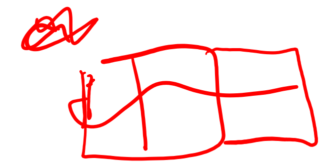
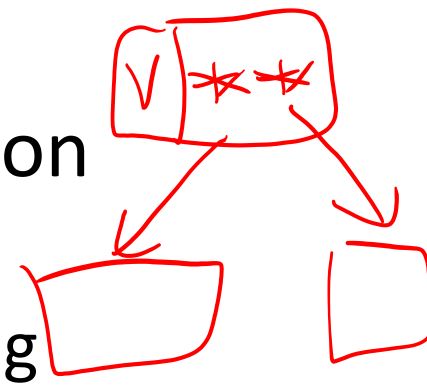
Announcements

Announcements



- HW3 has just been updated- problem 3b removed, Extra Credit updated accordingly
- If you have not yet signed up for an oral exam slot, do that today! (Check BB for link)



- Exam questions have been posted, due on Sunday
  - Please look soon so you can clarify anything needed!

h(1) h(3)

- Note on use of "advanced" data structures

[1, 3, 1, 5, 6, 1, 3]

[1 3 5 ...]

sets, hashmaps

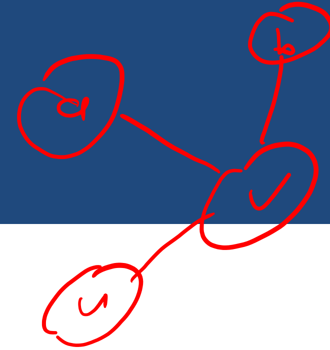
{1, 3, ...}

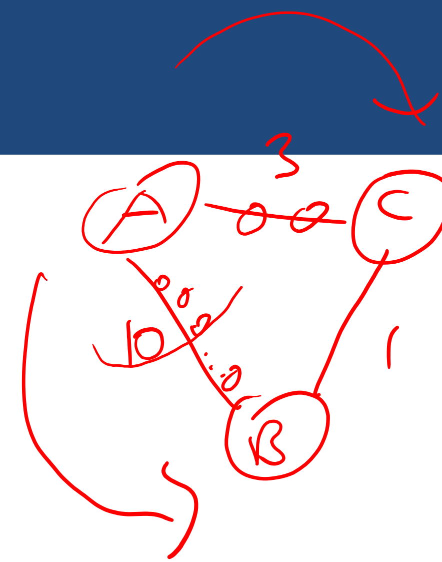
Announcements

- Exam procedure:
  - Join your examiner's "room" on BB Collaborate
  - Join at the start of your timeblock 3/block
  - The examiner will create a "breakout room" for the person being examined
  - Process: 15 minutes/problem
    - 4 Present your solution (3-4 minutes). You can refer to your submitted solution.
    - 4 Answer questions (~ 5 minutes)
    - 7 Propose a solution for a modified version of the problem (6-7 minutes)
  - You can leave after your exam is over
  - Have camera on!

