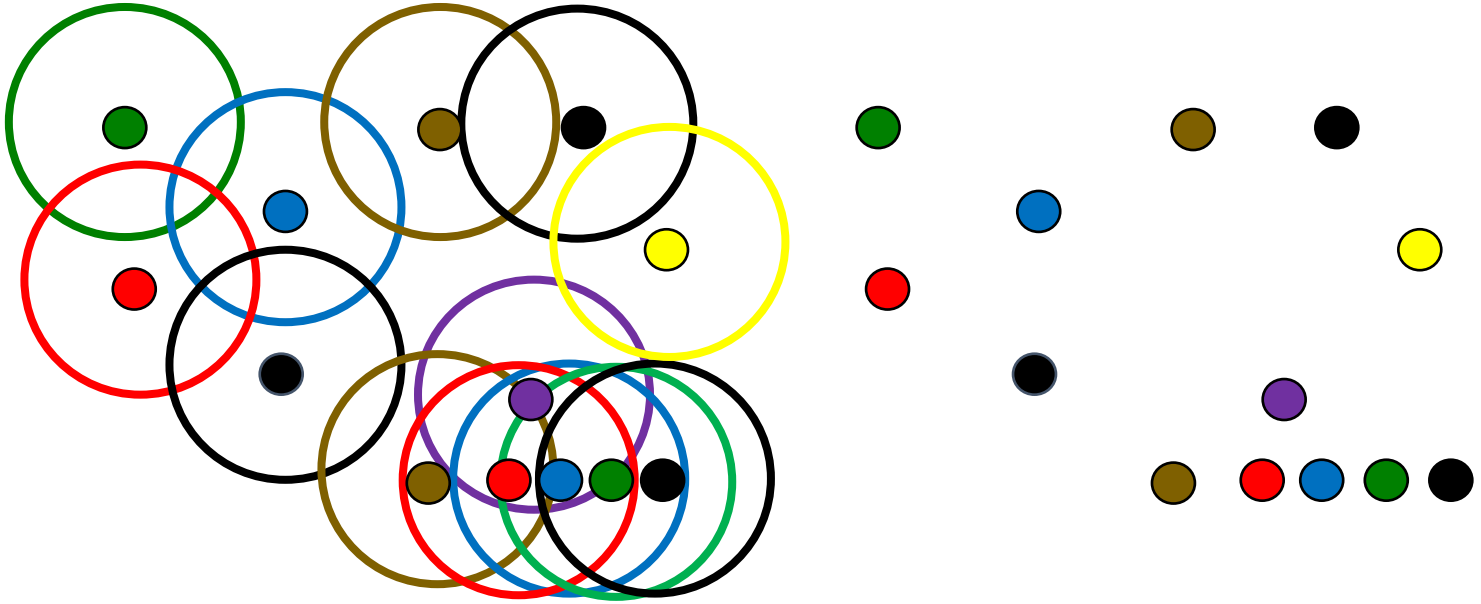
Introduction: Unit Disk Graphs

Unit Disk Graph.
Intersection graph of unit circles in the plane.



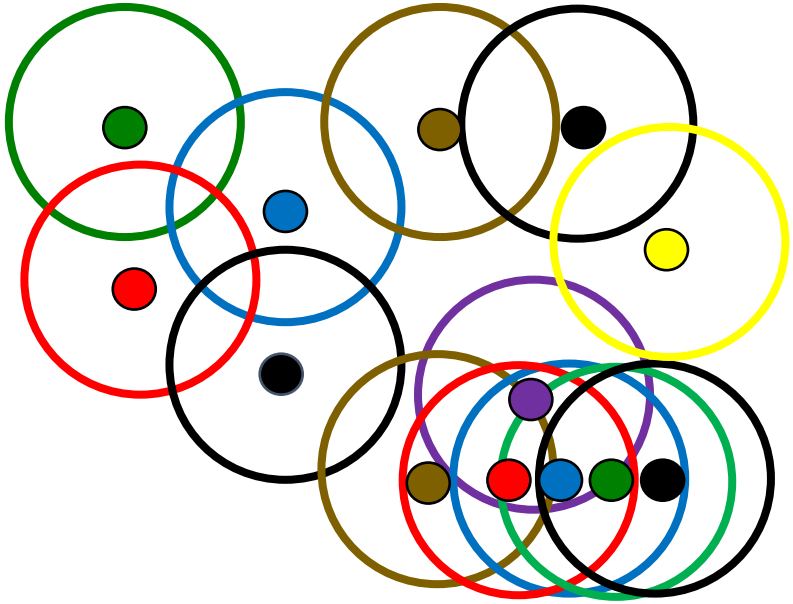
Introduction: Unit Disk Graphs

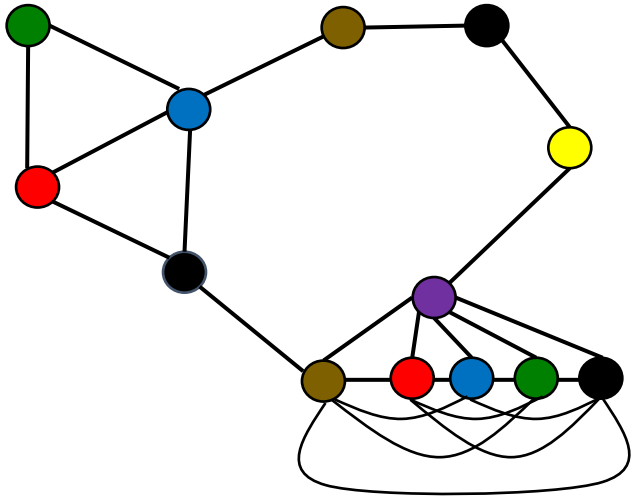
Unit Disk Graph.
Intersection graph of unit circles in the plane.



Introduction: Unit Disk Graphs

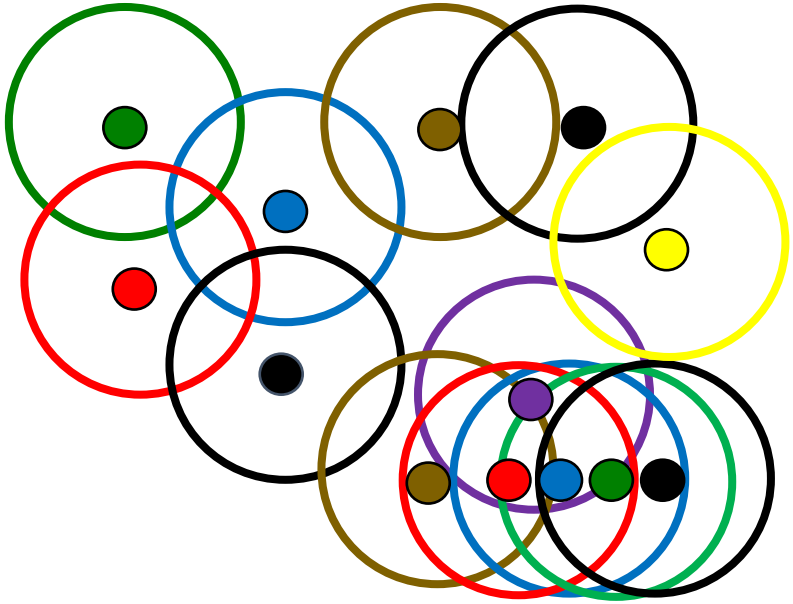
Unit Disk Graph.
Intersection graph of unit circles in the plane.



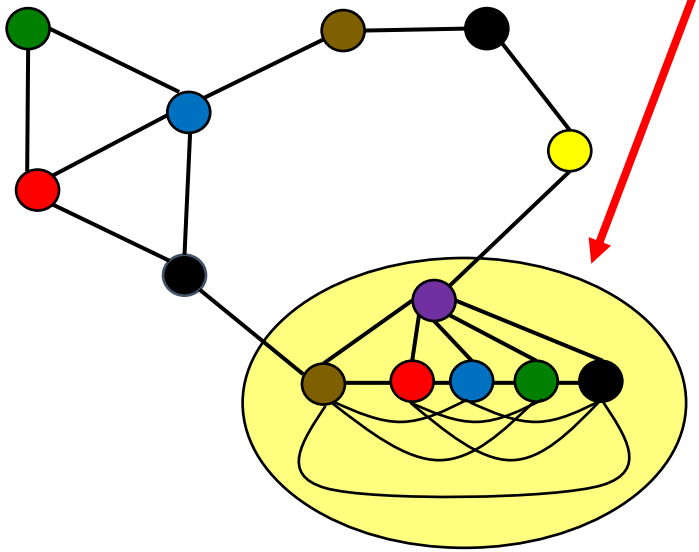


Introduction: Unit Disk Graphs

Unit Disk Graph.
Intersection graph of unit circles in the plane.



Large cliques! (Not planar.)



Introduction: Tree Decomposition

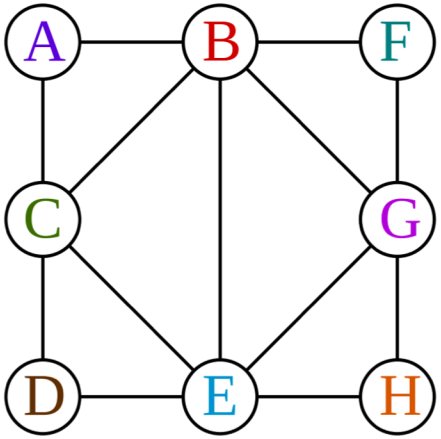
A **tree decomposition** of a graph G is a pair (T,β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$, such that

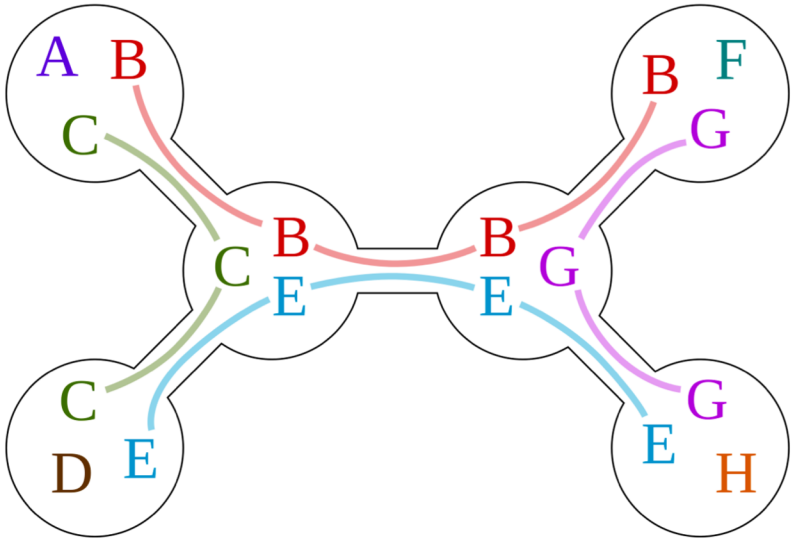
(i) For each $v \in V(G)$,

$$T[\{x \in V(T) : v \in \beta(x)\}]$$

is a non-empty tree.

(ii) For each $\{u,v\} \in E(G)$, there is $x \in V(T)$ such that $u,v \in \beta(x)$.





Introduction: Tree Decomposition

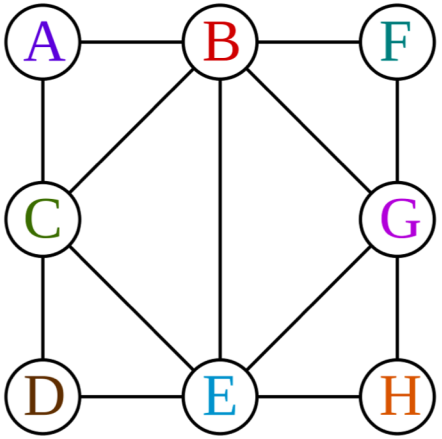
A **tree decomposition** of a graph G is a pair (T,β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$, such that

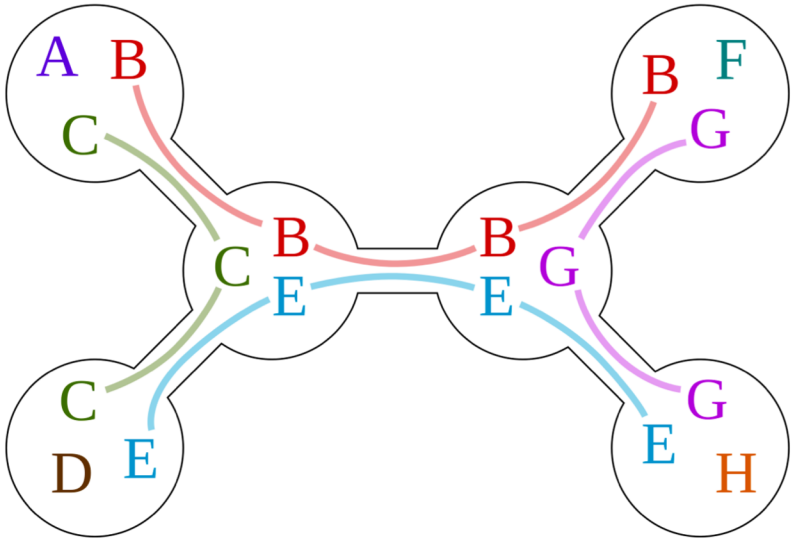
(i) For each $v \in V(G)$,

$$T[\{x \in V(T) : v \in \beta(x)\}]$$

is a non-empty tree.

(ii) For each $\{u,v\} \in E(G)$, there is $x \in V(T)$ such that $u,v \in \beta(x)$.





Width of (T,β). Size of the largest ***bag*** + 1.

Introduction: Tree Decomposition

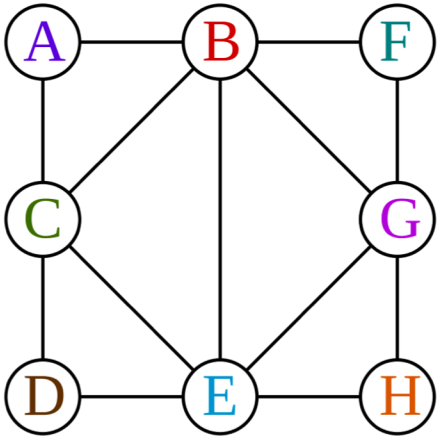
A **tree decomposition** of a graph G is a pair (T,β) where T is a tree and $\beta : V(T) \to 2^{V(G)}$, such that

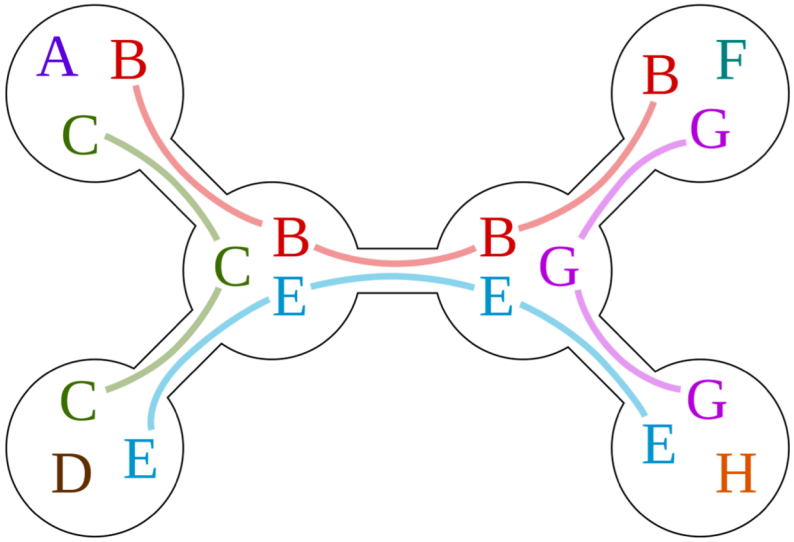
(i) For each $v \in V(G)$,

$$T[\{x \in V(T) : v \in \beta(x)\}]$$

is a non-empty tree.

(ii) For each $\{u,v\} \in E(G)$, there is $x \in V(T)$ such that $u,v \in \beta(x)$.





Treewidth of *G*. Minimum width of a tree decomposition of G.

Introduction: Branch on Cliques

Decomposition I(a). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w \cdot \gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w \cdot \gamma)$.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

We would like to utilize this theorem to reach the case where the treewidth is small.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

But the max. clique size γ can be large!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

But the max. clique size γ can be large!

(Unlike real life,) cliques are not a major issue!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. Branch to shrink the max. clique size!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. Branch to shrink the max. clique size!

Find a max. size clique C (polynomial time for UDGs).

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. Branch to shrink the max. clique size!

Find a max. size clique C (polynomial time for UDGs).

In VC (FVS), at most 1 (resp. 2) vertices of C can be unchosen. (**Branch!**)

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. Branch to shrink the max. clique size!

Find a max. size clique C (polynomial time for UDGs).

In Cycle Packing, there exists an optimal solution where at most $O(1)$ vertices of C are not used to pack local cycles. (**Branch!**)

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. Branch to shrink the max. clique size!

Find a max. size clique C (polynomial time for UDGs).

Number of branches: $\binom{|V(C)|}{c}$.

Parameter decrease: $|V(C)|$-c.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w \cdot \gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w \cdot \gamma)$.

But we can have large grids as minors!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

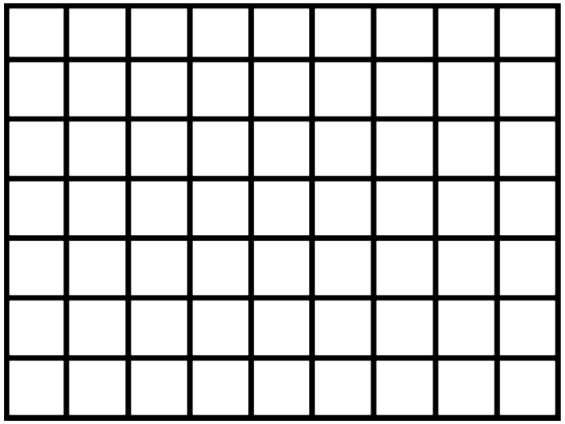
But we can have large grids as minors!

