

# deaps

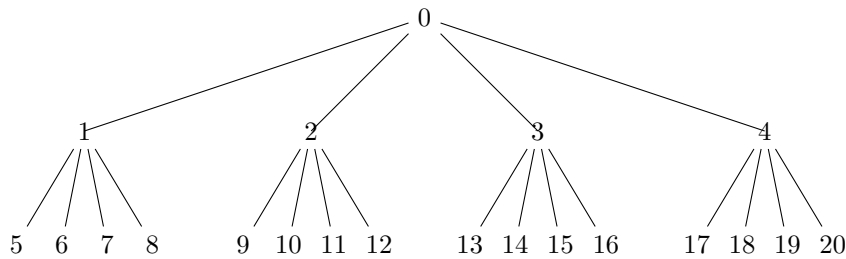
## CSCI 112, Lab 9

**d-ary heaps:** A **d-ary heap** is like a binary heap, but (with one possible exception) non-leaf nodes have  $d$  children instead of 2 children.

**Indexing in the array:** • The root of the tree is stored at position 0.

- The  $d$  children of node at position  $n$  are stored at positions  $dn + i$  for  $i = 1, \dots, d$ .

For example, in a 4-ary heap, the root is at 0, the children of 0 are at  $0 + 1 = 1$  through  $0 + 4 = 4$ , the children of 1 are at  $4 + 1 = 5$  through  $4 + 4 = 8$ , and so on, as in the figure:



- To find the parent of a node at  $i$ , take  $\lfloor ((i - 1)/d) \rfloor$ . Some examples from the tree above:

$$\text{parent}(8) = \lfloor ((8 - 1)/4) \rfloor = \lfloor 7/4 \rfloor = \lfloor 1.75 \rfloor = 1$$

$$\text{parent}(9) = \lfloor ((9 - 1)/4) \rfloor = \lfloor 2 \rfloor = 2$$

$$\text{parent}(10) = \lfloor ((10 - 1)/4) \rfloor = \lfloor 9/4 \rfloor = 2$$

What is the height of a  $d$ -ary heap of  $n$  elements in terms of  $n$  and  $d$ ?

As can be seen by examining the above figure, the maximum number of elements that can be stored in a  $d$ -ary heap of height  $h$  is

$$\max(n) = \sum_{i=1}^h d^i = \frac{d^{h+1} - 1}{d - 1}$$

In the above example,  $d = 4$ ,  $h = 2$ , giving

$$\frac{d^{h+1} - 1}{d - 1} = \frac{4^{2+1} - 1}{4 - 1} = \frac{64 - 1}{3} = 21$$

So it checks out.

To find  $h$ , given  $d$  and  $n$ , we need to find the smallest  $h$  such that the above fraction is greater than or equal to  $n$ , in other words,

$$h = \min \left\{ h : \frac{d^{h+1} - 1}{d - 1} \geq n \right\}$$

Working with that inequality, we want the smallest  $h$  such that

$$\begin{aligned}
\frac{d^{h+1} - 1}{d - 1} &\geq n \\
d^{h+1} - 1 &\geq n(d - 1) \\
d^{h+1} &\geq n(d - 1) + 1 \\
\log_d(d^{h+1}) &\geq \log_d(n(d - 1) + 1) \\
h + 1 &\geq \log_d(n(d - 1) + 1) \\
h &\geq \log_d(n(d - 1) + 1) - 1
\end{aligned}$$

and hence

$$h = \lceil \log_d(n(d - 1) + 1) - 1 \rceil$$

Let's try that on our example. With  $d = 4$  and  $n = 21$  we get

$$\begin{aligned}
h &= \lceil \log_4(n(d - 1) + 1) - 1 \rceil \\
&= \lceil \log_4((21)(3) + 1) - 1 \rceil \\
&= \lceil \log_4(63 + 1) - 1 \rceil \\
&= \lceil \log_4(64) - 1 \rceil \\
&= \lceil 3 - 1 \rceil \\
&= 2
\end{aligned}$$

So that works. With one more item, though,  $d = 4$  and  $n = 22$ , and we get

$$\begin{aligned}
h &= \lceil \log_4(n(d - 1) + 1) - 1 \rceil \\
&= \lceil \log_4((22)(3) + 1) - 1 \rceil \\
&= \lceil \log_4(66 + 1) - 1 \rceil \\
&= \lceil \log_4(67) - 1 \rceil \\
&= \lceil 3.033 - 1 \rceil \\
&= 3
\end{aligned}$$

So, yes, with one more item we will need another level in the height of our tree, which is obvious from the example and so our formula checks out here, too.

- We follow one path from root to leaf, looping over up to  $d$  children at each level, so running time is  $O(d \log_d n)$ . We loop over  $d$ , but for a  $d$ -ary heap  $d$  is a constant and does not depend on the input, so the loop is constant for all inputs, so we could say  $O(\log_d n)$

Give an efficient implementation of INSERT in a  $d$ -ary max-heap. Analyze its running time in terms of  $d$  and  $n$ .

- Nothing has to be changed as long as we use the new definition of PARENT. Time is  $O(\log_d n)$ .

Give an efficient implementation of INCREASE-KEY( $A, i, k$ ), which flags an error if  $k < A[i]$ , but otherwise sets  $A[i] = k$  and then updates the  $d$ -ary max-heap structure appropriately. Analyze its running time in terms of  $d$  and  $n$ .

- Nothing has to be changed as long as we use the new definition of PARENT. Time is  $O(\log_d n)$ .