

# Problem session 3

Dibran Dokter 1047390 & Marnix Lukasse 1047400

September 26, 2019

## 4

### 4.1

We can transform a array that represents a max-heap into a array representing a min-heap by following process:

For each (initially) non-leaf node  $N$ , starting at the first non-leaf node moving towards the root of the tree (index in array =  $\lfloor n/2 \rfloor - 1$  where  $n$  is number of nodes/array length):

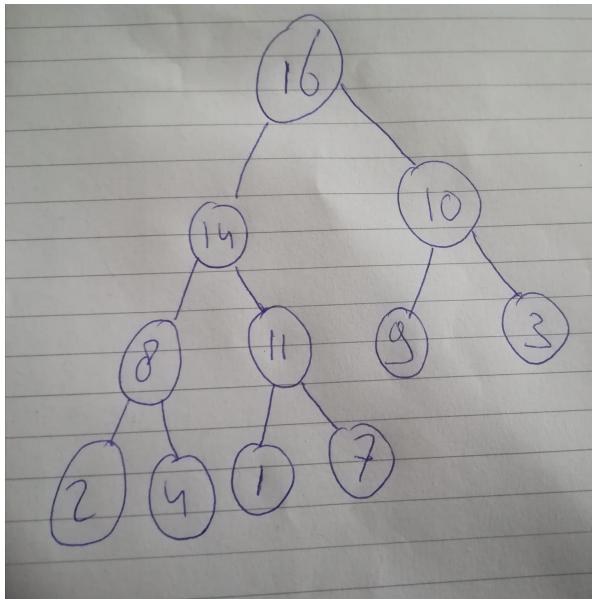
Check which of its child nodes is the lowest, swap places with that node. Now check if node  $N$  has any child nodes, and if so: is any of these child nodes lower than  $N$ ? If so, swap places with that node. Keep continuing this process until node  $N$  has no more conflicting child nodes (regarding the min-heap property).

After this is done for all initial non-leaf nodes, the array/tree should be a min-heap. The complexity of this process is  $\mathcal{O}(n \cdot \log(n))$ , namely for 0.5 of the nodes we have to start swapping down levels (in worst case swapping  $\log(n)$  times, as  $\log(n)$  is the number of levels in the tree).

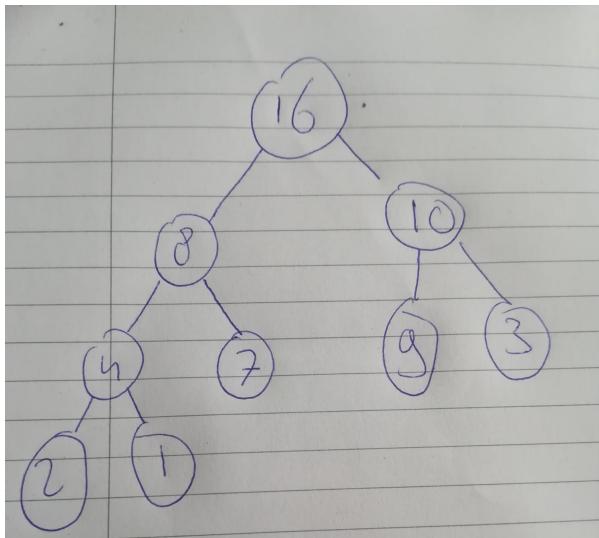
For some reason we are able to reduce this  $\mathcal{O}(n \log n)$  complexity to  $\mathcal{O}(n)$  by doing some math, the professor showed this in the lecture.

## 4.2

1)



2)

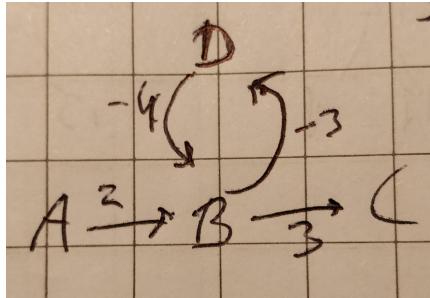


## 4.3

Q:	0	1	2	3	4	5
	0	30	40	$\infty$	$\infty$	$\infty$
			40	50	80	$\infty$
				50	70	$\infty$
					70	110

#### 4.4

The following graph will not find the shortest path since the theoretical shortest path is  $-\infty$ . This is caused by the loop from B  $\rightarrow$  D that has a negative weight. In this case it will try to find the shortest path by looping through this loop infinitely many times.



#### 4.5

We can use a slightly altered version of Dijkstra for this problem, lets call it Dijkstra\*. With Dijkstra\* you do the normal Dijkstra routine, but you don't just keep track of the distance cost to a certain node, you add the number of hops needed to reach that distance.

Normally if Dijkstra finds a different route to a node, it checks 'This route cost < Known route cost?' With Dijkstra\* this check would become: '(This route cost < Known route cost) OR ((This route cost == known route cost) AND (This route number hops < known route hops)).

If this check is true, then the cost/hop info should be updated. If we do Dijkstra\* for every vertex, we can collect all the easiest arrays by looking at the hop counts.

There might be optimizations possible so we don't have to run the full Dijkstra\* for every vertex. If the graph is undirected, we could use the information we already know. For example, if we already did Dijkstra\* with node A as starting point, and easiest A->B turned out to be 2, we can already fill in this hop count of 2 for Dijkstra\* with B as starting point and we don't have to actually check any routes from B to A, we already know the shortest one is with hop count 2.

#### 4.6

To solve this problem we first run Dijkstra with the chosen  $v$  as the source node. This will give us the shortest path from  $v$  to all other vertices.

Then we run Dijkstra for all the other vertices and note the shortest path from every vertex to  $v$ .

After this we can add those paths to get the shortest path from any chosen vertex to any other vertex via  $v$ . The complexity of this algorithm is  $|V| \cdot |V|^2 = |V|^3$  when using arrays.

This complexity can be improved to  $|V| \cdot |E| + |V| \lg |V|$  when using a Fibonacci heap.

#### 4.7

We can solve this problem using the following algorithm:

Network of roads  $G = (V, E)$

```
1     For each r in list:  
2         {  
3             Include r in a copy of G
```

```
4     Run dijkstra from starting point s  
5 }
```

Then you are able to compare the results, and check in which version the cost (from s) to t is the lowest. The extra road r in this version is the best to build if the goal is to minimize cost  $s \rightarrow t$

Note that we assumed that a road is undirected, if the roads are directed then we should run Dijkstra from both s as starting point and t. Then take the average of these results or something to see what extra road gives us the best overall improvement.