Graphs

Dijkstra's Algorithm





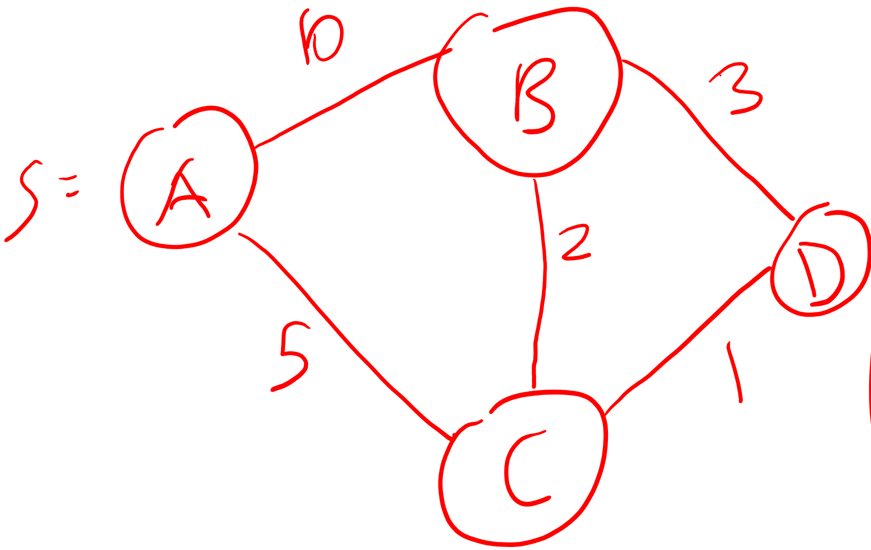

```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0

H = makequeue(V)  (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

if v is still in H:

Dijkstra's Algorithm

A → C → B



	A	B	C	D
prev:	∅	∅	∅	∅
		C	A	C
dist:	~~∞~~	~~∞~~	~~∞~~	~~∞~~
	0	~~10~~ 7	5	6

H: ~~A: 0~~

~~B: ∞ 10~~ 7

~~C: ∞ 5~~

~~D: ∞ 6~~

In-Class Exercise

N = # nodes
M = # edges

- Suppose we implement the priority queue using an array (A[i] = key value of node *i*)
- What is the running time?

[~~0~~, ∞, ∞, ..., ∞]
-1

$O(N) + O(M) + O(N^2) = O(N^2)$

$(M \le N^2)$

for all $u \in V$:
    dist$(u) = \infty$
    prev$(u) =$ nil
dist$(s) = 0$

$O(N)$

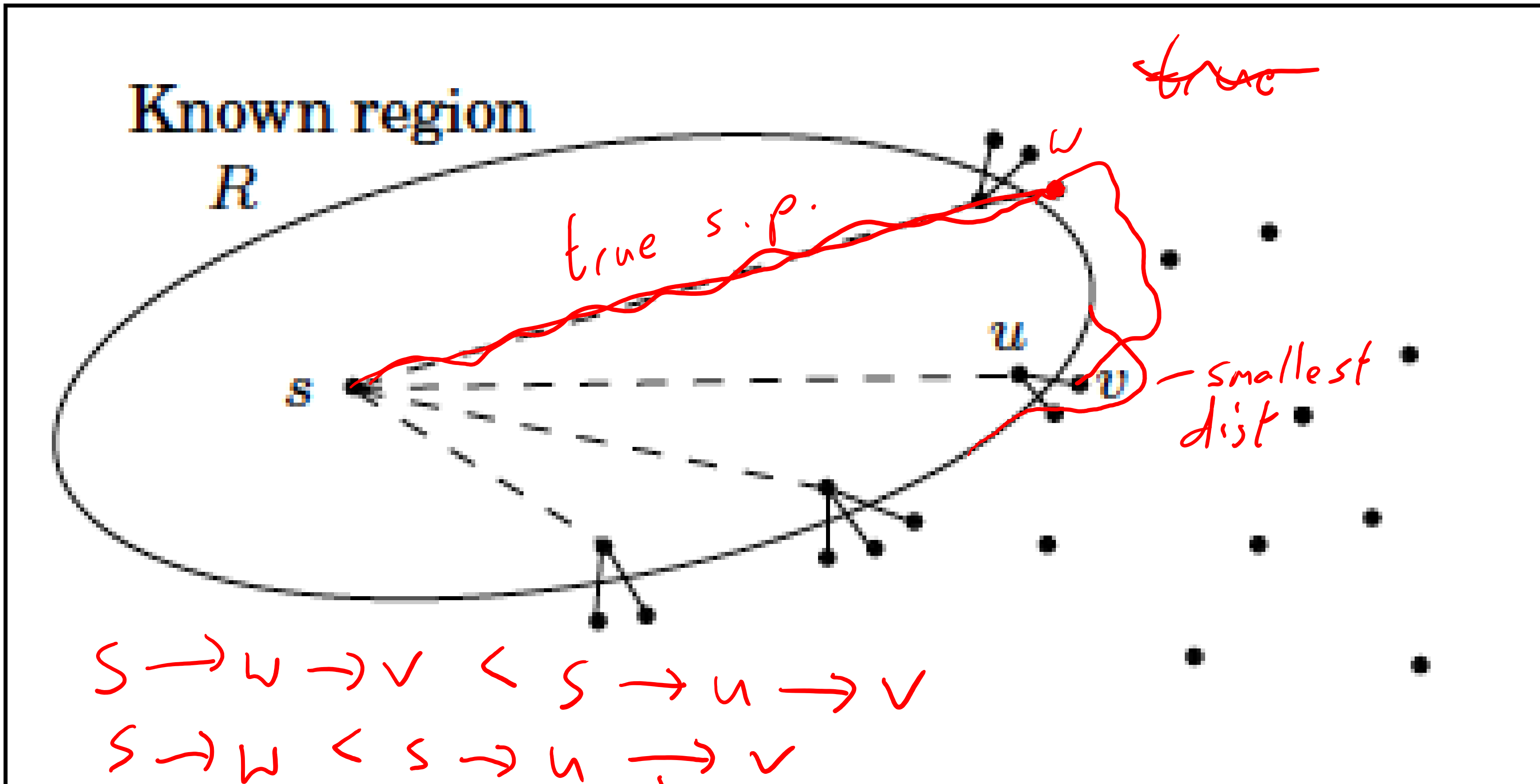
$O(N)$

$H =$ makequeue(V) (using dist-values as keys)
while H is not empty:
    $u =$ deletemin(H) $O(N) \cdot N = O(N^2)$
    for all edges $(u, v) \in E$:
        if dist$(v) >$ dist$(u) + l(u, v)$: $O(M)$
            dist$(v) =$ dist$(u) + l(u, v)$ $O(M)$
            prev$(v) = u$ $O(M)$
            decreasekey(H, v) $O(1) \cdot O(M)$

$O(M)$

Another Way to Think About This

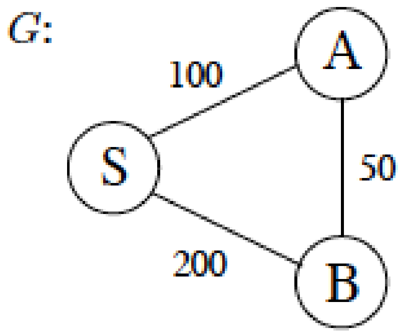
$s \to w < s \to u \to v$

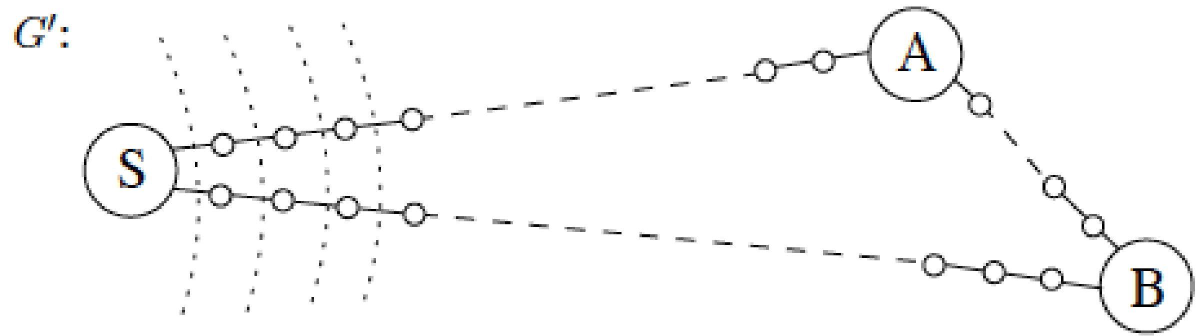


$s \to w \to v < s \to u \to v$

$\underbrace{s \to w}_{dist(w)} < \underbrace{s \to u \to v}_{dist(v)}$

Dijkstra's Algorithm





Dijkstra's Algorithm: Running Time

- Dijkstra's algorithm uses a priority queue:
  - Insert
  - Decrease-key
  - Delete-min
- Suppose we implement as an array, where A[*i*] holds key of node *i*
- What is the running time for delete-min?

  $O(N)$

- For decrease-key? $O(1)$

Dijkstra's Algorithm: Running Time

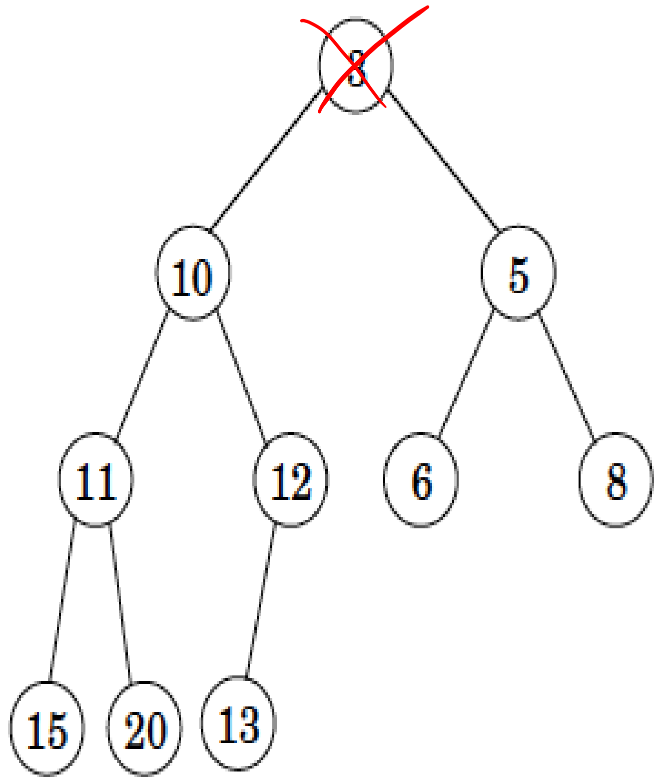
```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0

