Hashing

Wolfgang Mulzer

1 Verkettung

- `put`(k, v): Durchsuche die verkettete Liste bei `elements`$[h(k)]$ nach k. Ersetze den Wert durch v, bzw. lege einen neuen Eintrag an.
- `get`(k): Durchsuche die verkettete Liste bei `elements`$[h(k)]$ nach k. Gib den Wert für k zurück, bzw. wirf eine `NoSuchElementException`.
- `remove`(k): Durchsuche die verkettete Liste bei `elements`$[h(k)]$ nach k. Lösche den Eintrag für k, bzw. wirf eine `NoSuchElementException`.

2 Offene Adressierung (lineares Sondieren)

- `put`(k, v):

```
pos <- h(k)
delPos <- NULL
for i := 1 to N do
  if (elements[pos].k == k)
    elements[pos] <- (k,v)
    return
  if (elements[pos] == DELETED && delPos == NULL)
    delPos <- pos
  if (elements[pos] == NULL)
    break
  pos <- (pos + 1) mod N
if (delPos != NULL)
  elements[delPos] <- (k,v)
else if (elements[pos] == NULL) then
  elements[pos] <- (k,v)
else
  throw TableFullException
```

- `get`(k):

```
pos <- h(k)
for i := 1 to N do
  if (elements[pos] == NULL)
    throw NoSuchElementException
  if (elements[pos].k == k)
    return elements[pos].v
```

```
    pos <- (pos + 1) mod N
  throw NoSuchElementException
```

- remove(k):

```
  pos <- h(k)
  for i := 1 to N do
    if (elements[pos] == NULL)
      throw NoSuchElementException
    if (elements[pos].k == k)
      elements[pos] <- DELETED
      return
    pos <- (pos + 1) mod N
  throw NoSuchElementException
```

3 Kuckuck

- get(k):

```
  if (elements[h1(k)].k == k)
    return elements[h1(k)].v
  if (elements[h2(k)].k == k)
    return elements[h2(k)].v
  throw NoSuchElementException
```

- remove(k):

```
  if (elements[h1(k)].k == k)
    elements[h1(k)] <- NULL
    return
  if (elements[h2(k)].k == k)
    elements[h2(k)] <- NULL
    return
  throw NoSuchElementException
```

- put(k, v):

```
  if (elements[h1(k)].k == k)
    elements[h1(k)] <- (k,v)
    return
  if (elements[h2(k)].k == k)
    elements[h2(k)] <- (k,v)
    return
  if (|S| == N)
    throw TableFullException
  pos <- h1(k)
  for i := 1 to N do
    if (elements[pos] == NULL)
      elements[pos] <- (k,v)
      return
    (k,v) <-> elements[pos]
```

```
  if (pos == h1(k))
    pos <- h2(k)
  else
    pos <- h1(k)
Waehle neue Hashfunktionen und baue die Tabelle neu auf.
put(k,v)
```