(Unlike real life,)
grids are great!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

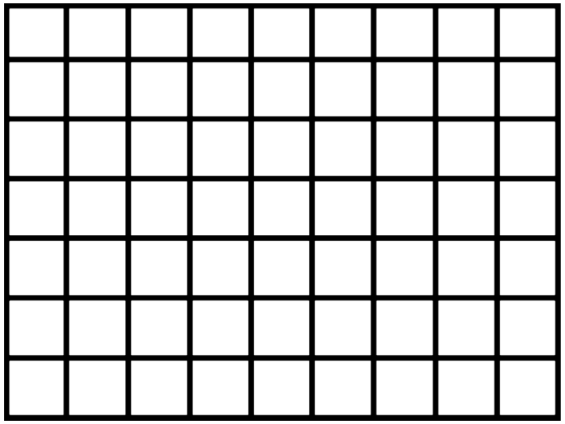
(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.



Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.

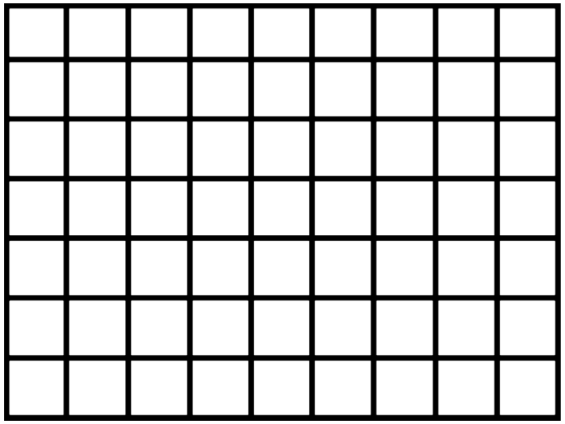


No!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

(Connected) Vertex Cover, Feedback Vertex Set, Cycle Packing. A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.



Yes!

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

Case Distinction:

1. Clique of size k^ε? Branch, parameter decreases by ~k^ε. (No such clique? $\gamma=O(k^\varepsilon)$.)
2. Grid minor of size $c\sqrt{k}$x$c\sqrt{k}$? Yes/No. (No such grid? tw(G) $=O(\sqrt{k}\cdot\gamma)$.)
3. Treewidth $O(k^{0.5+\varepsilon})$. Solve in time $O^*(2^{O(k^{0.5+\varepsilon})})$ or $O^*(2^{O(k^{0.5+\varepsilon}\log k)})$.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

Case Distinction: **Tradeoff**

1. Clique of size k^ε? Branch, parameter decreases by **~k^ε**. (No such clique? $\gamma=O(k^\varepsilon)$.)
2. Grid minor of size $c\sqrt{k}$x$c\sqrt{k}$? Yes/No. (No such grid? tw(G) $=O(\sqrt{k}\cdot\gamma)$.)
3. Treewidth $O(k^{0.5+\varepsilon})$. Solve in time $\boldsymbol{O^*(2^{O(k^{0.5+\varepsilon})})}$ **or** $\boldsymbol{O^*(2^{O(k^{0.5+\varepsilon}\log k)})}$.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

Case Distinction:

1. Clique of size $k^{0.25}$? Branch, param. decreases by ~$k^{0.25}$. (No such clique? $\gamma=O(k^{0.25})$.)
2. Grid minor of size $c\sqrt{k}$x$c\sqrt{k}$? Yes/No. (No such grid? tw(G) $=O(\sqrt{k}\cdot\gamma)$.)
3. Treewidth $O(k^{0.75})$. Solve in time $O^*(2^{O(k^{0.75})})$ or $O^*(2^{O(k^{0.75}\log k)})$.

Introduction: Branch on Cliques

Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w\cdot\gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\gamma)$.

Case Distinction: **Time $O^*(2^{O(k^{0.75})})$ or $O^*(2^{O(k^{0.75}\log k)})$.**

1. Clique of size $k^{0.25}$? Branch, param. decreases by ~$k^{0.25}$. (No such clique? $\gamma=O(k^{0.25})$.)
2. Grid minor of size $c\sqrt{k}$x$c\sqrt{k}$? Yes/No. (No such grid? tw(G) $=O(\sqrt{k}\cdot\gamma)$.)
3. Treewidth $O(k^{0.75})$. Solve in time $O^*(2^{O(k^{0.75})})$ or $O^*(2^{O(k^{0.75}\log k)})$.

Introduction: Branch on Cliques

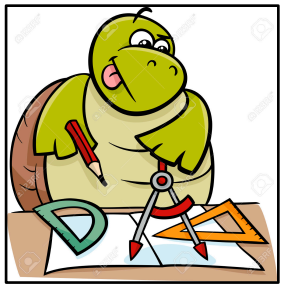
Decomposition I(b). Given w and a UDG G of max. clique size γ, in time $O^*(2^{O(w \cdot \gamma)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w \cdot \gamma)$.

What about…

- Better running time?
- Problems where the existence of a large grid does not seem useful?

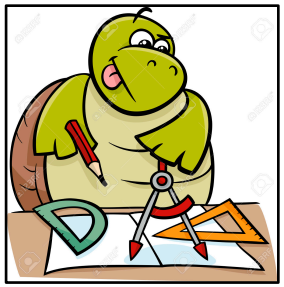
Outline

- Introduction
- **Decomposition II: General Decomposition**
- Decomposition III: Contraction Decomposition



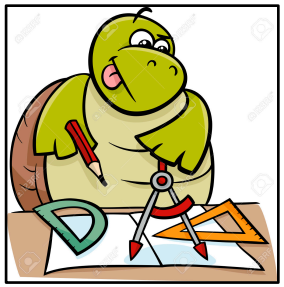
Decomposition II(a): Statement

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).



Decomposition II(a): Clique-Grid Graph

Unit disk graph *G* → clique-grid graph *G'*.

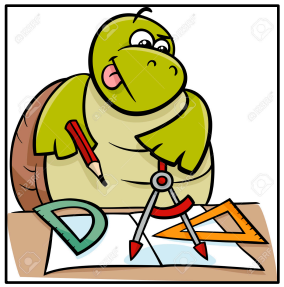


Decomposition II(a): Clique-Grid Graph

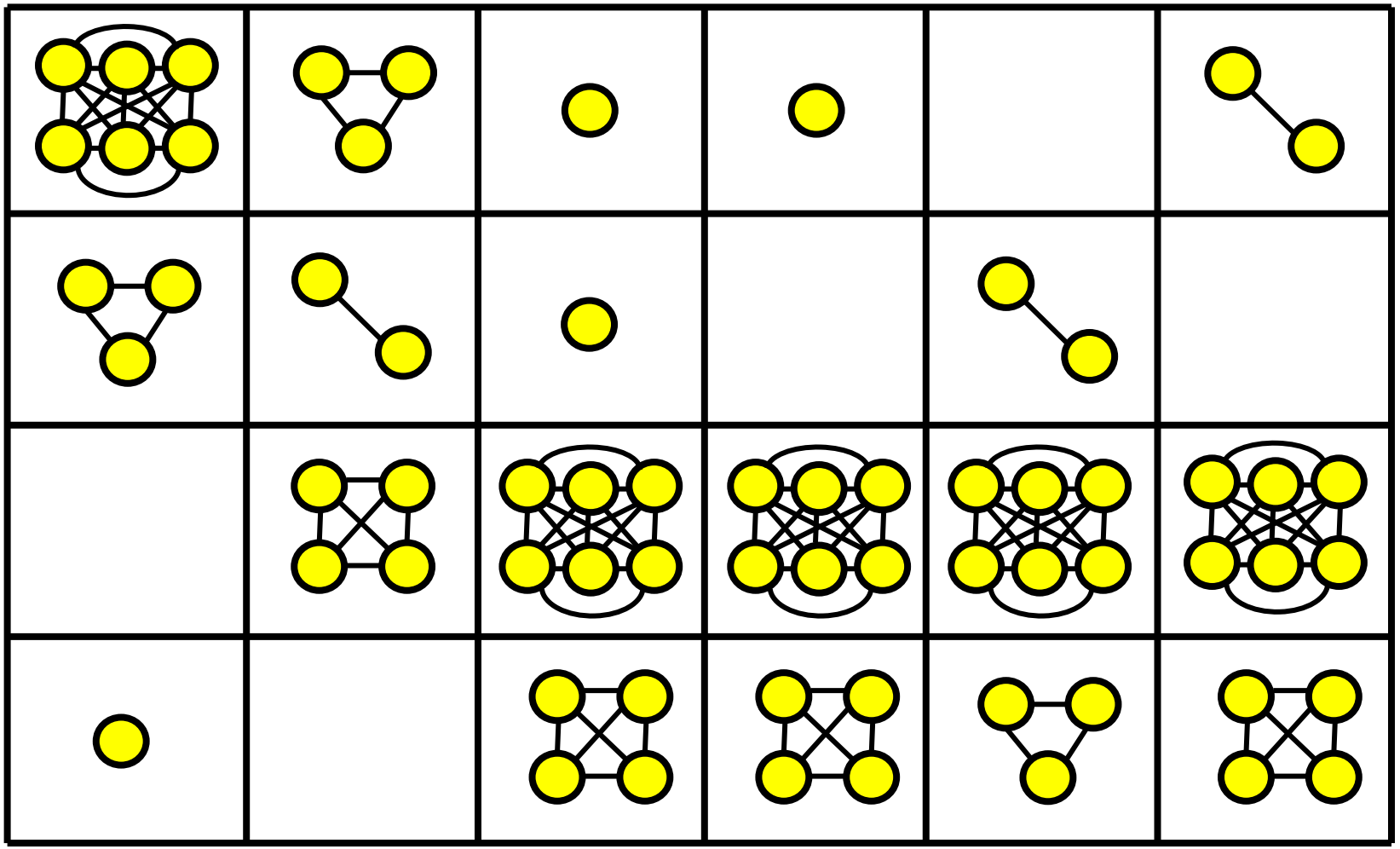
Unit disk graph G → clique-grid graph G'.

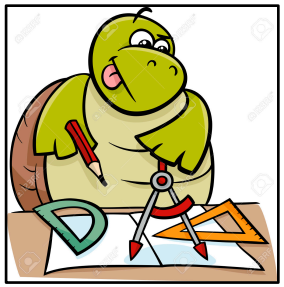
Clique-grid graph. There is f: $V(G) \rightarrow [t]\text{x}[t']$ such that

1. For all $(i,j) \in [t]\text{x}[t']$, $f^{-1}(i,j)$ is a clique.
2. For all $\{u,v\} \in E(G)$, $|i-i'| \leq 2$ and $|j-j'| \leq 2$, where $f(u)=(i,j)$ and $f(v)=(i',j')$.

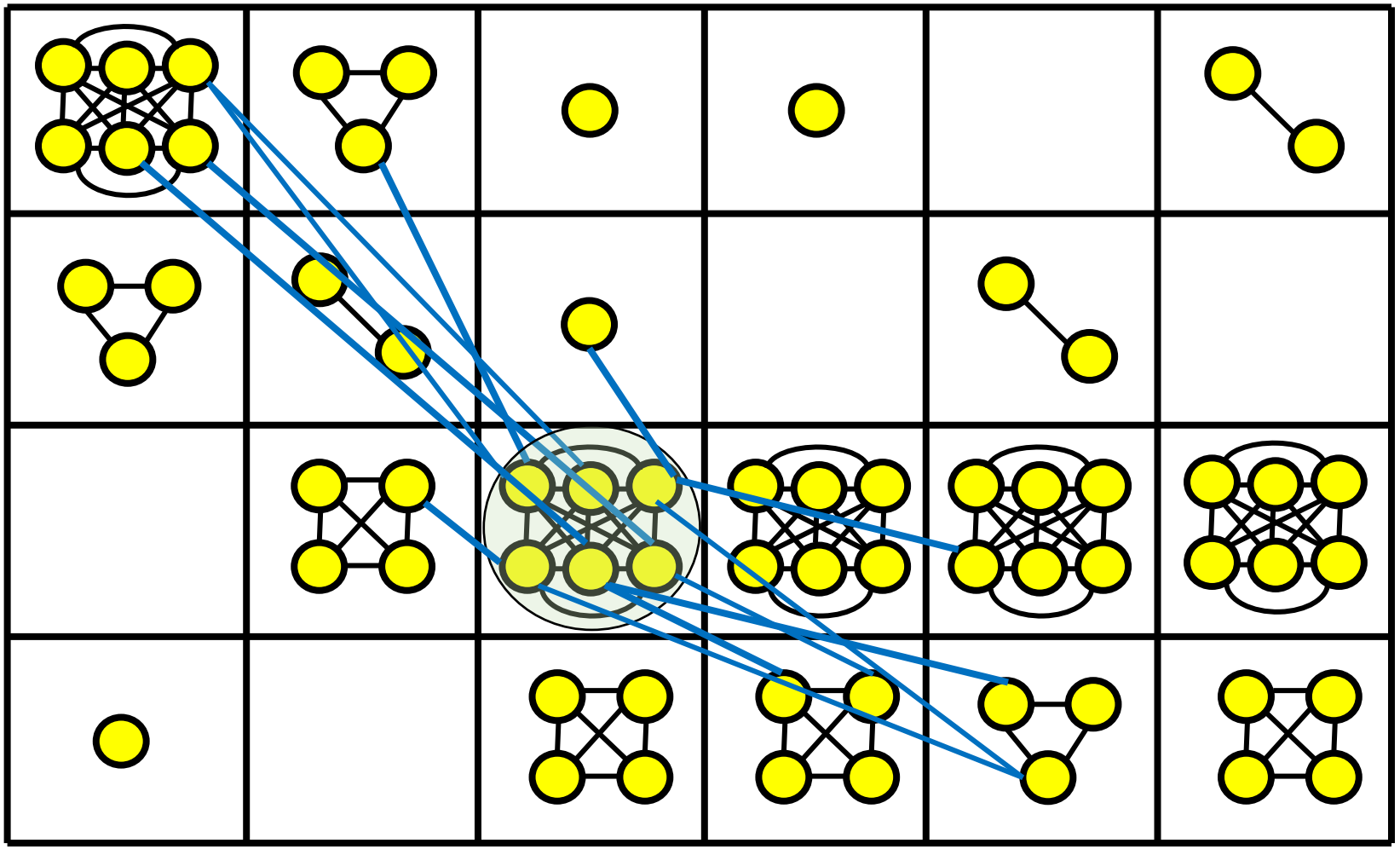


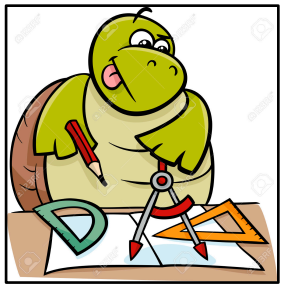
Decomposition II(a): Clique-Grid Graph



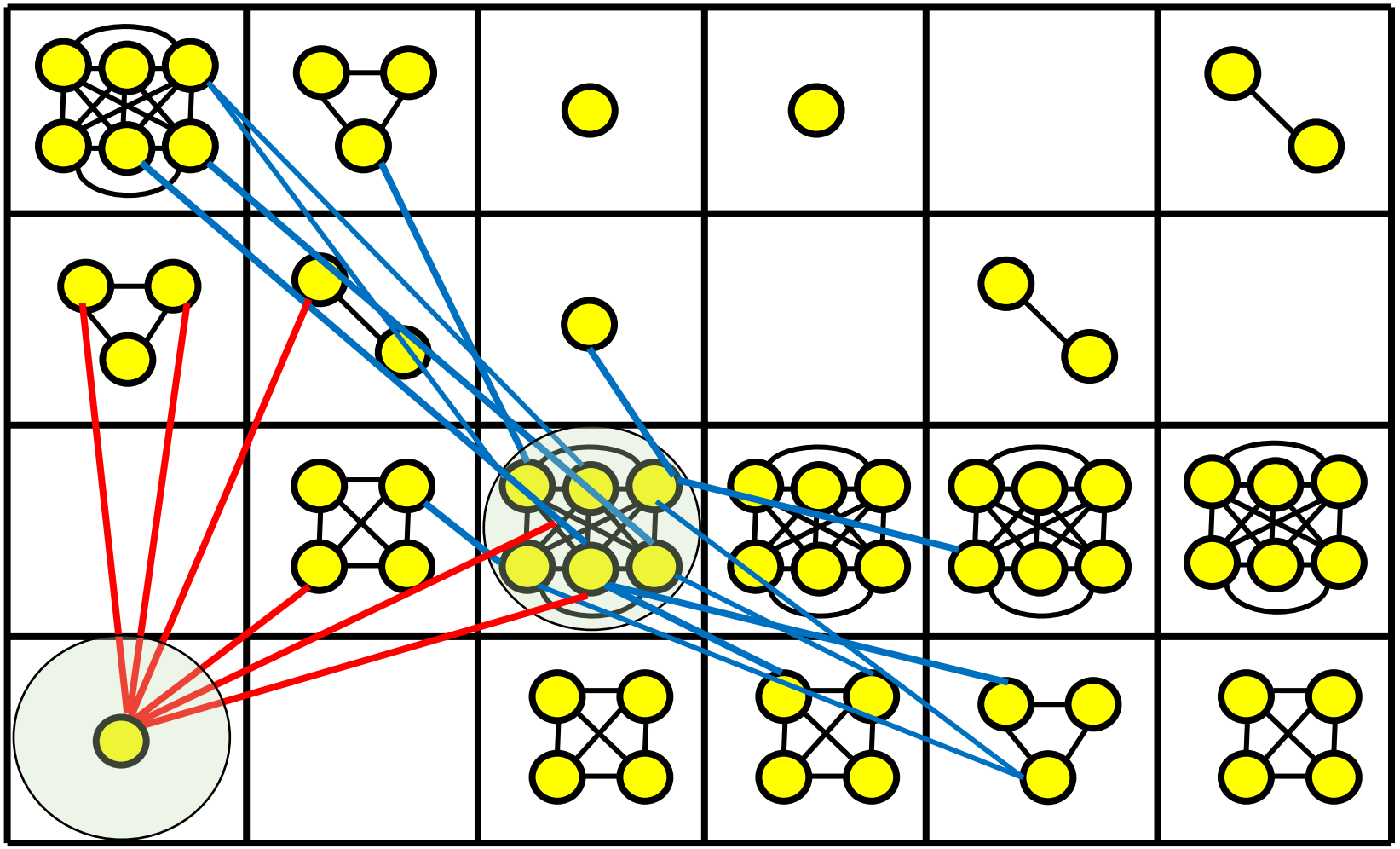


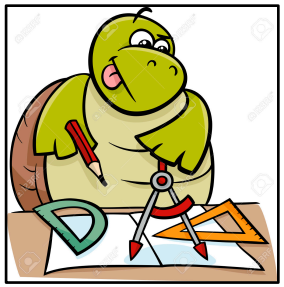
Decomposition II(a): Clique-Grid Graph





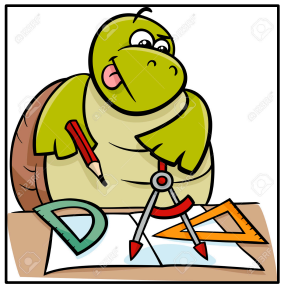
Decomposition II(a): Clique-Grid Graph





Decomposition II(a): Backbone Graph

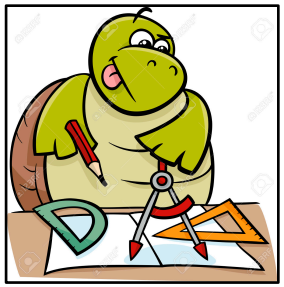
Unit disk graph *G* → clique-grid graph *G'* → backbone graph *B*.



