CS 240E S25

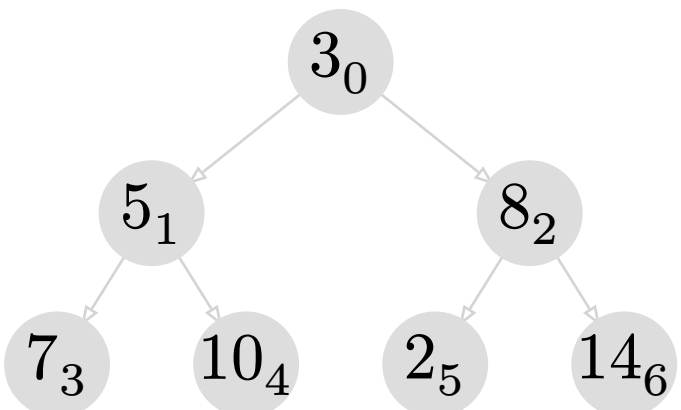
Tutorial 2

May 16

- Heapify
- Amortized analysis
- Potential method
- Merging Binary Heaps
- Increase-key
- Application of heaps

Q1: Heapify

Execute `heapify` on the following array (displayed here in tree form with array indices as subscripts):



Q2: Amortized analysis - Successor in BST

We have a binary search tree storing n distinct keys. The operation `successor(x)` returns the in-order successor of x, which is the node z with x.key $<$ z.key and no other keys are stored in between (`null` if such z does not exist), in $\Theta(\text{height})$ time. Consider the algorithm to print all keys in the tree T in increasing order:

```
x = T.get_min()
print(x.key)
while(T.successor(x) is not null):
    x = T.successor(x)
    print(x.key)
```

(a) Give an asymptotic bound on the worst-case runtime of this algorithm, if the height of T is in $\Theta(\log n)$.

Q2: Amortized analysis - Successor in BST

```
x = T.get_min()
print(x.key)
while(T.successor(x) is not null):
    x = T.successor(x)
    print(x.key)
```

(b) Show that the amortized runtime of successor is $O(1)$ (and therefore the runtime of the algorithm is $\Theta(n)$).

Q3: Potential Method for Amortized Analysis

You are implementing a n-bit binary counter. It is an array of n bits, with the leftmost bit being the least significant bit. The counter starts at 0 and increments by 1 each time you call the operation `increment()`:

```
increment(A[0..n-1]):
    i = 0
    while(A[i] != 0):
       A[i] = 0
       ++i
    A[i] = 1
```

The run-time for `increment` is $\Theta(k)$, where k is the final value of variable i, which is $\Theta(n)$ in the worst case.

Show the amortized cost per increment is $\Theta(1)$ by choosing a proper potential function.

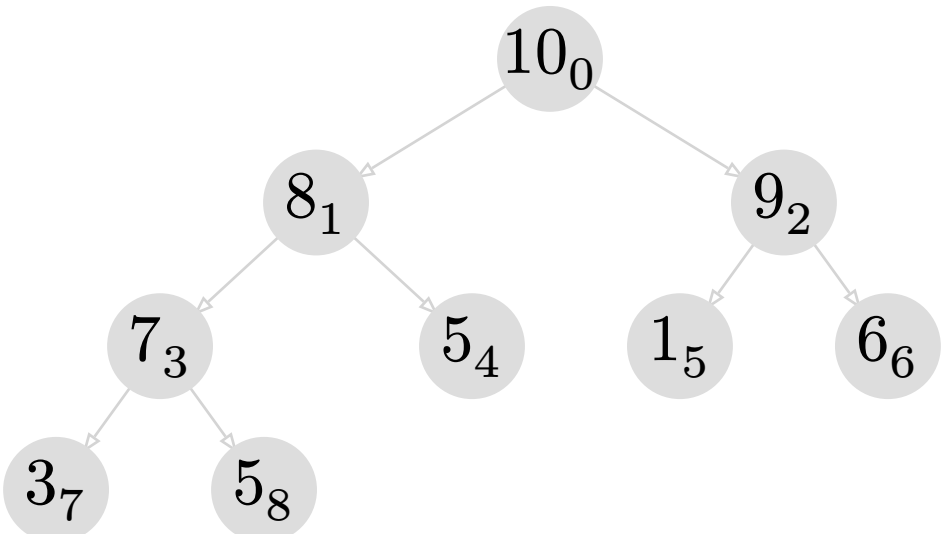
Q4: Merging Binary Heaps

Let H_1 and H_2 be two binary heaps that both store exactly $2^h - 1$ items for some $h \in \mathbb{N}$.

Show how to merge these two heaps in $O(h)$ run-time.

- presuming both heaps are stored as trees (not arrays)
- the output must again be a binary heap (satisfy the structural property)

Q5: increase-key



Here is a binary heap. Show the intermediate steps when performing `increase-key(id: 8, new-key: 12)` using `fix-up`.

* `increase-key` increases the specified node's key to a larger new key, while keeping the heap order property after the operation.

Additional problems

Q6: Minimum function values

You have n functions $f_1, f_2, ..., f_n$ defined as:

$$f_i(x) = a_i x + b_i$$

where a_i and b_i are constants. All $a_i > 0$.

Design an algorithm to find the m minimum values of $f_i(x)$ for all $x \in \mathbb{N}$ (including $x = 0$) efficiently.

Input:

n, All a_i and b_i for $i = 1, ..., n$, and m.

Q7: Median of first i numbers

You have an array of integers $a[i]$ of length n. For all even $0 \leq i \leq n$, output the median of $a[0], a[1], \ldots, a[i]$.