H = makequeue(V)   (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

Binary heaps

min-heap

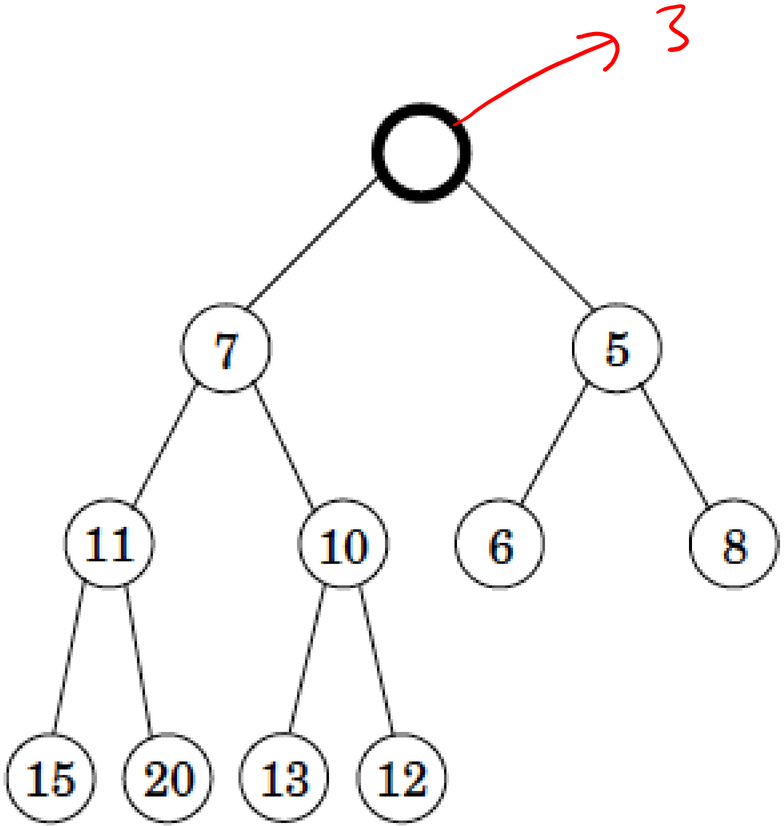
- A binary heap is a complete binary tree
  - Every level must be full before a new level is started
  - Fill from L → R
- The key-value of any node is less than or equal to those of its children
- Which element has the smallest value? root