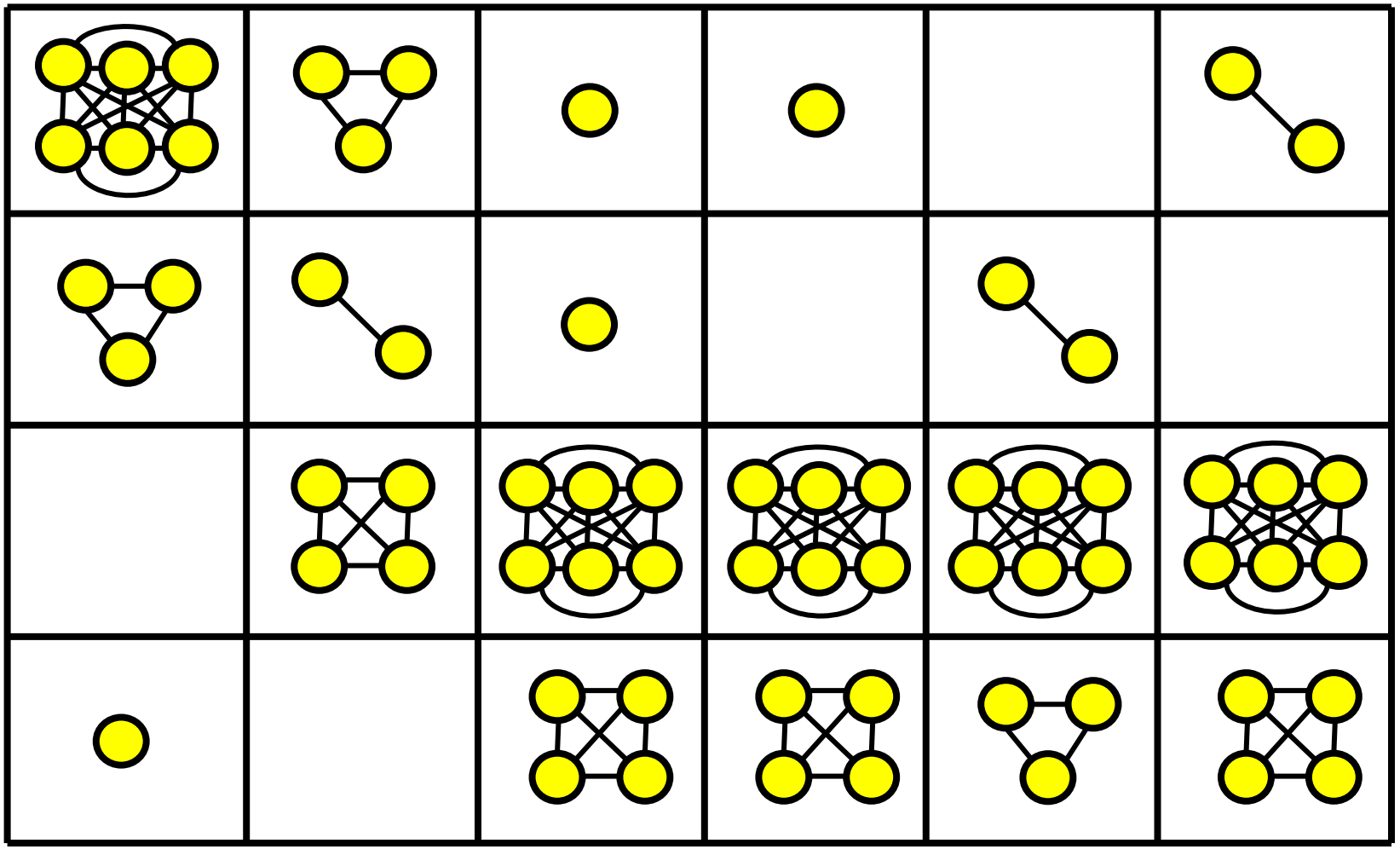
Decomposition II(a): Backbone Graph

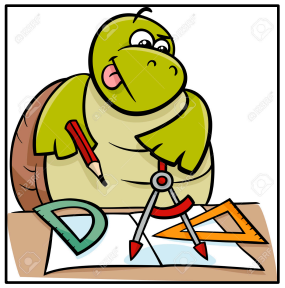
Unit disk graph *G* → clique-grid graph *G'* → backbone graph *B*.

Backbone graph (informal). Keep minimum number of vertices such that any two cells that have adjacent vertices in *G* also have adjacent vertices in *B*.

