

Problem Set 5 — Heaps, Non-comparison sorts, Red-black trees, Hashing

Suggested practice problems, from CLRS: Ch 11.1 (1 and 2); 11.2-3; 12.2 (3, 4, and 5); 12.3-5; 13.3 (1, 2, and 4)

1. In this problem, we will investigate d -ary max-heaps: A d -ary heap is one in which each node has at most d children, whereas, in a binary heap, each node has at most 2 children.

- (a) We can represent a d -heap in an array which the second element is the root. Then for any parent node x its children are located $(x * d) + 1, (x * d) + 2, \dots, (x * d) + d$. And for any child can find its parent by $(x - 1)/d$.
- (b) If the d -ary heap is completely filled then the i th level will be d^i . So the to find the nodes up to the last level of the heap at level l would be:

$$\sum_{i=0}^l d^i = \frac{d^{l+1} - 1}{d - 1} \quad (1)$$

This function describes the amount of nodes, or n , so we need to solve for l which would be the height.

$$\begin{aligned} n &= \frac{d^{l+1} - 1}{d - 1} \\ n(d - 1) &= d^{l+1} - 1 \\ \log_d(n(d - 1) + 1) - 1 &= l \end{aligned} \quad (2)$$

This height is $\Theta(\log_d(n(d - 1) + 1) - 1)$.

- (c) Re-write function PARENT(i) for d -ary heaps, and give a new function CHILD(i, j) that gives the j -th child of node i (where $1 \leq j \leq d$).

```

1: function PARENT( $i$ )
2:   return  $(i - 1)/d$ 
3: function CHILD( $x, j$ )
4:   return  $(i * d) + j$ 

```

- (d) Describe, and give pseudocode for, the algorithm $\text{MAX-HEAPIFY}(A, i)$ for d -ary heaps and give a tight analysis for the worst-case running time of your algorithm.

```

1: function MAX-HEAPIFY( $A, i$ )
2:    $largest \leftarrow i$ 
3:   for  $x \leftarrow 1$  to  $d$  do
4:     if  $\text{Child}(A, x) > i$  then
5:        $largest \leftarrow \text{Child}(A, x)$ 
6:   if  $largest \neq i$  then
7:     exchange  $A[i]$  with  $A[largest]$ 
8:     Max-heapify( $A, largest$ )

```

The worst-case running time would be $\Theta(\log_d(n(d-1)+1) - 1)$ if the value floats to the bottom of the tree. If the tree is balanced.

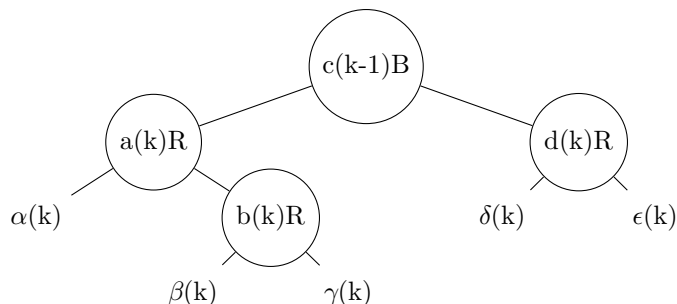
- (e) Describe (semi-formally) how to implement $\text{MAX-HEAPIFY}(A, i)$ in $O((\log_d n) \lg d)$ time. (*Hint: you need auxiliary data structures; the heap itself is not sufficient.*)
2. (From homework 4, skip if already submitted) Problem 8.2-4 from CLRS: Describe (semi-formally) an algorithm that, given n integers in the range 0 to k , preprocesses its input and then answers any query about how many of the n integers fall into a range $[a..b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

```

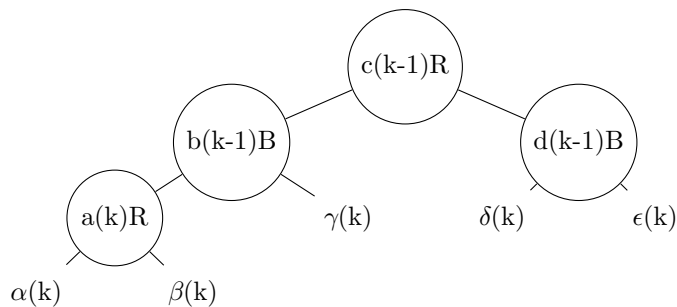
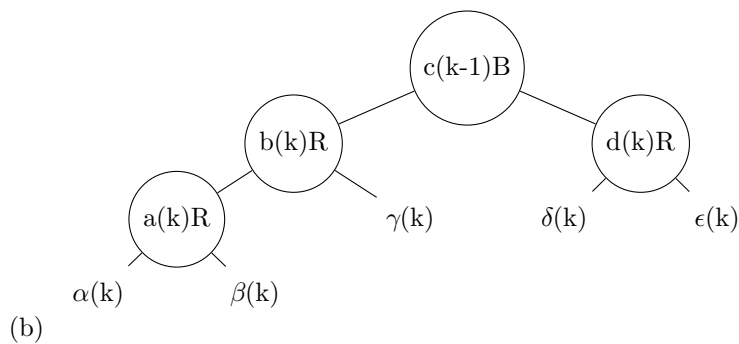
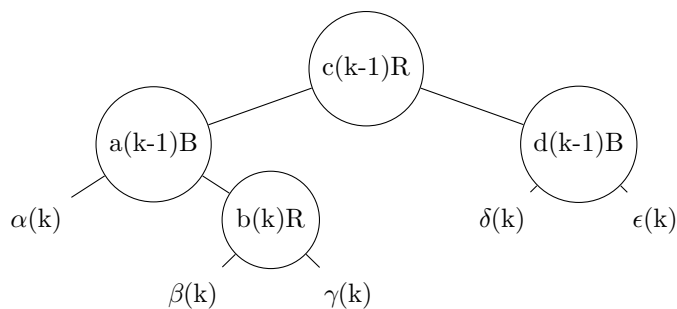
1: function PRE-PROCESS( $A, k$ )
2:    $C[0..k]$ 
3:   for  $i = 0$  to  $k$  do
4:      $C[i] = 0$ 
5:   for  $j = 1$  to  $A.length$  do
6:      $C[A[j]] = C[A[j]] + 1$ 
7:   for  $i = 1$  to  $k$  do
8:      $C[i] = C[i] + C[i - 1]$ 
9:    $A = C$ 
10: function RANGE( $A, k, a, b$ )
11:   Pre-Process( $A, k$ )
12:   return  $A[b] - A[a]$ 

```

3. Problem 13.3-5 from CLRS. (Describe semi-formally.) (*Hint: Follow the structure for an invariant.*)



13.3 (a)



13.6

4. **(Previous exam question)** Let $A[1..n]$ be an array of non-integers taken from some set K of size $k > 1$. (Note: For this problem, you are not given the set K or k ; this is only to illustrate that there are k distinct non-integer numbers. We only have access to elements through A . Further, note that k may be small or large: from constant to even larger than n .)

- Describe an algorithm that sorts A in expected time $O(n + k \lg k)$, and describe why it has this running time.
- What is the worst-case running time of your algorithm? Justify your answer.