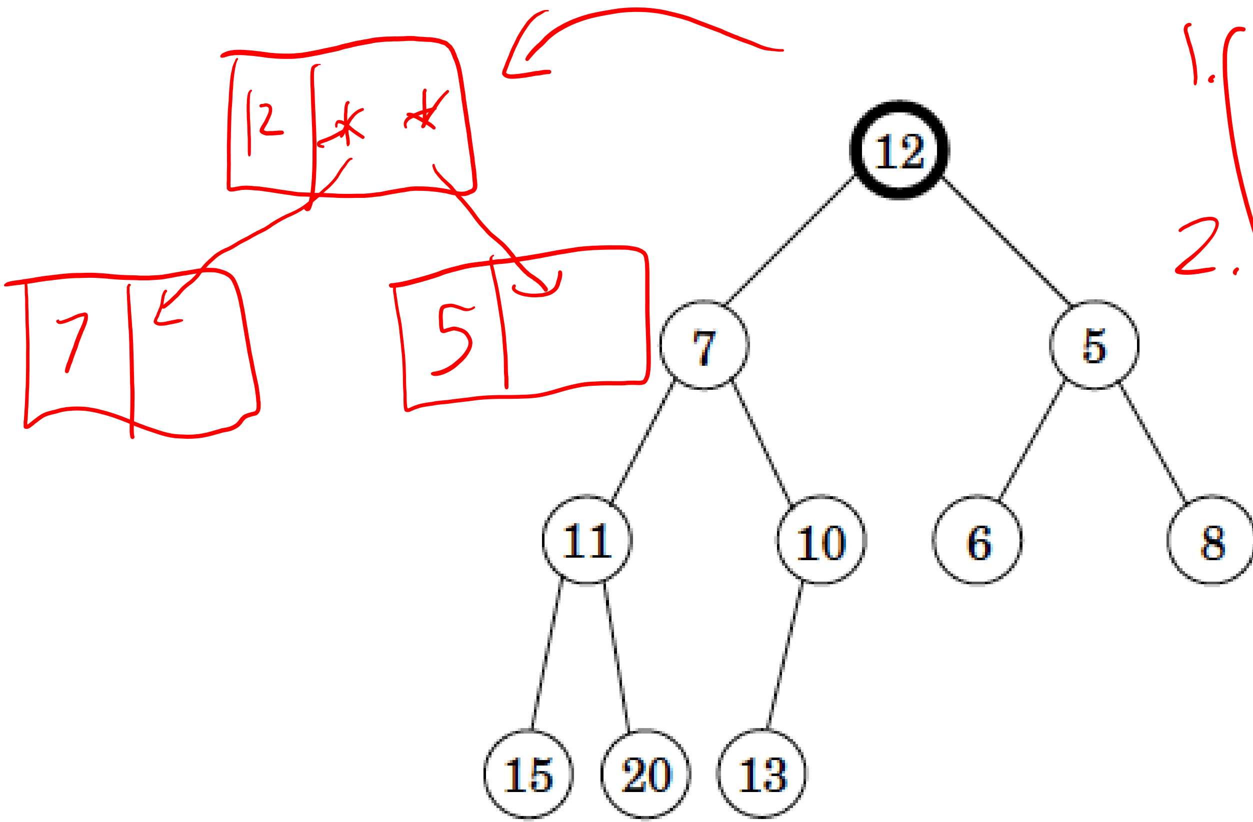
Binary heaps

- How do we delete-min?
  - Return the root value
  - Delete the root
  - Take the “last” element in the tree (lowest level, furthest right), and put it in the root element
  - Swap that element with whichever of its children are smallest, and let it “sift down”