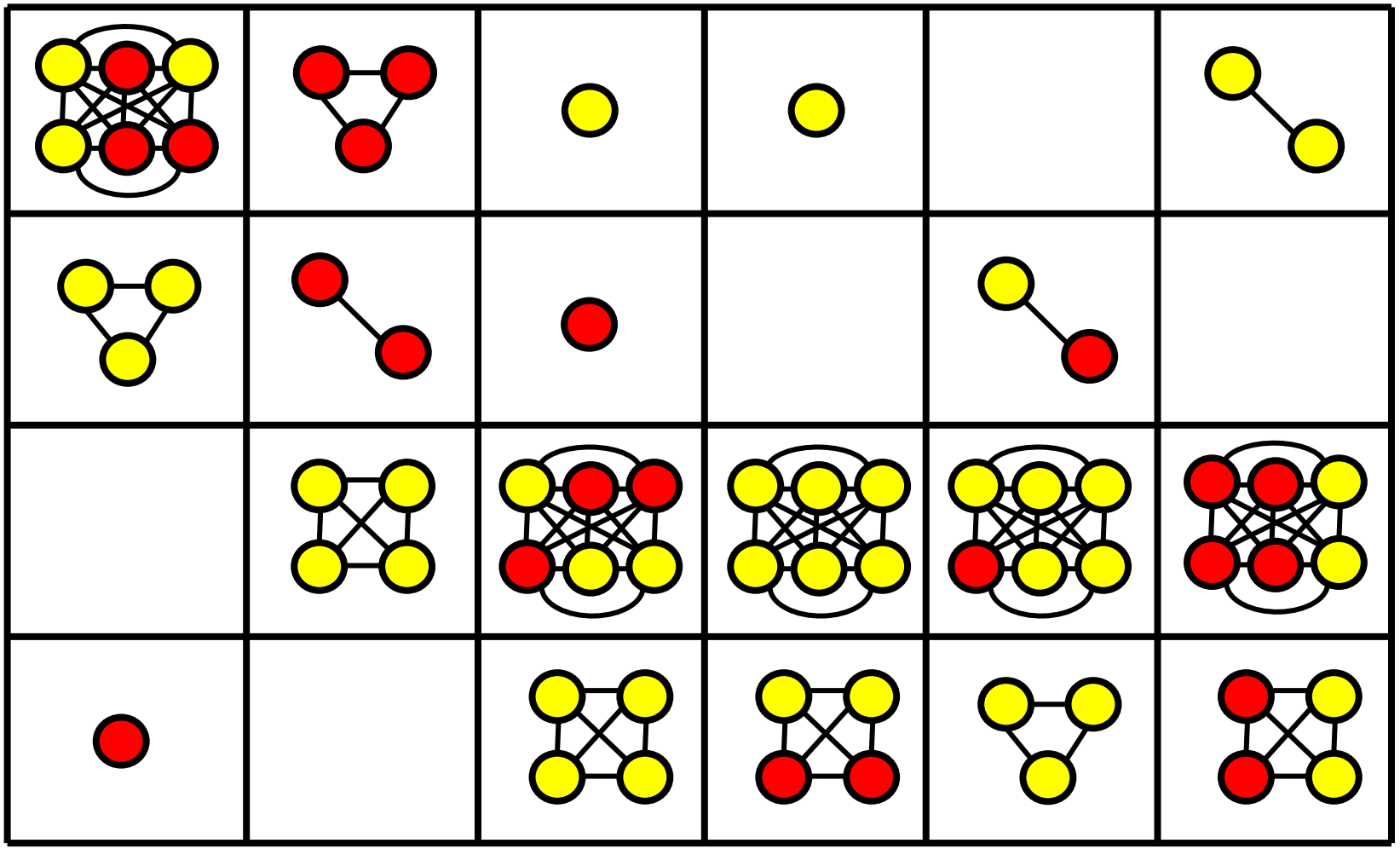


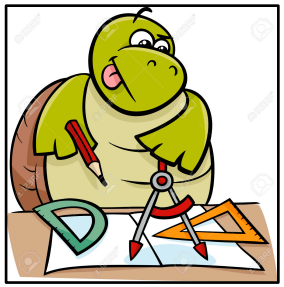
Decomposition II(a): Clique-Grid Graph



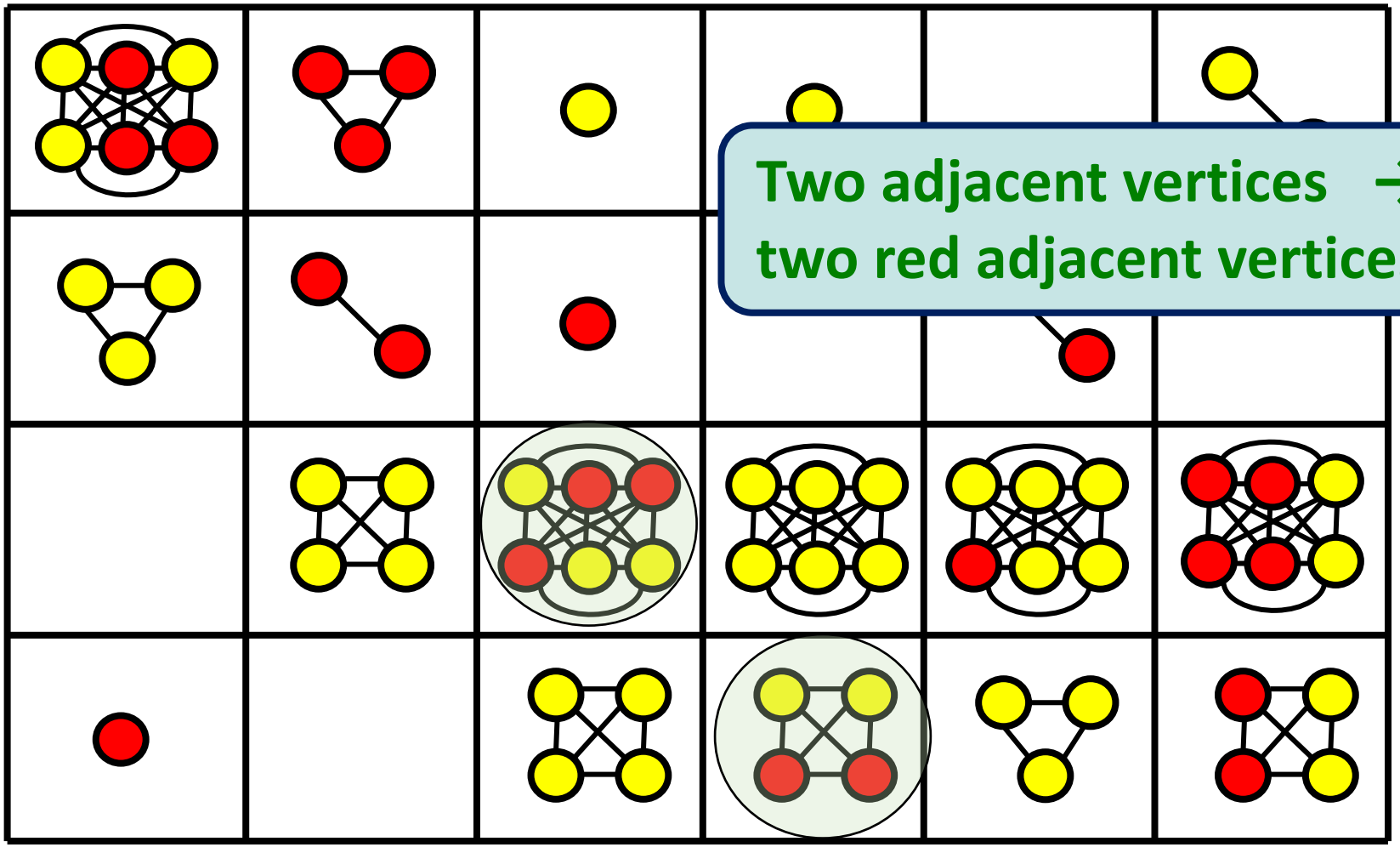


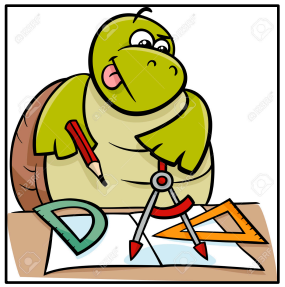
Decomposition II(a): Clique-Grid Graph



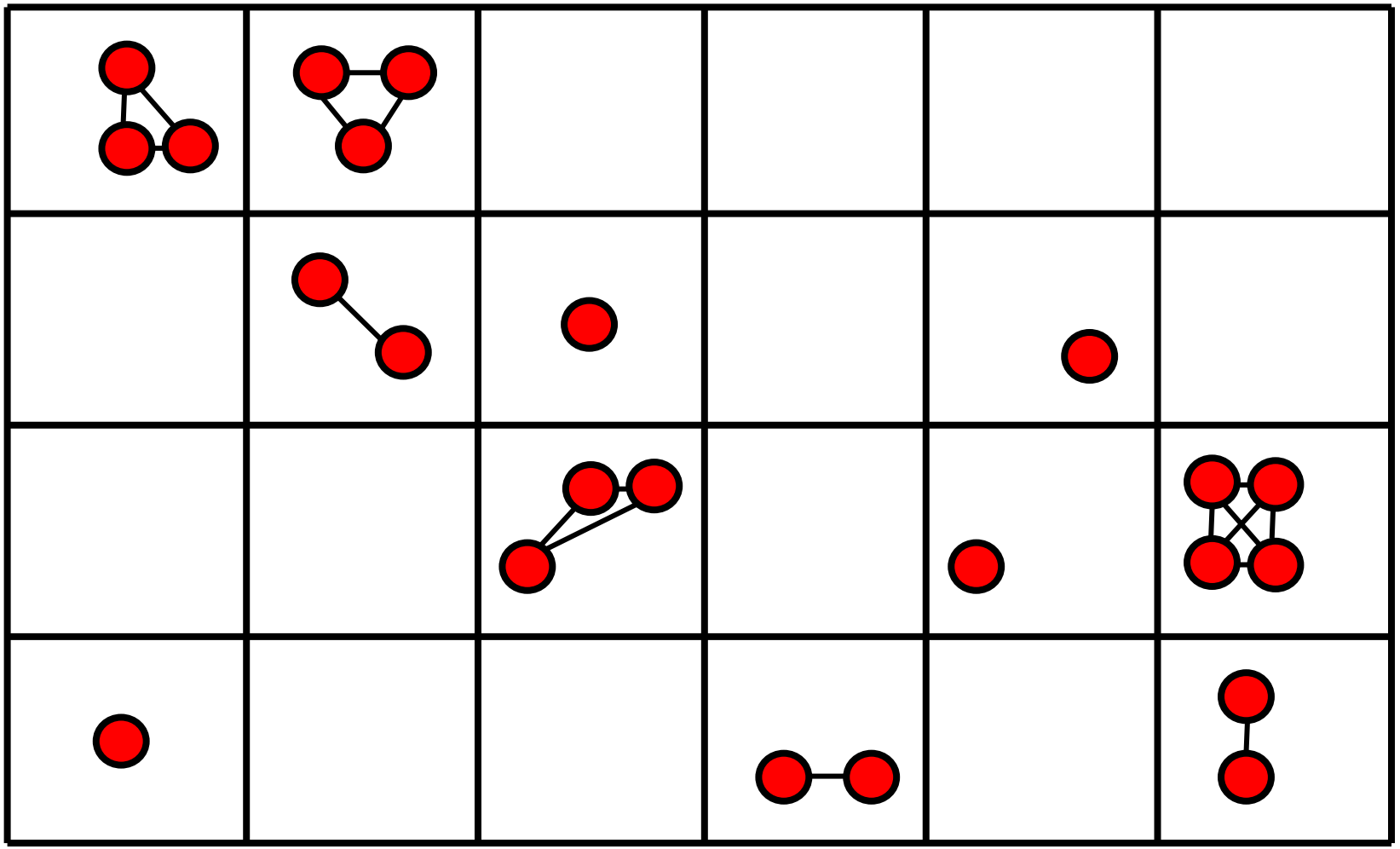


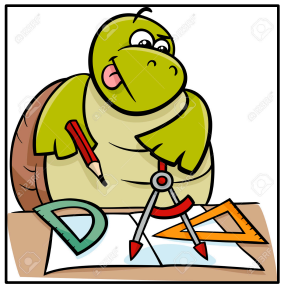
Decomposition II(a): Clique-Grid Graph



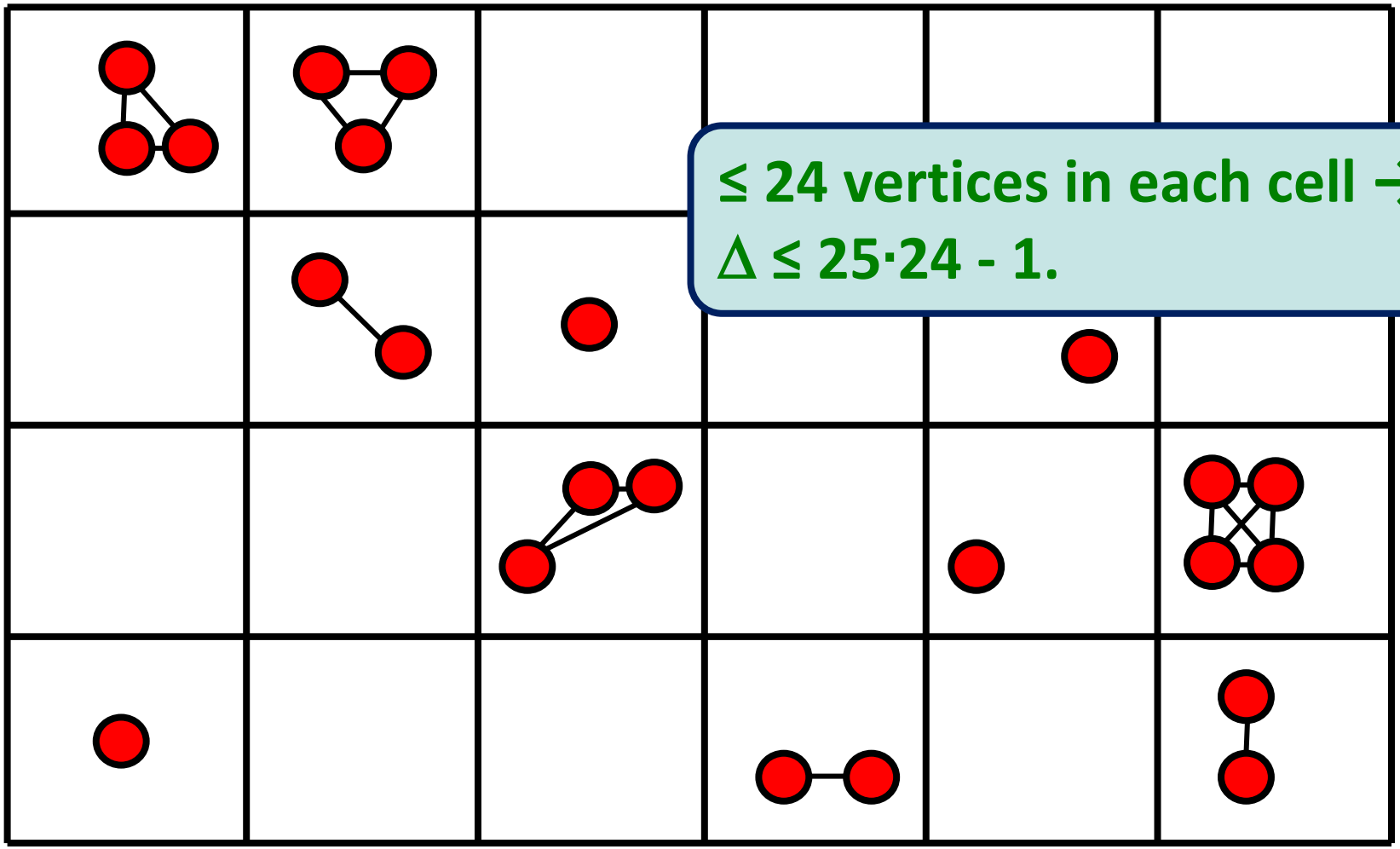


Decomposition II(a): Backbone Graph





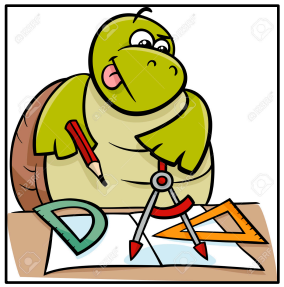
Decomposition II(a): Backbone Graph



≤ 24 vertices in each cell →
Δ ≤ 25·24 - 1.

Introduction: Branch on Cliques

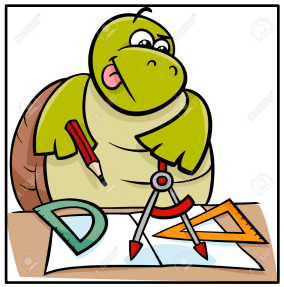
Decomposition I(b). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.



Decomposition II(a): Proof Sketch

Decomposition I(b). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.

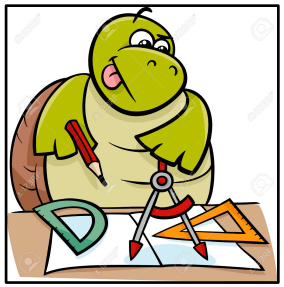
The backbone graph we compute for a UDG G is a UDG in itself, and its max. degree is bounded by a fixed constant.



Decomposition II(a): Proof Sketch

Decomposition I(b). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.

So in time $O^*(2^{O(w)})$... We either know that B has a wxw grid minor or output a tree decomp. of B of width $O(w)$.

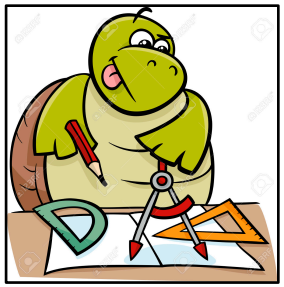


Decomposition II(a): Proof Sketch

Decomposition I(b). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.

So in time $O^*(2^{O(w)})$... We either know that G has a wxw grid minor or output a tree decomp. of B of width $O(w)$.

B is a subgraph of G.

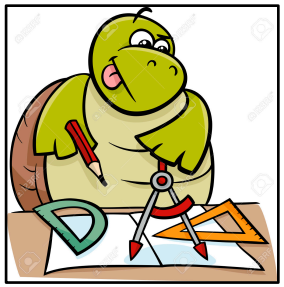


Decomposition II(a): Proof Sketch

Decomposition I(b). Given w and a UDG G of max. degree Δ, in time $O^*(2^{O(w\cdot\Delta)})$ either determine that G has a wxw grid minor or output a tree decomp. of width $O(w\cdot\Delta)$.

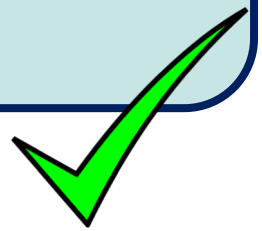
So in time $O^*(2^{O(w)})$... We either know that G has a wxw grid minor or output a tree decomp. of G of width $O(w)$.

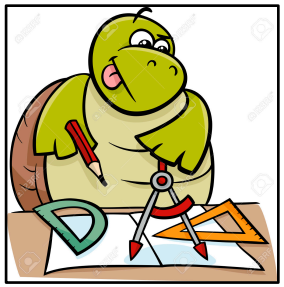
Replace every vertex of B by all the vertices in its cell. (Not difficult to verify that this results in a tree decomp. of G.)



Decomposition II(a): Statement

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

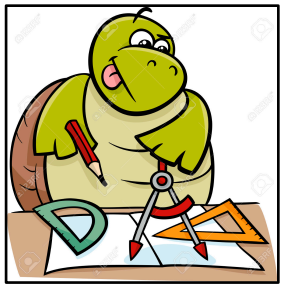




Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

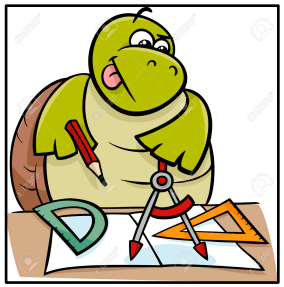


Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.

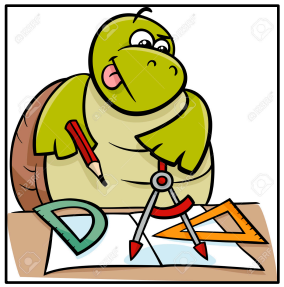


Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
  (Easy for VC/FVS/Cycle Pack./Long Cycle/Exact Cycle).

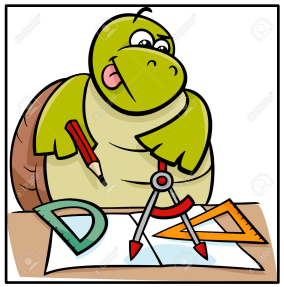


Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.



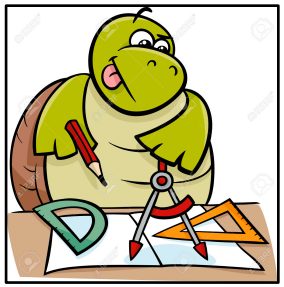
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.

(Does not work for Exact Cycle!)

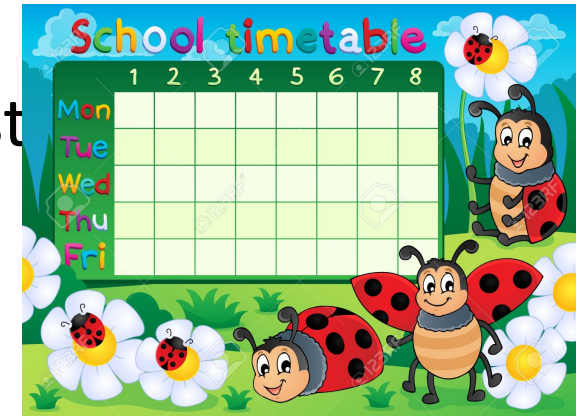


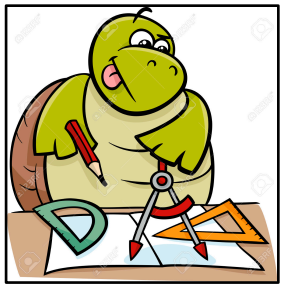
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-inst
- Special tree decomposition? DP.





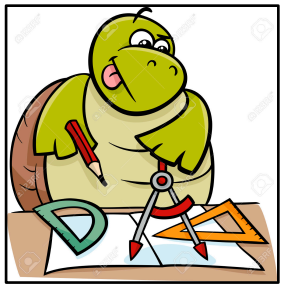
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

Easy for VC/FVS: In each bag (of size $O(k^{1.5})$), only $O(\sqrt{k})$ may not be chosen.



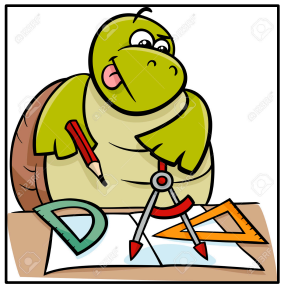
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

Easy for VC/FVS: Time $\boldsymbol{O^*(2^{O(k^{0.5}\log k)})}$.



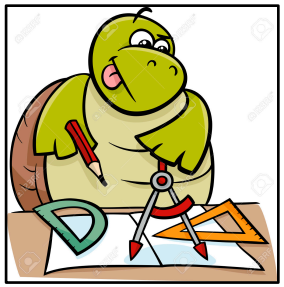
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

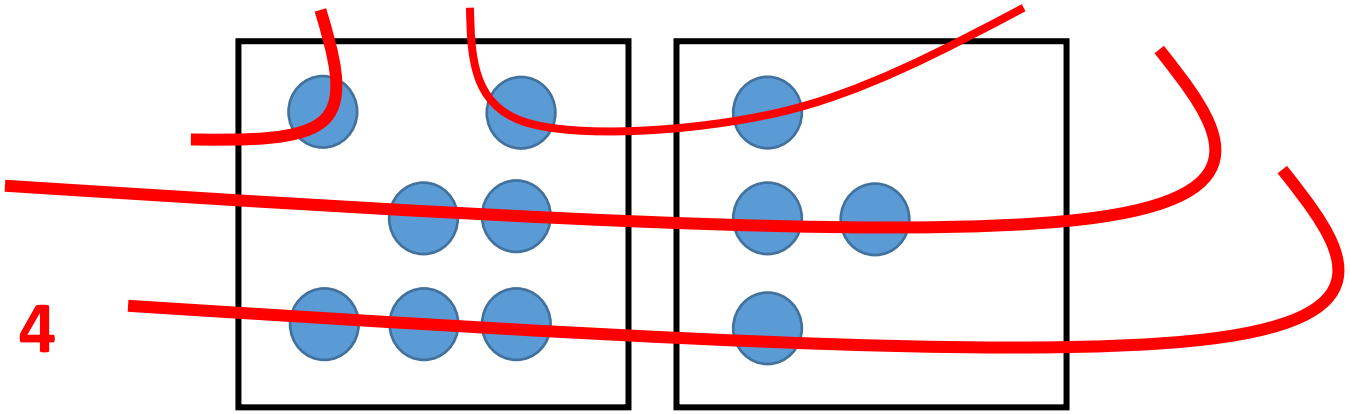
What about Cycle Packing and Long Cycle/Path?

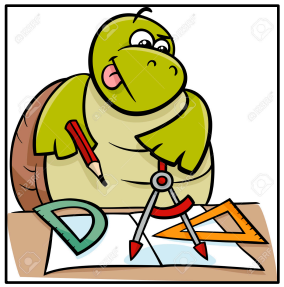


Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Cycle Packing. How many cycles ``pass'' through two cliques?



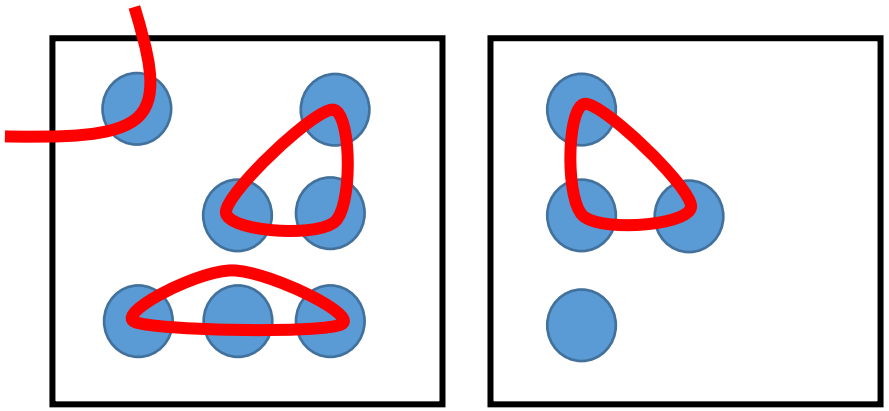


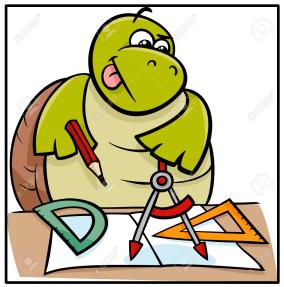
Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Cycle Packing. How many cycles ``pass'' through two cliques?

4

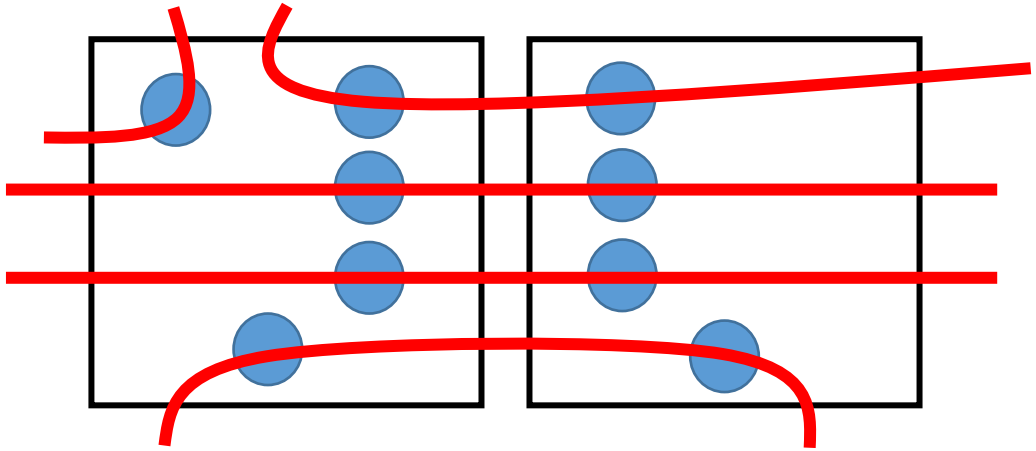




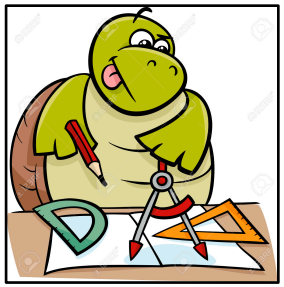
Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Cycle Packing. How many cycles ``pass'' through two cliques?



