Handin 3

Thomas Vinther & Jens Kristian Refsgaard Nielsen

19-09-18

Analysis of 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.

Subtask a.) How would you represent a d-ary heap in an array
We represent the d-ary heap as an array by maintaining the heap property $A[Parent(i)] \geq A[i]$ but we change the Parent function as follows
Parent(i)

Time	Line nr	Code
1	1	if i < d+2
d	1	return 1
1	3	m = 0
n/d	4	while i > md+1
1	5	m++
1	6	return m

This amounts to the function

$$\text{Parent}(i) = \begin{cases} 1 & \text{if } i < d+2 \\ m & \text{if } i = (m-1)d+2, \ldots, md+1 \end{cases} \tag{0.1}$$

With runtime $O(n/d)$, we will later see the importance of keeping the n/d notation.

Subtask b.) What is the height of a d-ary jeap of n elements in terms of n and d
The zeroth level of a d-ary tree has 1 element, this element has d children, and each of these in turn have d children, so in total we end up having k^h nodes at the h'th level. Now it is clear that the height of the tree is $O(\log_d(n))$.

Subtask c.) Give an efficient implementation of EXTRACT-MAX in a d-ary max-heap. Analyse its running time in terms of d and n.
As HEAP-EXTRACT-MAX does not in itself have anything to do with the -aryity of the tree no modification is necessary. However HEAP-EXTRACT-MAX calls MAX-HEAPIFY(A,1), which we indeed need to modify to MAX-HEAPIFY'(A,m):

Time	Line nr	Code
1	1	largest = m
d	2	kids = [(m-1)d+2,(m-1)d+3,...,dm+1]
d	3	for k in kids
2	4	if k $\leq$A.heapsize and A[k] > A[largest]
1	5	largest = k
1	6	if largest $\neq$ m
3	7	exchange A[m] with A[largest]
a	8	MAX-HEAPIFY'(A,largest)

Now consider the runtime, line 1 through 7 takes $1 + d + d(2(1)) + 1(3) = O(d)$ so the call in line 8 is also $a = O(d)$. In the worst case we have to MAX-HEAPIFY' once for each layer of the tree, and we've seen that this was $\log_d(n)$ so we get a total runtime of $O(d \log_d(n)) = O(\log_d(n))$.

Subtask d.) & e.) We wish to modify MAX-HEAP-INSERT to work on our d-ary trees.
The base kit for MAX-HEAP-INSERT(A,key) will work without modification. However the HEAP-INCREASE-KEY(A,i,key) will need modification as follows

Time	Line nr	Code
1	1	if key < A[i]
1	2	error
1	3	A[i] = key
$\log_d(n)$+n/d	4	while i > 1 and A[Parent(i)]>A[i]
3+n/d	5	exchange A[i] with A[Parent(i)]
1+n/d	6	i = Parent(i)

The n/d is from the Parent function. In total we get

$$\begin{aligned} \sum_{j=1}^{\log_d(n)} \frac{n}{d^j} &= n \sum_{j=1}^{\log_d(n)} \frac{1}{d^j} \\ &= n\left(\frac{1}{d-1} - \frac{1}{d^{\log_d(n)}(d-1)}\right) \\ &= \frac{n-1}{d-1} \end{aligned}$$

Here the $\frac{n}{d^j}$ summands represent the Parent function being called j times on n, because the parent funciton pretty much just represents division by d. And we call it $\log_d(n)$ times. In total the new version runs in linear time.