Binary heaps: Delete-min



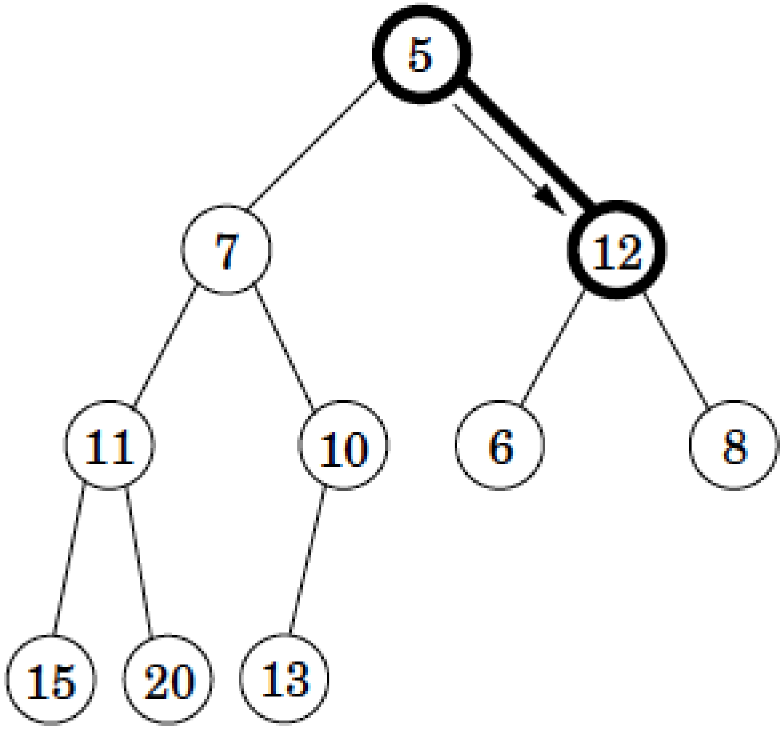
Binary heaps: Delete-min



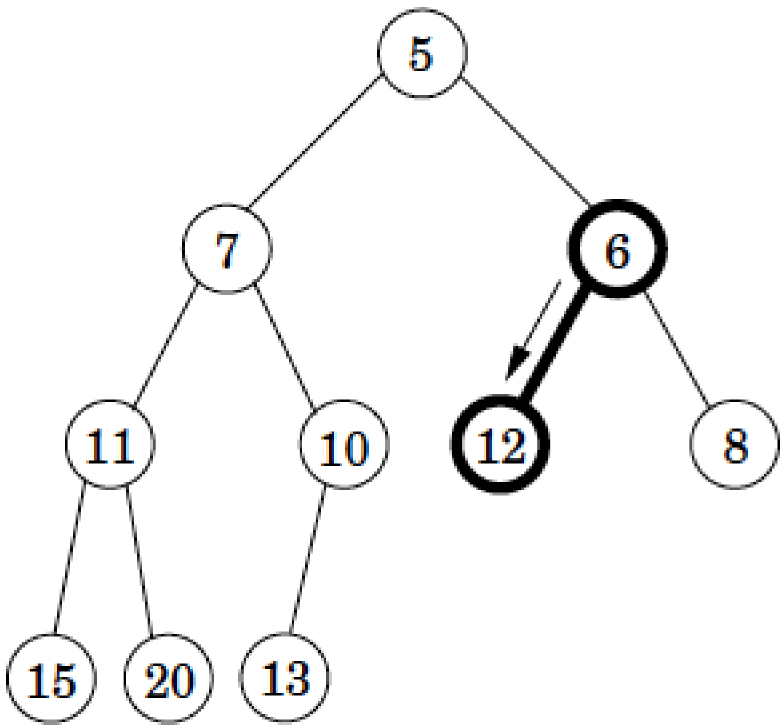
~~1. Who are 5's children, who is 12's other~~
~~2. Have 12 point to those~~

1. Who are 5's children?
2. Who is 12's other child?
3. Have 5 * → 12, 7
4. Have 12 → 5's children
5. Update 7, 6, 8 parent *