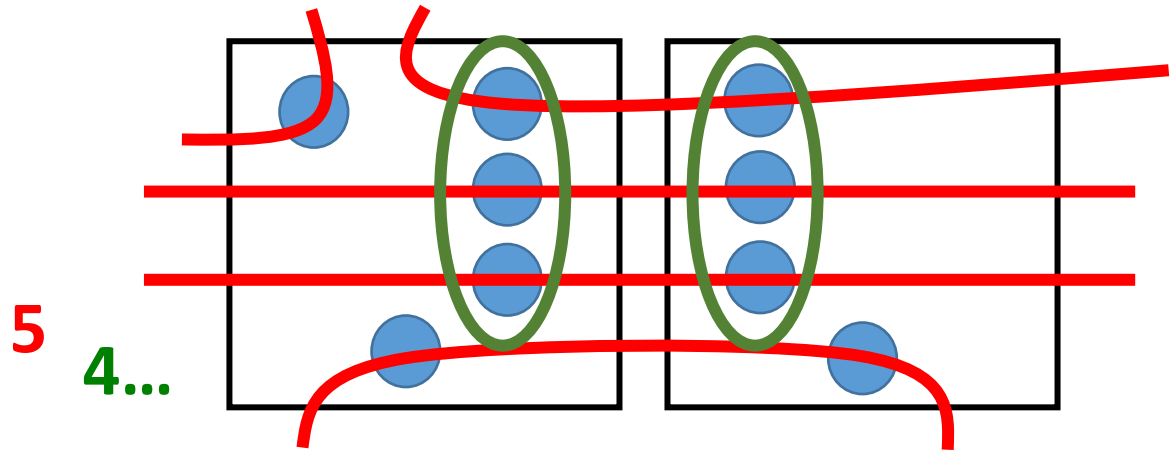
5

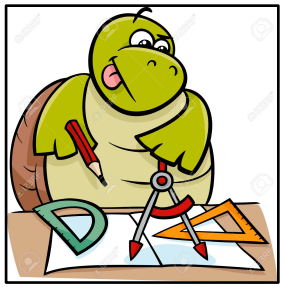


Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Cycle Packing. How many cycles ``pass'' through two cliques?

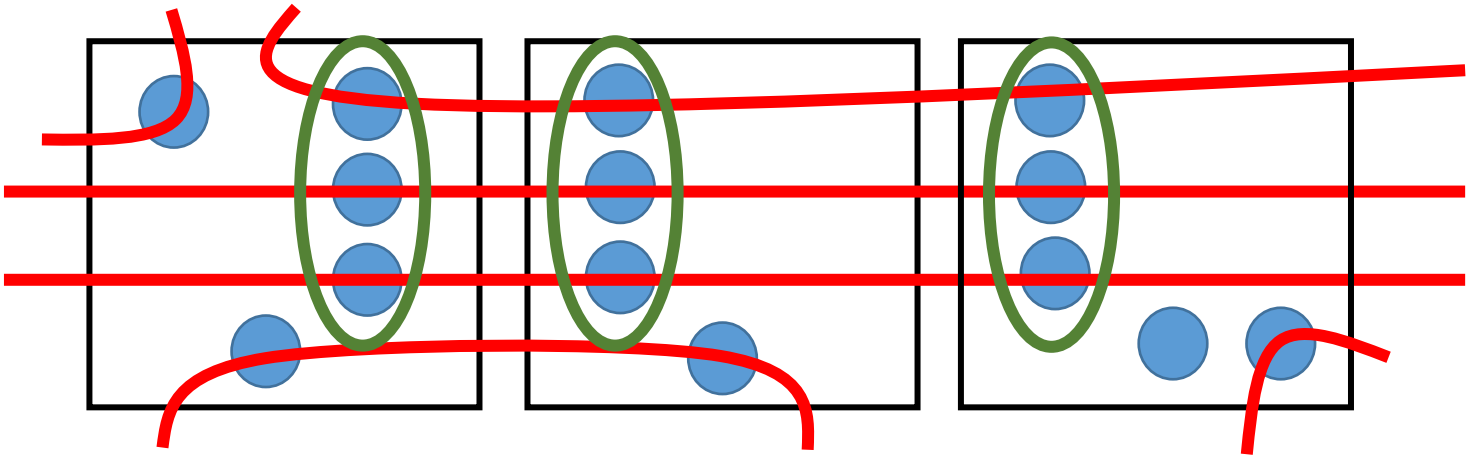


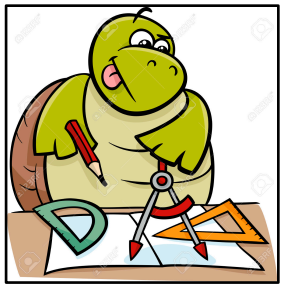


Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Cycle Packing. How many cycles ``pass'' through two cliques?

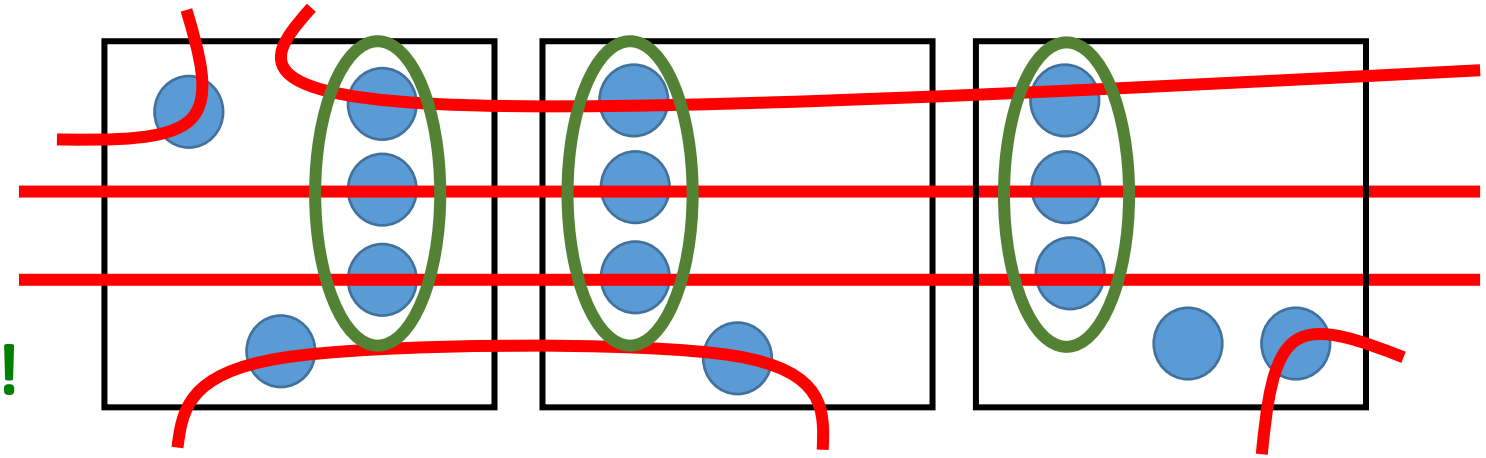


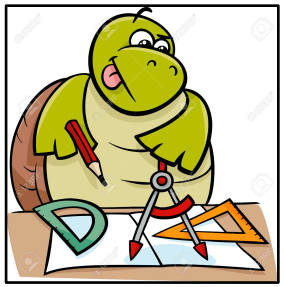


Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by *O*(1).

Cycle Packing. Every cycle that passes through a cell uses at most two of its vertices, so the interaction is *O*(1).

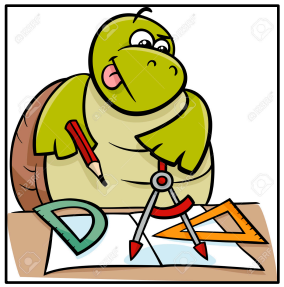




Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Long Cycle/Path and Exact Cycle. We also have an exchange argument that bounds the interaction between any two cells.

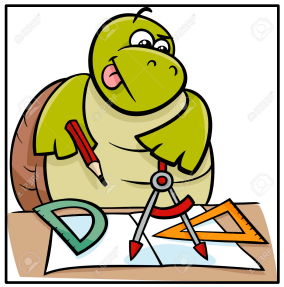


Decomposition II(a): Applications

Existence of a solution where the interaction between any two cells is bounded by $O(1)$.

Long Cycle/Path and Exact Cycle. We also have an exchange argument that bounds the interaction between any two cells.

Note. The exchange is done in a way that ensures bounded interaction for all pairs of cells simultaneously.



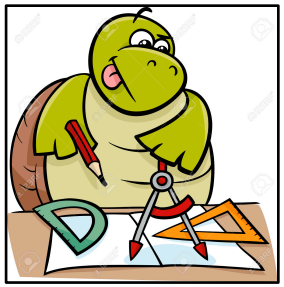
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

What about Cycle Packing and Long Cycle/Path?



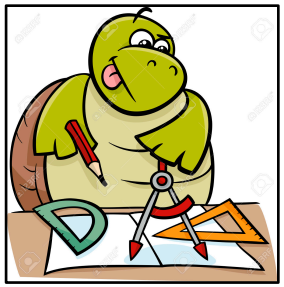
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

Cycle Packing/Long Cycle/Path: In each bag (of size $O(k^{1.5})$), only $O(\sqrt{k})$ are ``reserved'' for the future.



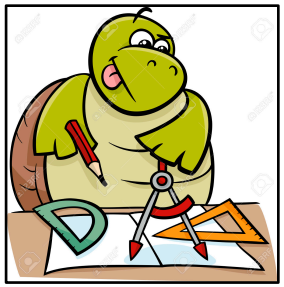
Decomposition II(a): Applications

Decomposition II(a). Given w and a UDG G, in time $O^*(2^{O(w)})$ either determine that G has a wxw grid minor or compute a tree decomposition where each bag consists of $O(w)$ ``cells'' (cliques).

How to use this decomposition?

- Bound the max clique size by $O(k)$.
- A $c\sqrt{k}$x$c\sqrt{k}$ grid minor? Yes/No-instance.
- Special tree decomposition? DP.

Cycle Packing/Long Cycle/Path: Time $\boldsymbol{O^*(2^{O(k^{0.5}\log k)})}$.

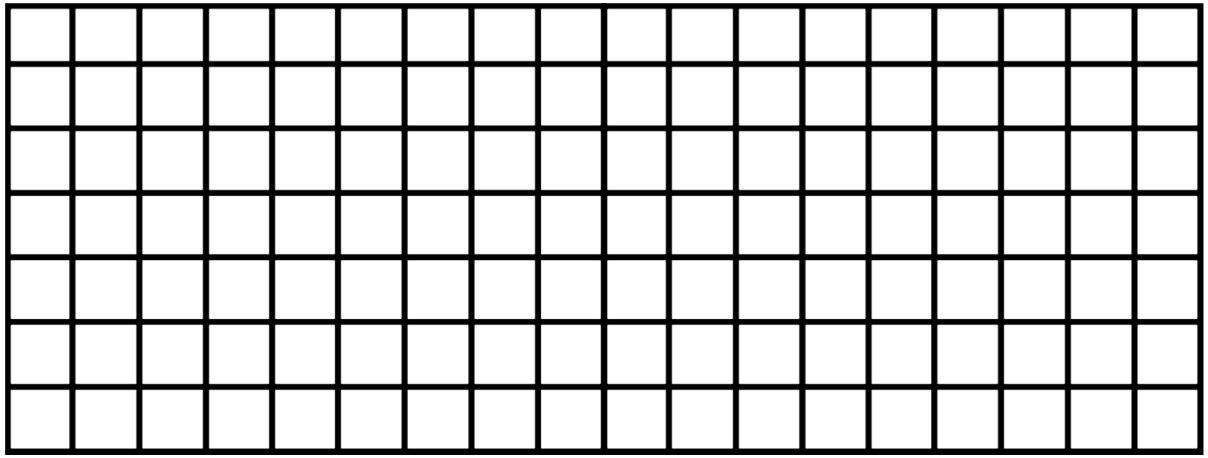


Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

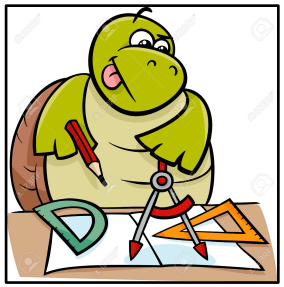
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).

1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1



Exact Cycle

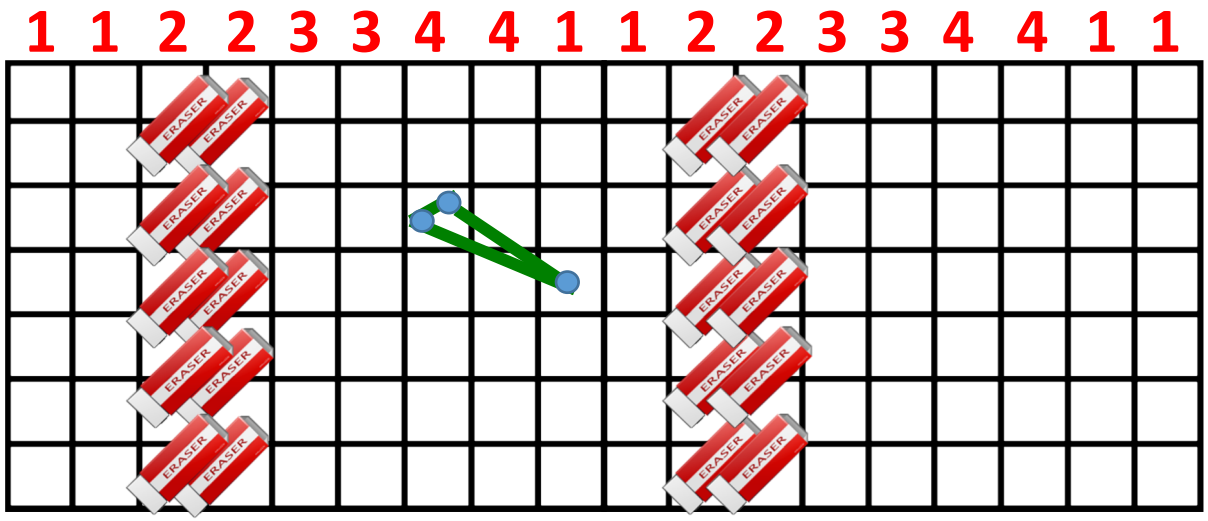
k=3



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

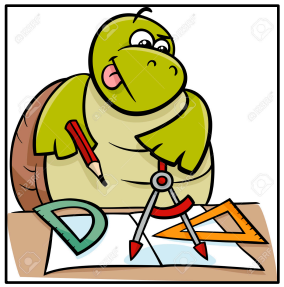
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Exact Cycle

***k*=3**

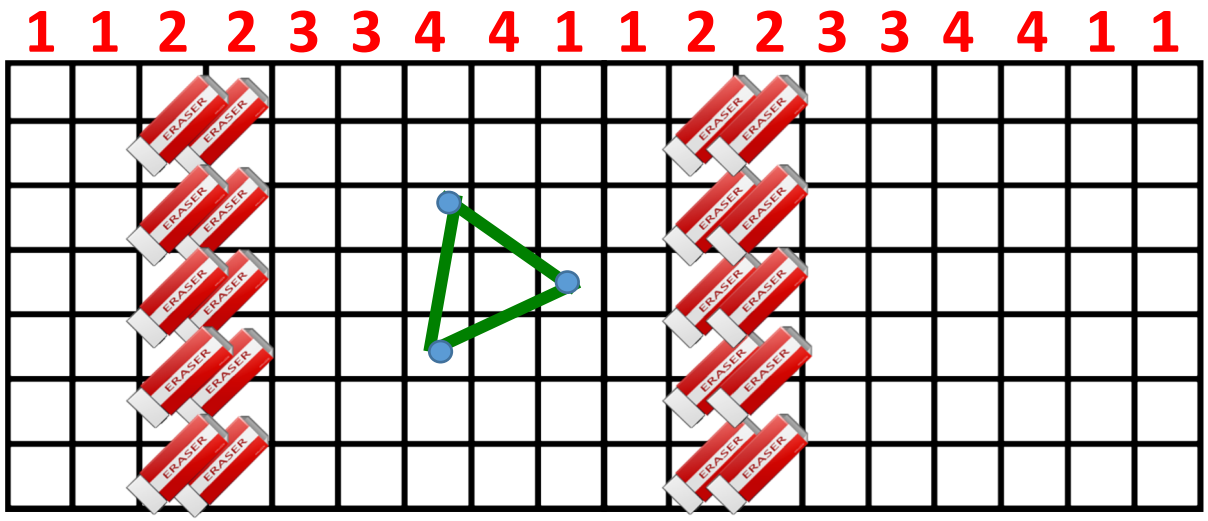
Deleting 2 (or 3) is fine.



