6.897
L1.4
Feb. 10, 2003

⑤ Don't store min recursively → Fix Insert

- Successor checks for min specially:
  if $x < min[S]$: return $min[S]$

```
Insert(x, S)
  if x < min[S]: swap x & min[S].
  if min[sub[S][high(x)]] ≠ nil (nonempty):
      Insert(low(x), sub[S][high(x)])
  else: min[sub[S][high(x)]] ← low(x)
      Insert(high(x), summary[S])
  if x > max[S]: max[S] ← x.
```

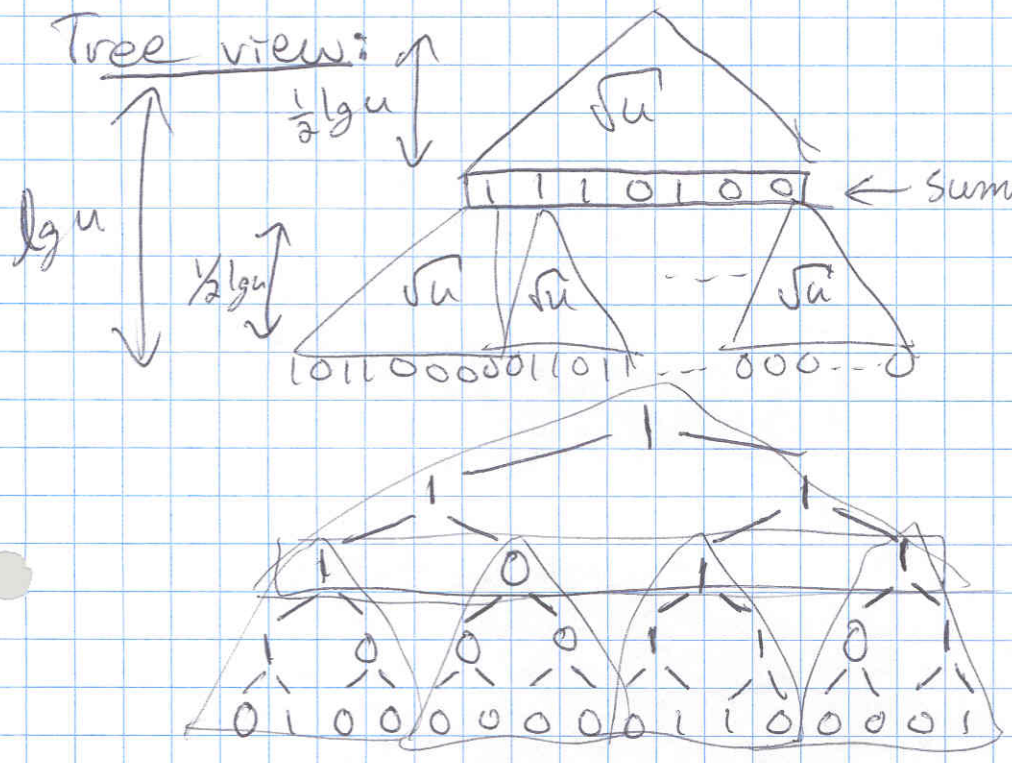
$O(\lg \lg u)$.

6.897
L2.1
Feb. 12, 2003

⑥ Delete(x, S)

```
if min[S] = nil or x < min[S]: return
if min[S] = x: i ← min[summary[S]]
               x ← i·√|S| + min[sub[S][i]]
               min[S] ← x
Delete(low(x), sub[w][high(x)])
if min[sub[S][high(x)]] = nil (now empty):
    Delete(high(x), summary[S])
    <<in this case, first rec. call was cheap>>
```

$O(\lg(\lg u))$



OR gate

"binary search" on levels.