Binary heaps: Delete-min



Binary heaps: Delete-min



In-Class Exercise

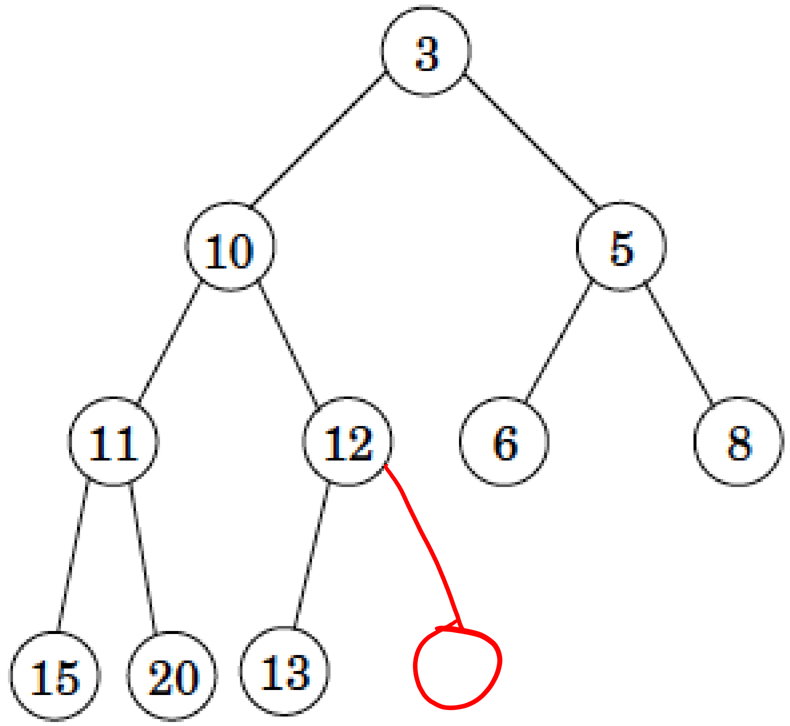
- Suppose the binary heap has *n* elements
- Analyze the running time of a single delete-min operation $O(\log n)$

∵ depth of tree is $O(\log n)$

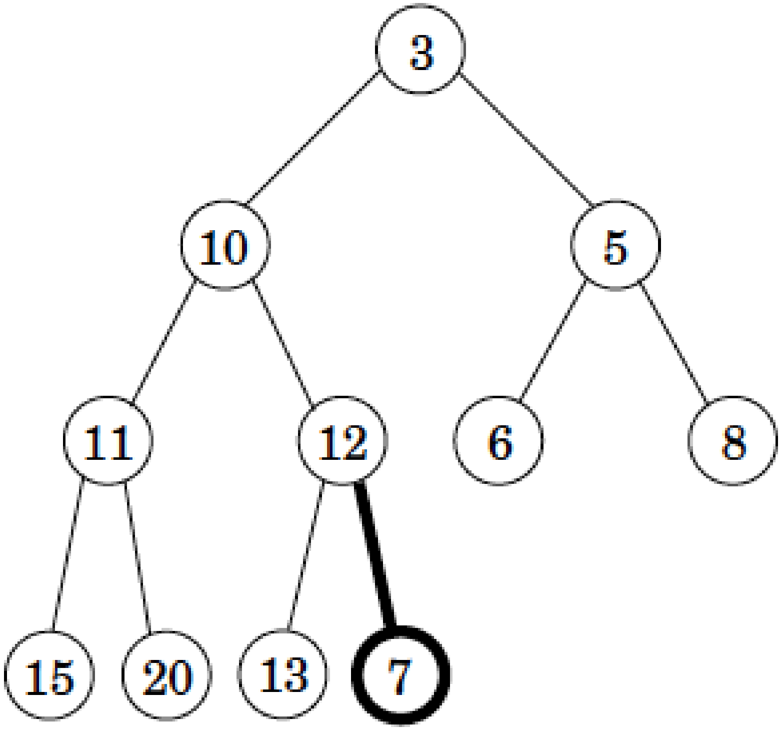
Binary heaps

- How do we insert?
  - Place the new element in the first empty spot (bottom level, furthest right)
  - Let it "bubble up" by swapping it with its parent, for as long as the parent is larger