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

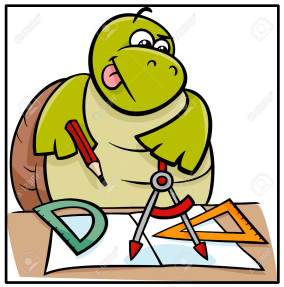
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Exact Cycle

***k*=3**

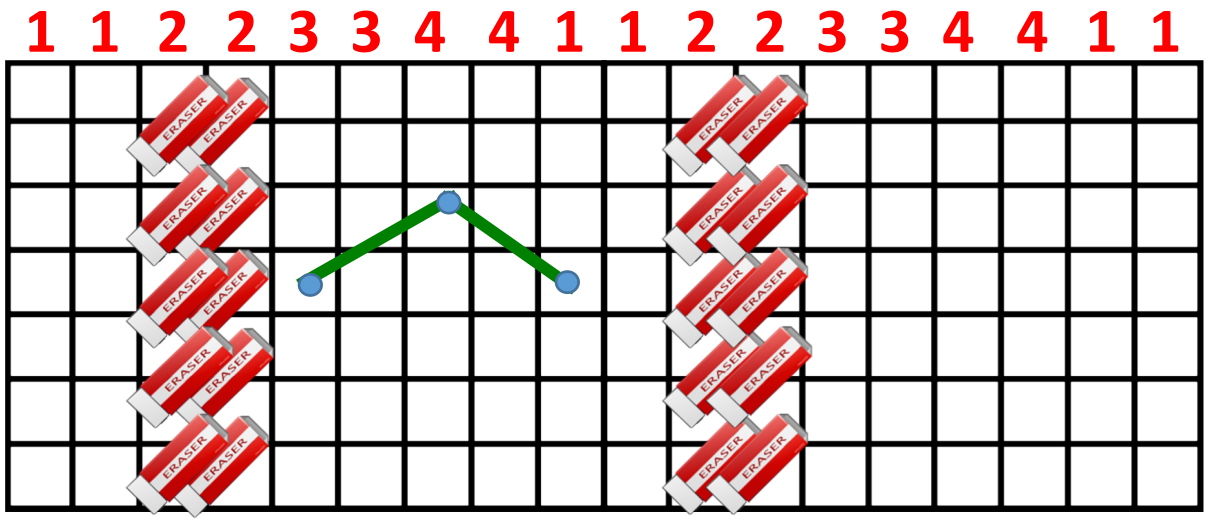
Deleting 2 (or 3) is fine.



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

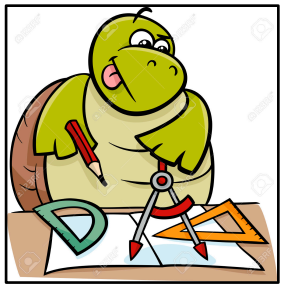
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Exact Path

***k*=3**

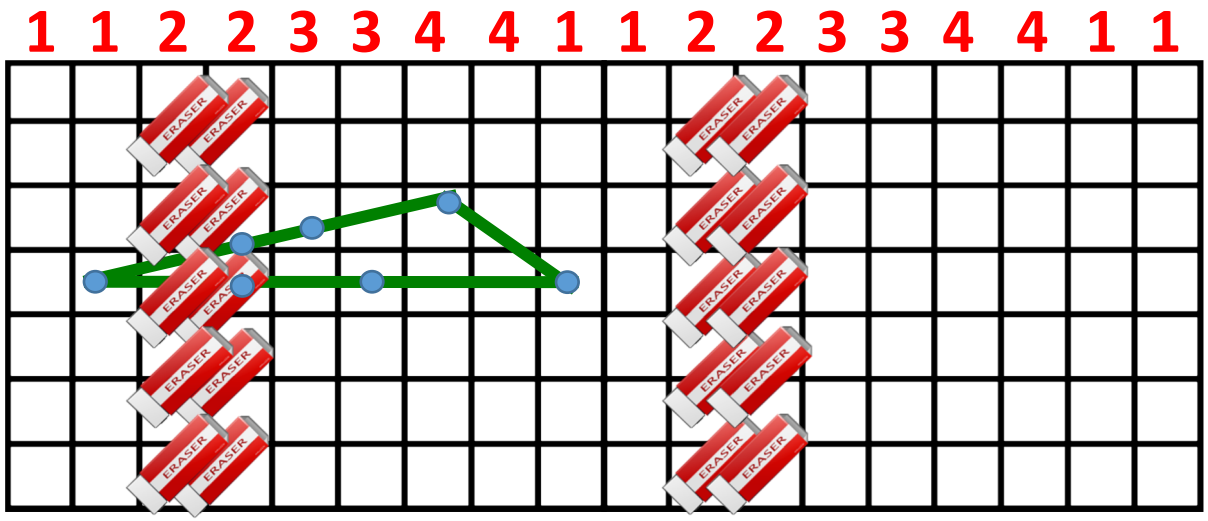
Deleting 2 is fine.



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

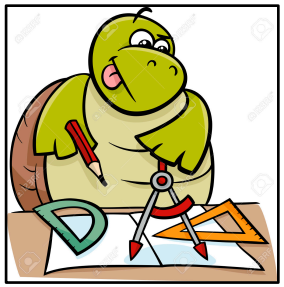
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Long Cycle?

k=3

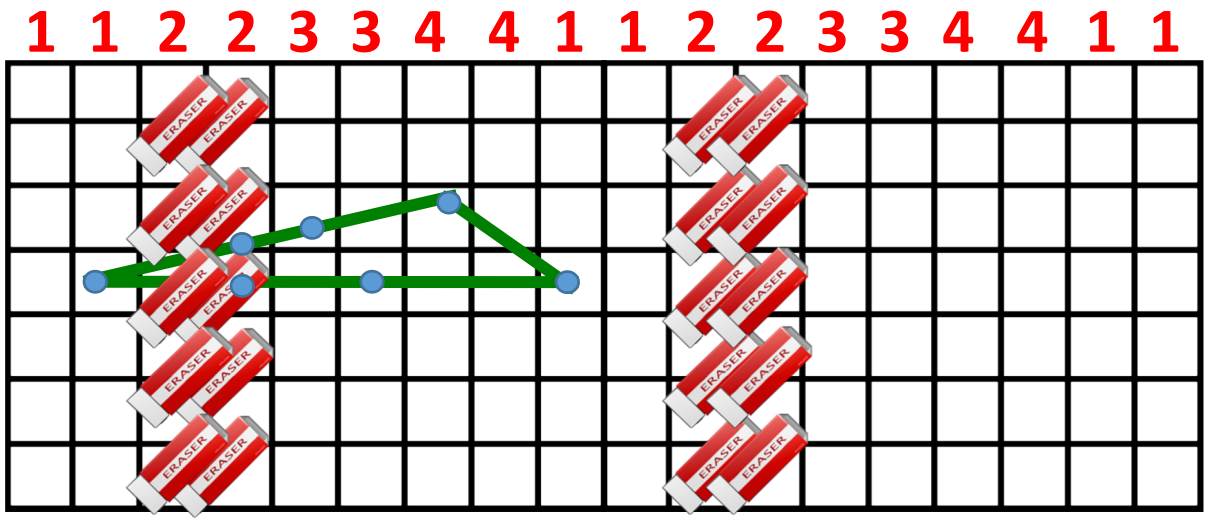
???



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

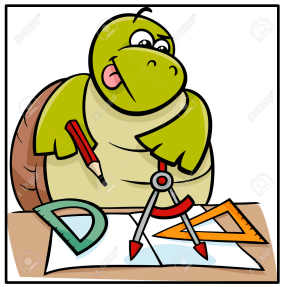
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Long Cycle

k=3

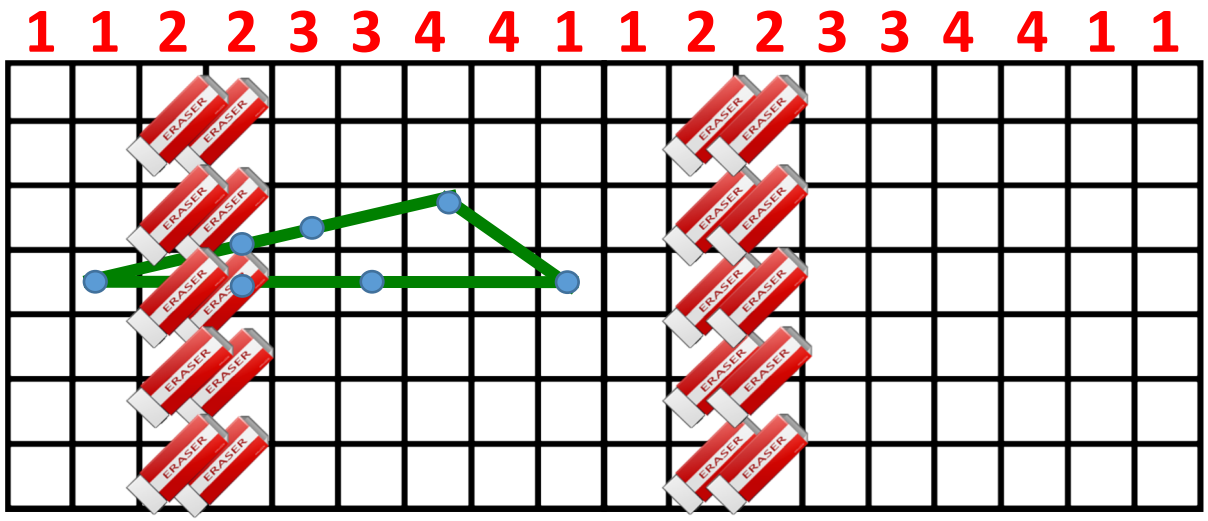
With a minor (known) trick, can handle this case.



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

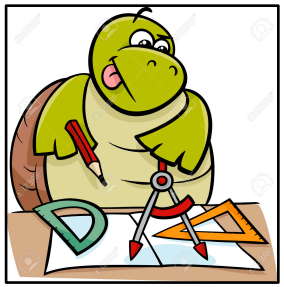
1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



FVS?

***k*=3**

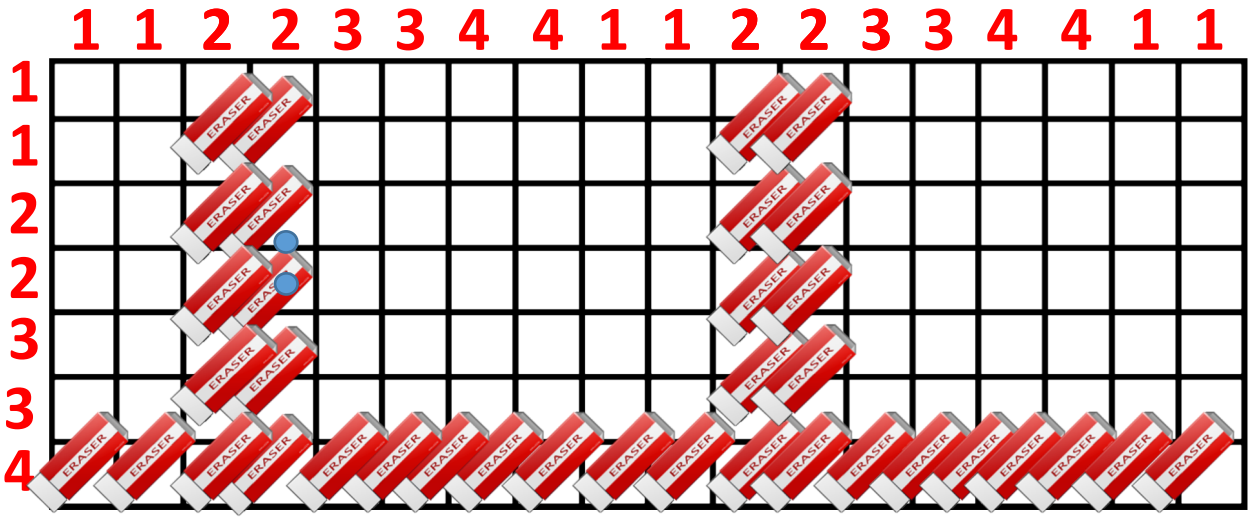
???



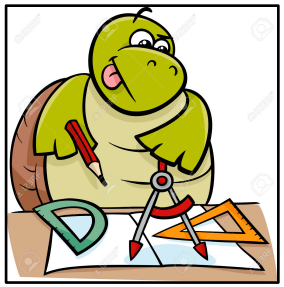
Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



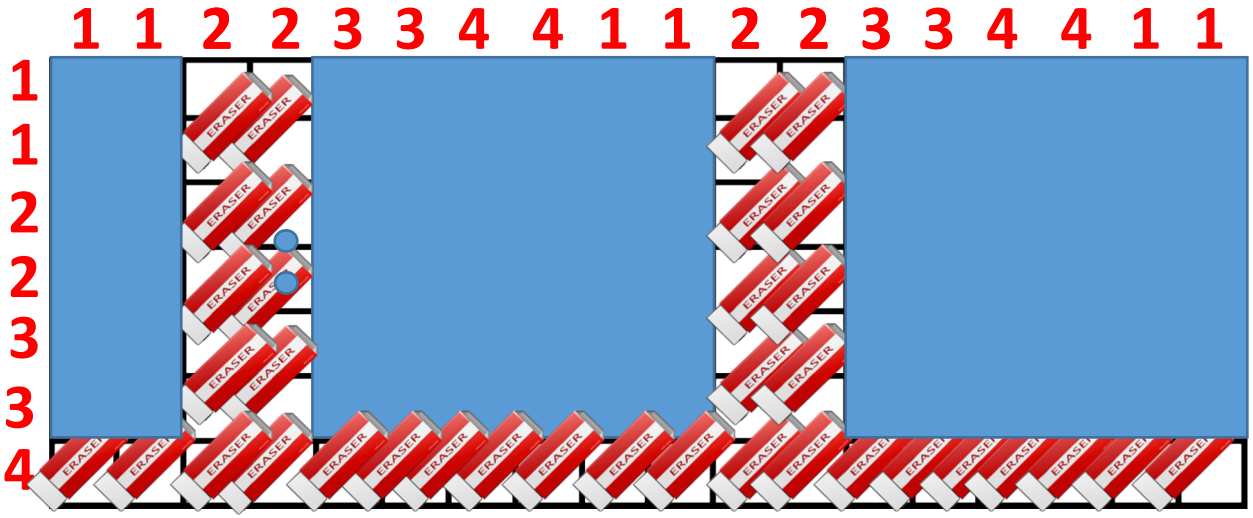
Repeat for rows.



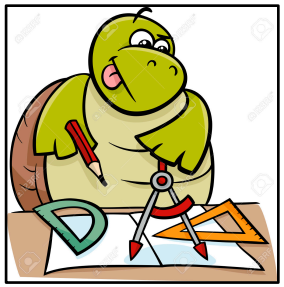
Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



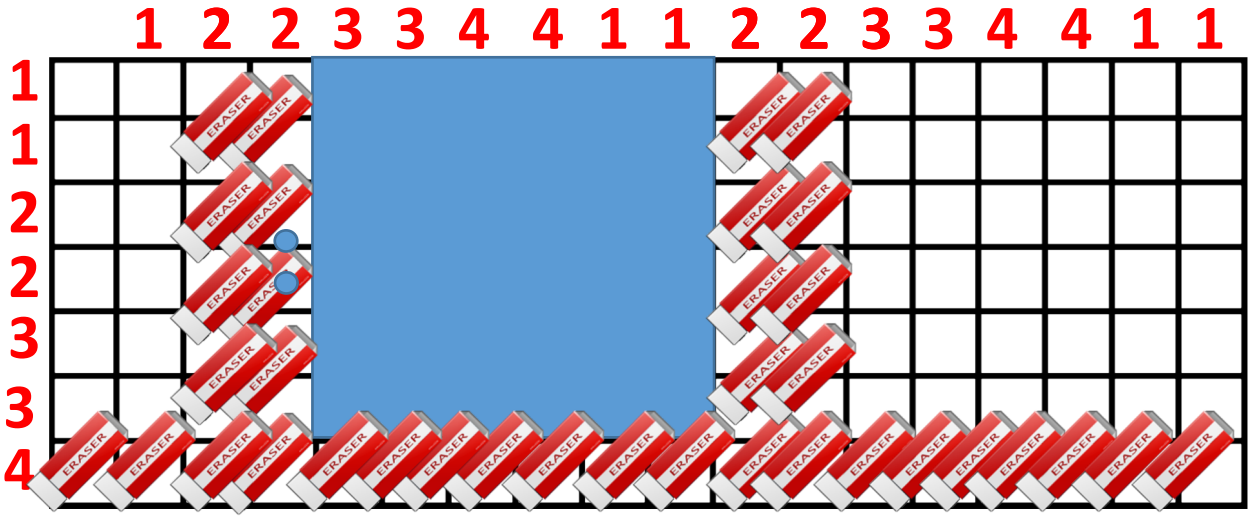
Repeat for rows.



Decomposition II(b): Outline

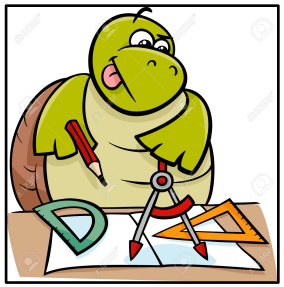
Decomposition II(b): Based on Baker's Technique.

1. Label **pairs of consecutive** columns/rows of the clique-grid graph by 1,…,k+1. ``Guess'' column and row labels whose removal results in a 2$k$x2$k$ ``piece'' that has a solution (if one exists).



Repeat for rows.

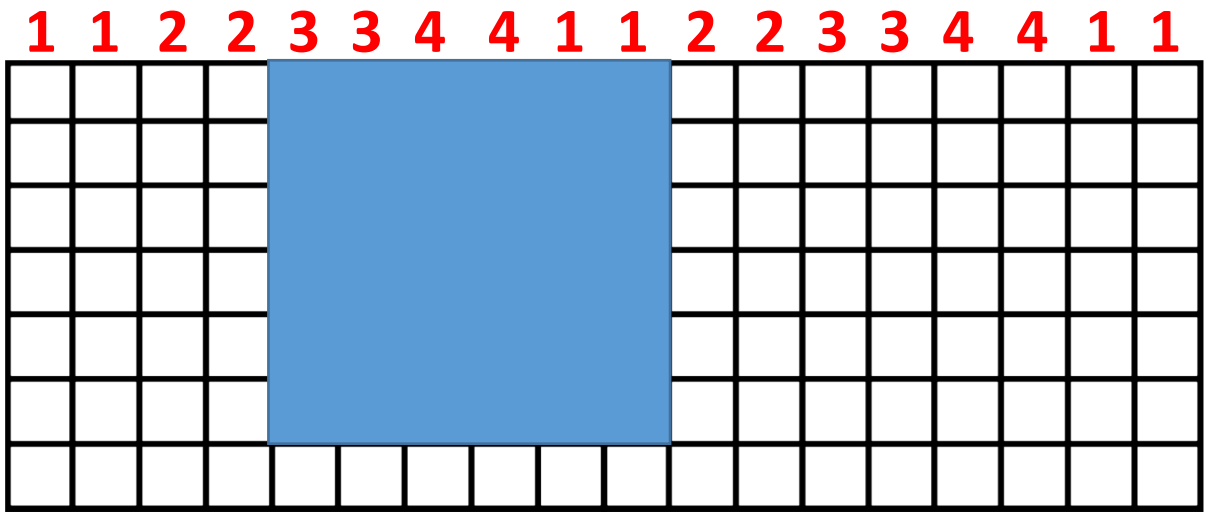
Now, zoom into a ``piece''.

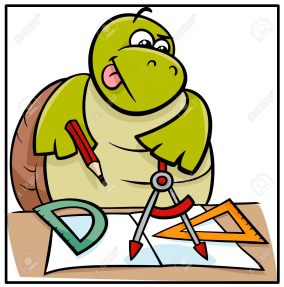


Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

2. Label **pairs of consecutive** columns (rows) of the piece by $1,\ldots,\sqrt{k}$. ``Guess'' a column (row) label such that a solution (if one exists) uses at most $\sqrt{k}$ vertices from the columns (rows) of this label **in total**.

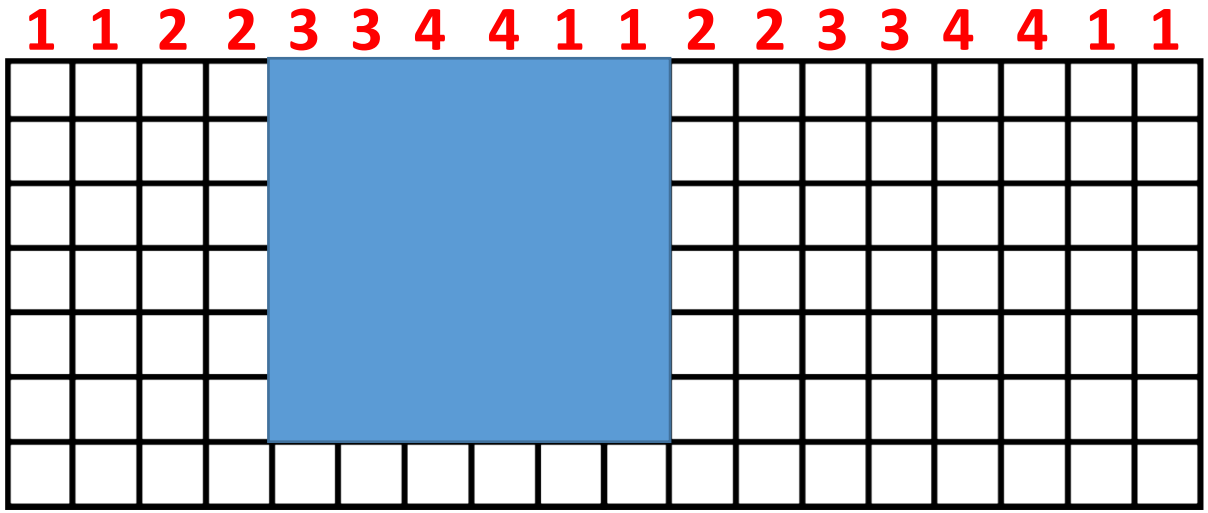




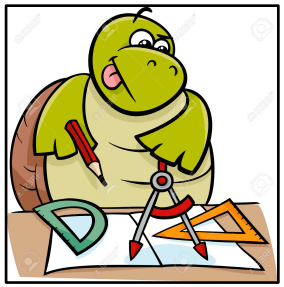
Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

2. Label **pairs of consecutive** columns (rows) of the piece by $1,\ldots,\sqrt{k}$. ``Guess'' a column (row) label such that a solution (if one exists) uses at most $\sqrt{k}$ vertices from the columns (rows) of this label **in total**.



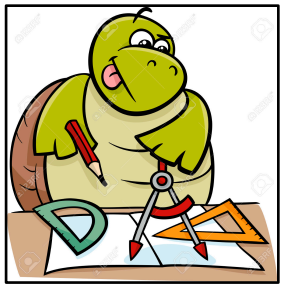
Such a label must exist for Long/Exact Cycle/Path



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

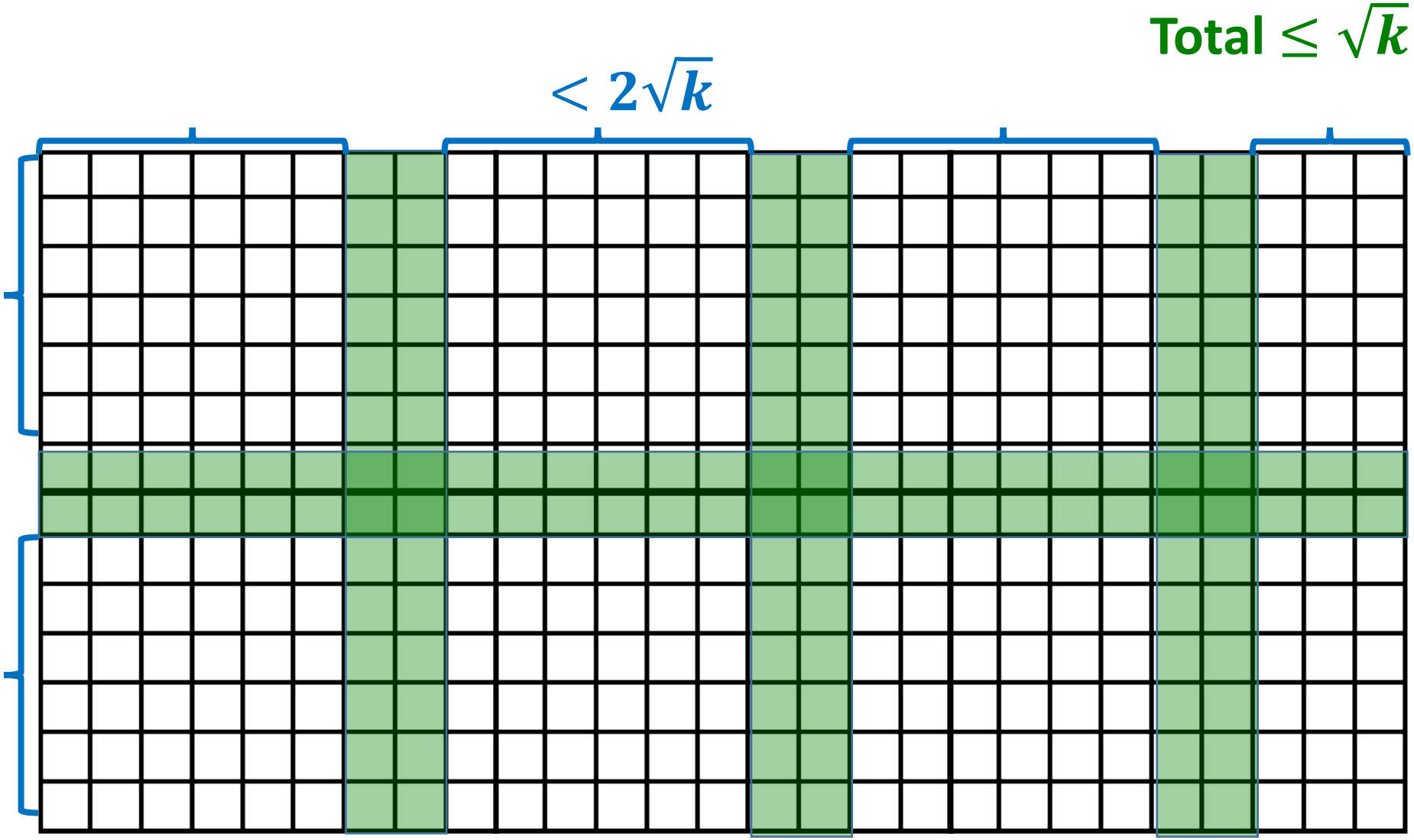
3. ``Guess'' which vertices in the columns and rows of the chosen labels are used, and delete the rest of the vertices in these columns and rows.



Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

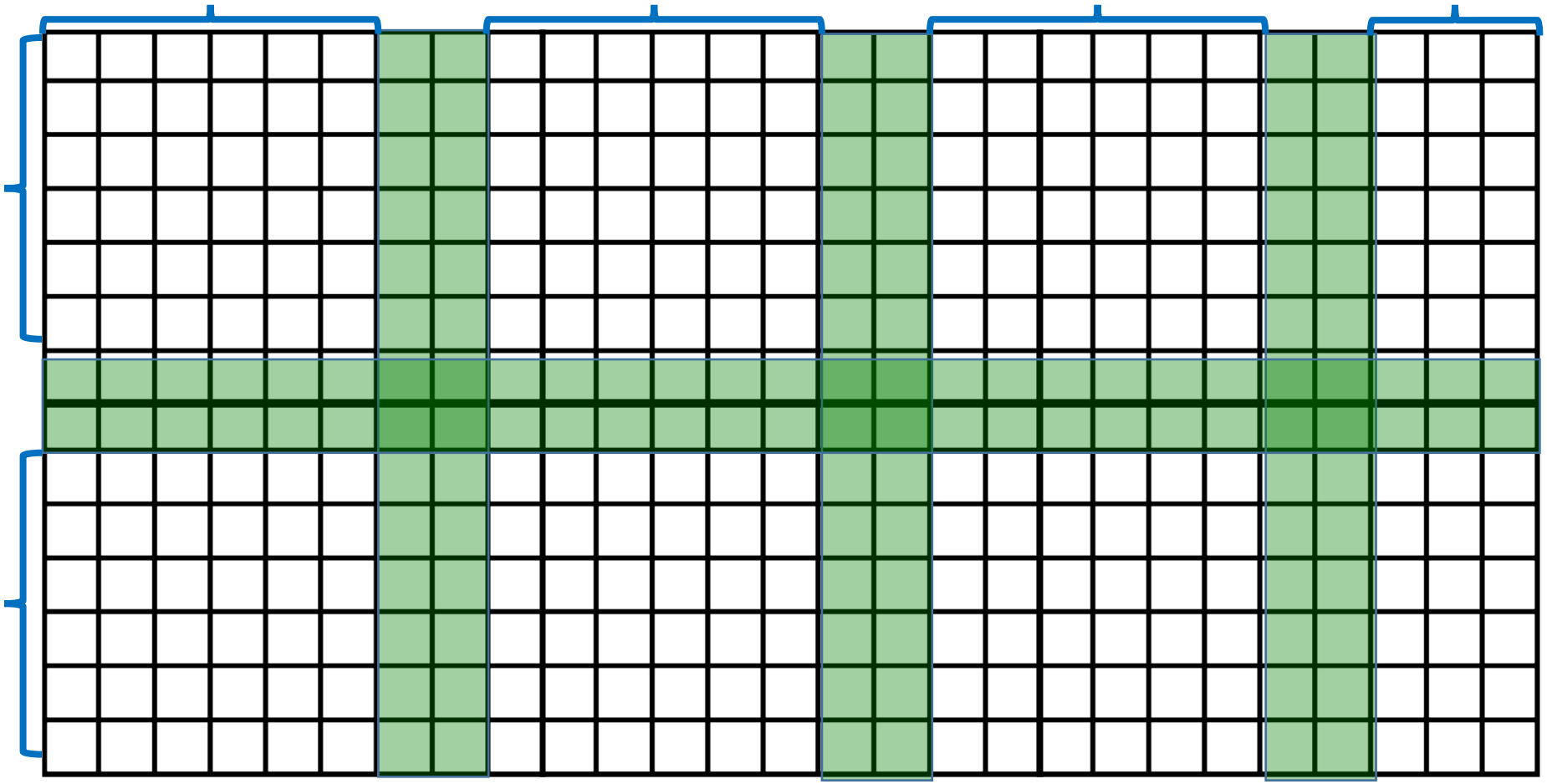
3. ``Guess'' which vertices in the columns and rows of the chosen labels are used, and delete the rest of the vertices in these columns and rows.
4. At this point, we can construct a tree decomposition such that each of its bags consists of $O(\sqrt{k})$ cells.

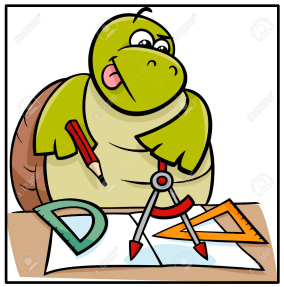

$< 2\sqrt{k}$
Total $\leq \sqrt{k}$

- Insert the $< \sqrt{k}$ green cells that are occupied to all bags.
- In each ``subpiece'', insert **three** columns at a time.

Total $\leq \sqrt{k}$

$< 2\sqrt{k}$



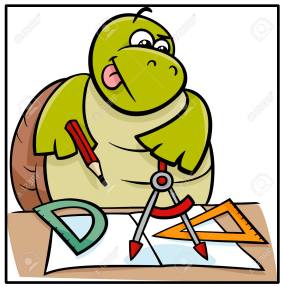


Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

3. ``Guess'' which vertices in the columns and rows of the chosen labels are used, and delete the rest of the vertices in these columns and rows.
4. At this point, we can construct a tree decomposition such that each of its bags consists of $O(\sqrt{k})$ cells.

Now: As before, we bound the size of each cell and make use of the bounded interaction lemmas to design a DP algorithm.



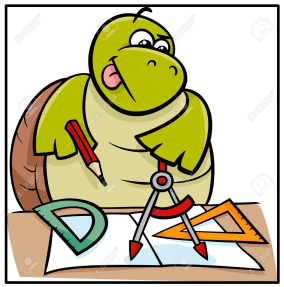
Decomposition II(b): Outline

Decomposition II(b): Based on Baker's Technique.

3. ``Guess'' which vertices in the columns and rows of the chosen labels are used, and delete the rest of the vertices in these columns and rows.
4. At this point, we can construct a tree decomposition such that each of its bags consists of $O(\sqrt{k})$ cells.

Now: As before, we bound the size of each cell and make use of the bounded interaction lemmas to design a DP algorithm.

(Solve Long/Exact Cycle/Path in time $O^*(2^{O(k^{0.5}\log k)})$.)



Decomposition II(b): Outline

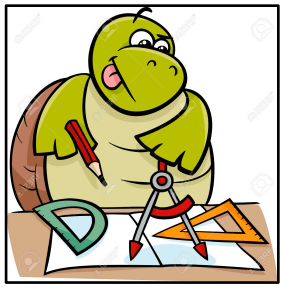
Decomposition II(b): Based on Baker's Technique.

3. ``Guess'' which vertices in the columns and rows of the chosen labels are used, and delete the rest of the vertices in these columns and rows.
4. At this point, we can construct a tree decomposition such that each of its bags consists of $O(\sqrt{k})$ cells.

Note. We constructed $O^*(2^{O(k^{0.5}\log k)})$ decompositions.

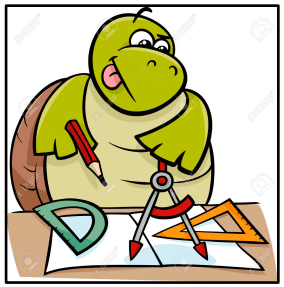
Outline

- Introduction
- Decomposition II: General Decomposition
- **Decomposition III: Contraction Decomposition**



Vertex/Edge Decomposition

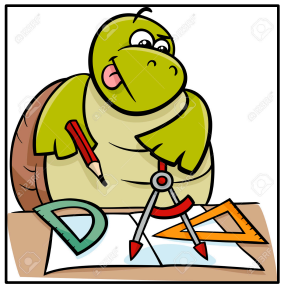
Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $f(k)$.



Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $f(k)$.

+ Compute the decomposition in polynomial time.



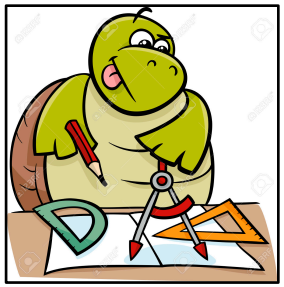
Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $O(k)$.

Planar graphs. [Baker, J. ACM'94]

Bounded genus, apex-minor-free. [Eppstein, Algorithmica'00]

***H*-minor-free.** [Demaine, Hajiaghayi, Mohar, FOCS'05], [DeVos, Ding, Oporowski, Sanders, Reed, Seymour, Vertigan, JCTB'04]



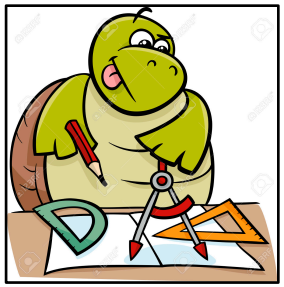
Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $O(k)$.

Planar graphs. [Baker, J. ACM'94]

Bounded genus, apex-minor-free. [Eppstein, Algorithmica'00]

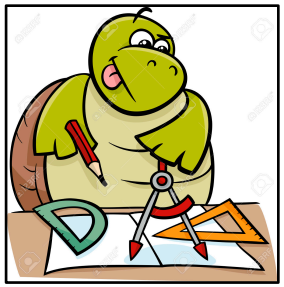
***H*-minor-free.** [Demaine, Hajiaghayi, Mohar, FOCS'05], [DeVos, Ding, Oporowski, Sanders, Reed, Seymour, Vertigan, JCTB'04]



Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $f(k)$.

Note. Does not exist for UDGs: consider a large clique!

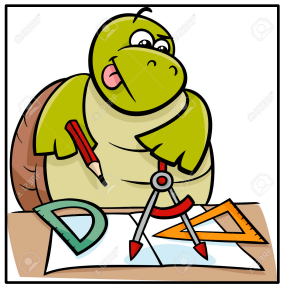


Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $f(k)$.

Note. Does not exist for UDGs: consider a large clique!

However... We sidestepped this issue previously by thinking of cells as units.