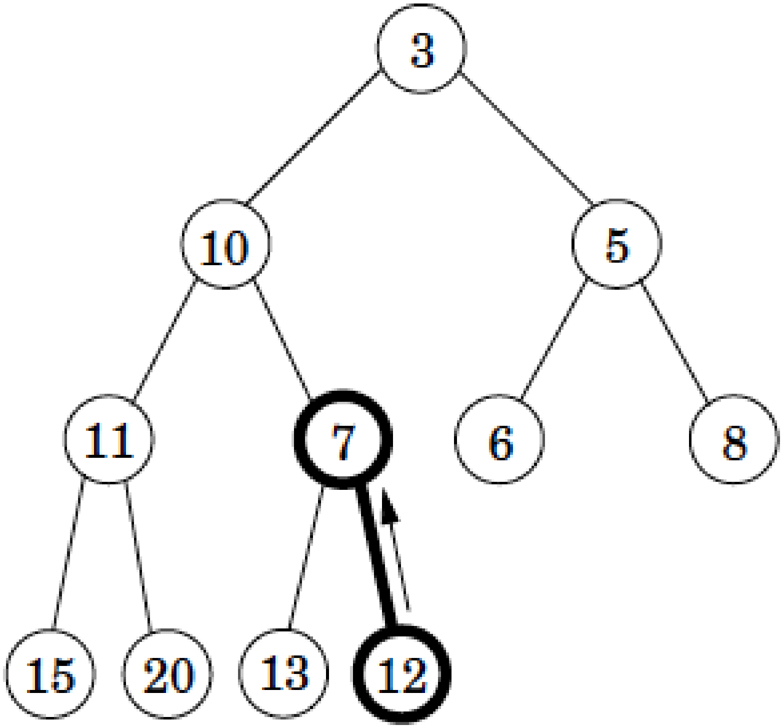
Binary heaps: Insert



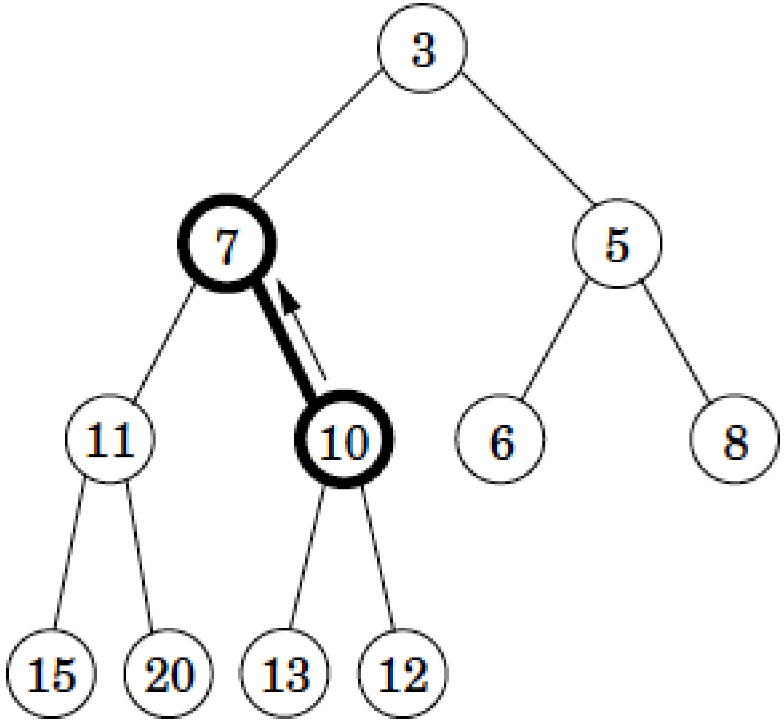
Binary heaps: Insert



Binary heaps: Insert



Binary heaps: Insert



Binary heaps: Insert

- What is the running time of a single insert operation? $O(\log n)$

- Similar implementation for decrease-key: do you see why?

Running time of Dijkstra's Algorithm

$O((N+M)\log N)$

```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0
```

$O(N)$

$O(\log N)\cdot N$

```
H = makequeue(V)   (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```

N times $\cdot\, O(\log N) = O(N\log N)$

$O(M)$

$O(M)$

$O(M)$

$O(M)$

$O(M)\cdot O(\log N) = O(M\log N)$

Running time of Dijkstra's Algorithm

Running time using binary heaps:
(|V| + |E|) log |V|
n m n

```
for all u ∈ V:
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0

H = makequeue(V)  (using dist-values as keys)
while H is not empty:
    u = deletemin(H)
    for all edges (u,v) ∈ E:
        if dist(v) > dist(u) + l(u,v):
            dist(v) = dist(u) + l(u,v)
            prev(v) = u
            decreasekey(H,v)
```