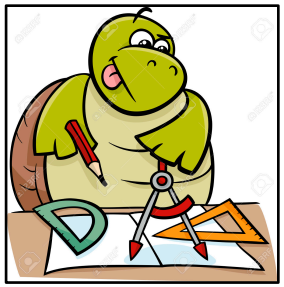


Vertex/Edge Decomposition

Vertex/Edge Decomposition Theorem. Let G be a graph, and k be a positive integer. Then, the vertex/edge set of G can be partitioned into k+1 sets (possibly empty) such that any k of these sets (alternatively, deleting any set) induce a graph of treewidth $f(k)$.

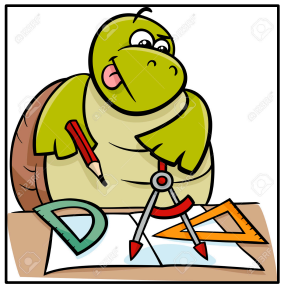
Useful to design approx. and parameterized algorithms for ``local'' problems.

For ``non-local'' problems …



Contraction Decomposition

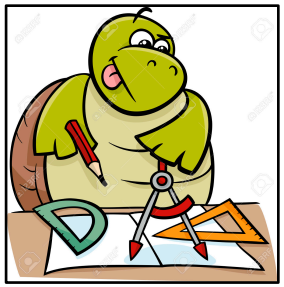
Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $f(k)$.



Contraction Decomposition

Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $f(k)$.

+ Compute the decomposition in polynomial time.



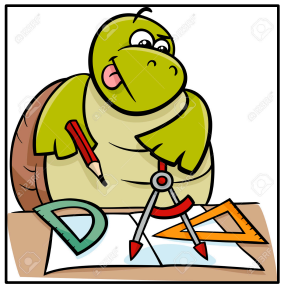
Contraction Decomposition

Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $O(k)$.

Planar graphs. [Klein, STOC'06], [Klein, SICOMP'08]

Bounded genus. [Demaine, Hajiaghayi, Mohar, Combinatorica'00]

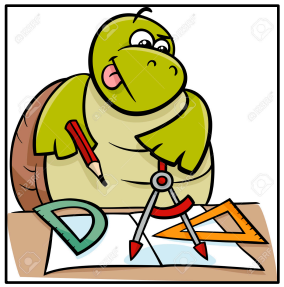
***H*-minor-free.** [Demaine, Hajiaghayi, Kawarabayashi, STOC'11]



Contraction Decomposition

Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $f(k)$.

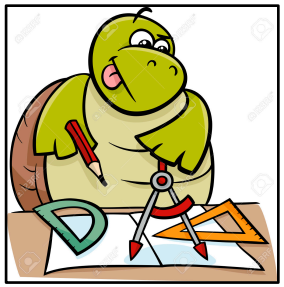
How to use? Suppose that we seek a sol. S of k edges.



Contraction Decomposition

Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $f(k)$.

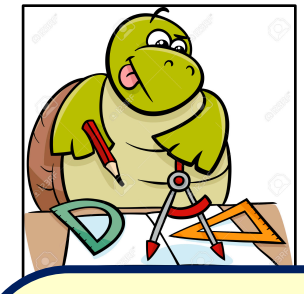
How to use? Suppose that we seek a sol. S of k edges. Then, there is a ``piece'' that does not intersect S. After we contract the piece, S is preserved.



Contraction Decomposition

Contraction Decomposition Theorem (CDT). Let G be a graph, and k be a positive integer. Then, the edge set of G can be partitioned into $k+1$ sets (possibly empty) such that contracting any of these sets induces a graph of treewidth $f(k)$.

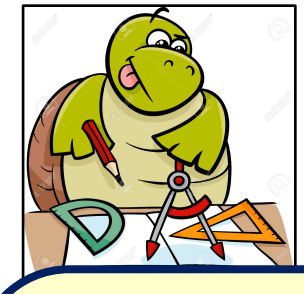
How to use? Suppose that we seek a sol. S of k edges. Then, there is a ``piece'' that does not intersect $S$. After we contract the piece, $S$ is preserved. Now, we solve a ``similar'' problem on a graph of bounded treewidth, and lift the sol. to the original problem.



Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

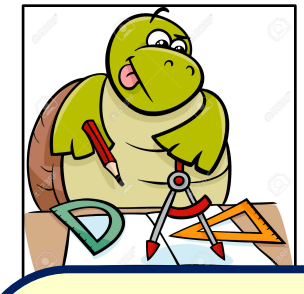
- $\cup_{i=1}^{\alpha k} E_i = E(G)$,



Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

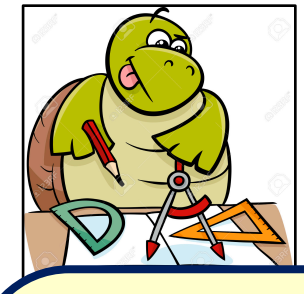
- $\cup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,



Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\bigcup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $f(k)$.



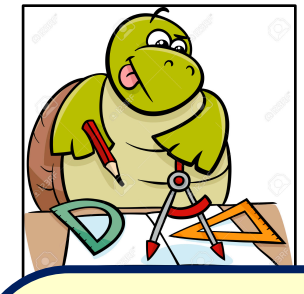
Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\cup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $f(k)$.

$E_1, E_2, \ldots, E_{\alpha k}$: **CDT.** A partition.

Weak CDT. ``Almost'' a partition.



Weak Contraction Decomposition

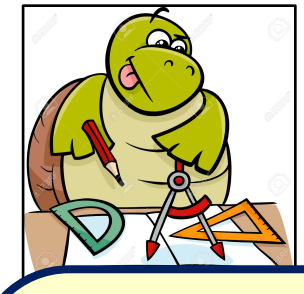
Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\cup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $f(k)$.

$E_1, E_2, \ldots, E_{\alpha k}$: **CDT.** A partition.

Weak CDT. ``Almost'' a partition.

Weak CDT suffices for algorithmic applications.

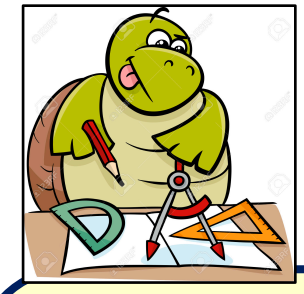


Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\bigcup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $O(k^2)$.

Theorem (CDT in UDGs). Given a **UDG** G and $k \in \mathbb{N}$, such subsets $E_1, E_2, \ldots, E_{\alpha k}$ exist and are computable in polynomial time.



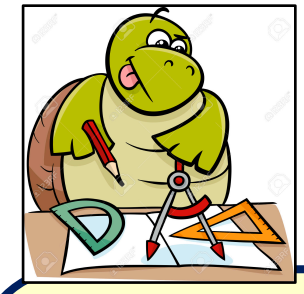
Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\bigcup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $O(k^2)$.

Theorem (CDT in UDGs). Given a **UDG** G and $k \in \mathbb{N}$, such subsets $E_1, E_2, \ldots, E_{\alpha k}$ exist and are computable in polynomial time.

Recall. UDGs do not admit a Vertex/Edge Decomposition.



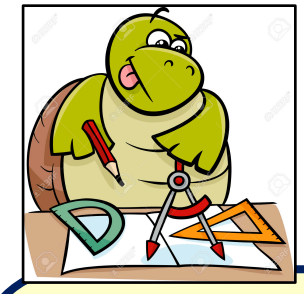
Weak Contraction Decomposition

Weak CDT. Let G be a graph, and $k \in \mathbb{N}$. There exist fixed constants $\alpha, \beta \in \mathbb{N}$ (independent of G and k), and subsets of edges of G, $E_1, E_2, \ldots, E_{\alpha k}$, such that

- $\bigcup_{i=1}^{\alpha k} E_i = E(G)$,
- each edge in $E(G)$ belongs to $\leq \beta$ sets among $E_1, E_2, \ldots, E_{\alpha k}$,
- contracting any E_i induces a graph of treewidth $\boldsymbol{O(k^2)}$.

Theorem (CDT in UDGs). Given a **UDG** G and $k \in \mathbb{N}$, such subsets $E_1, E_2, \ldots, E_{\alpha k}$ exist and are computable in polynomial time.

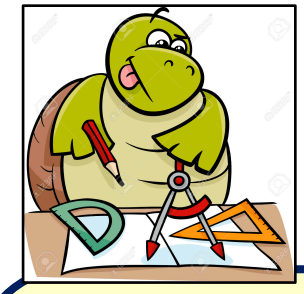
The construction that gives rise to this result is the one we will use in our applications; sidestep the quadratic dependency on k.



Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.

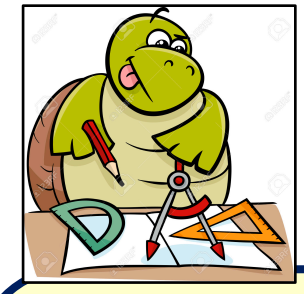


Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

For any bag B and any edge $\{u,v\} \in S$ where $u,v \in B$, u,v belong to either diff. pieces or a single piece **that induces a clique**.

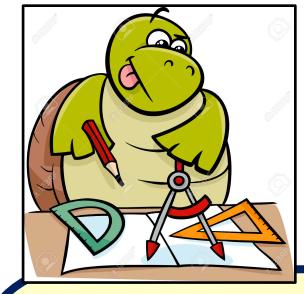


Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

How to use? Suppose that we seek a sol. S of k edges. Then, there is a decomposition where non-clique pieces do not intersect S. After we contract them, S is preserved.



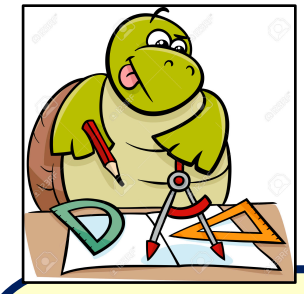
Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

How to use? Suppose that we seek a sol. S of k edges. Then, there is a decomposition where non-clique pieces do not intersect S. After we contract them, S is preserved.

What to do with the clique pieces? (We need to contract all pieces to have linear treewidth.)



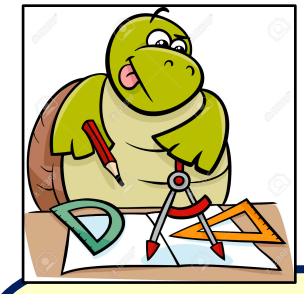
Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

How to use? Suppose that we seek a sol. S of k edges. Then, there is a decomposition where non-clique pieces do not intersect S. After we contract them, S is preserved.

For some cut problems, because cliques are hard to break, we can prove that the total num. of vertices in the cliques that S intersects is small.



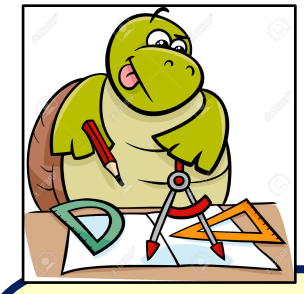
Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

How to use? Suppose that we seek a sol. S of k edges. Then, there is a decomposition where non-clique pieces do not intersect S. After we contract them, S is preserved.

For some cut problems, because cliques are hard to break, we can prove that the total num. of vertices in the cliques that S intersects is small. **(Will see example soon.)**



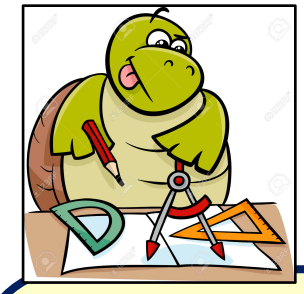
Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

How to use? Suppose that we seek a sol. S of k edges. Then, there is a decomposition where non-clique pieces do not intersect S. After we contract them, S is preserved.

Utilize the previous observation to solve the problem using DP over the tree decomposition.

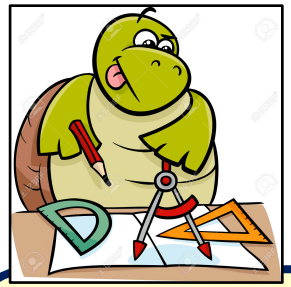


Weak Contraction Decomposition

Theorem (Informal). There is a polynomial-time algo. that, given a UDG G and $k \in \mathbb{N}$, outputs a collection of $O(k)$ tree decompositions of G with the following properties.

- Each bag is partitioned into $O(k)$ connected ``pieces''.
- For any set S of k edges, there is a tree decomposition in the collection where S is well-preserved.

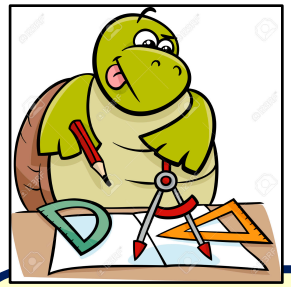
The proof of this theorem is quite technical, so we will not see it here.



Weak Contraction Decomposition

Applications (on UDGs).

- Minimum Bisection is solvable in time $2^{O(k)}n^{O(1)}$.
- s-Way Cut (and Steiner Cut) is solvable in time $k^{O(k)}n^{O(1)}$.
- Edge Multiway Cut Uncut is solvable in time $k^{O(k)}n^{O(1)}$.

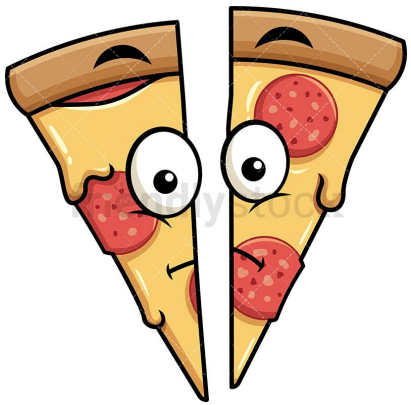


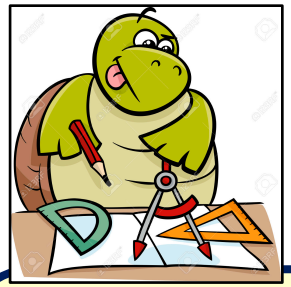
Weak Contraction Decomposition

Applications (on UDGs).

- Minimum Bisection is solvable in time $2^{O(k)}n^{O(1)}$.
- s-Way Cut (and Steiner Cut) is solvable in time $k^{O(k)}n^{O(1)}$.
- Edge Multiway Cut Uncut is solvable in time $k^{O(k)}n^{O(1)}$.

Given a graph G and a non-negative integer k, decide whether there is a partition (A,B) of $V(G)$ such that $|A|=|B|$ and $|E(A,B)| \leq k$.



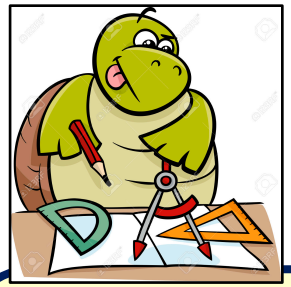


Weak Contraction Decomposition

Applications (on UDGs).

- Minimum Bisection is solvable in time $2^{O(k)}n^{O(1)}$.
- s-Way Cut (and Steiner Cut) is solvable in time $k^{O(k)}n^{O(1)}$.
- Edge Multiway Cut Uncut is solvable in time $k^{O(k)}n^{O(1)}$.

Given a graph G and a non-negative integer k, decide whether there is a subset X of $E(G)$ such that $|X| \leq k$ and G-X has at least s connected components.

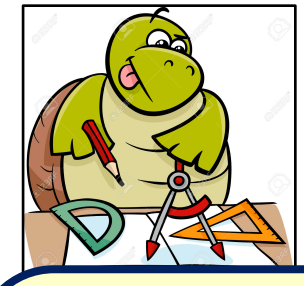


Weak Contraction Decomposition

Applications (on UDGs).

- Minimum Bisection is solvable in time $2^{O(k)}n^{O(1)}$.
- s-Way Cut (and Steiner Cut) is solvable in time $k^{O(k)}n^{O(1)}$.
- Edge Multiway Cut Uncut is solvable in time $k^{O(k)}n^{O(1)}$.

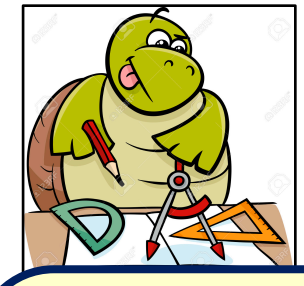
Given a graph G, a terminal set $T \subseteq V(G)$, an equivalence relation R on T, and a non-negative integer k, decide if there is $X \subseteq E(G)$ such that $|X| \leq k$ and in $G-X$ two terminals u,v belong to the same connected component iff $u \sim_R v$.



Example: Bisection

Claim (informal). Consider a bag B of a weak CDT that is ``good'' for a solution (X,Y). Then,

1. Every non-clique piece is contained in either X or Y.
2. The total number of vertices in clique pieces that cross (X,Y) is at most $2k$.

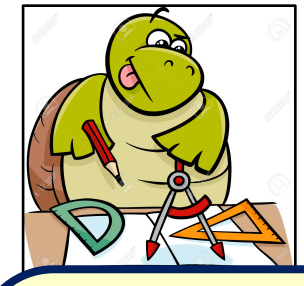


Example: Bisection

Claim (informal). Consider a bag B of a weak CDT that is ``good'' for a solution (X,Y). Then,

1. Every non-clique piece is contained in either X or Y.
2. The total number of vertices in clique pieces that cross (X,Y) is at most $2k$.

The number of edges contributed by a clique piece that crosses (X,Y) is at least the size of its not small side, which is at least half its size.



Example: Bisection

Claim (informal). Consider a bag B of a weak CDT that is ``good'' for a solution (X,Y). Then,

1. Every non-clique piece is contained in either X or Y.
2. The total number of vertices in clique pieces that cross (X,Y) is at most $2k$.

In the DP: (i) Which cliques cross and how they cross ($2^{O(k)}$); (ii) For all non-crossing pieces—what is their side ($2^{O(k)}$); (iii) additional info (number of edges, size of side).

Summary

- Introduction: Bounded Degree/Clique Size; Branch on Cliques
- Decomposition II(a+b): General Decomposition
- Decomposition III(a+b): Contraction Decomposition