IE 515 - Homework 1

Ahmet Yiğit Doğan

November 2, 2023

Problem 1

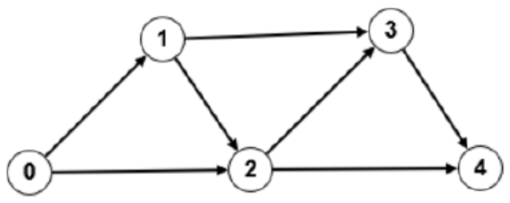

Figure 1: The graph for the first problem.

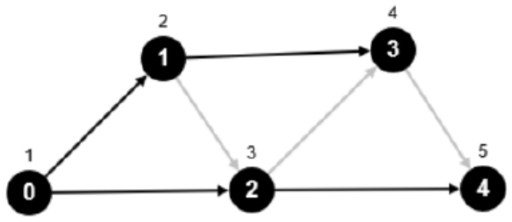

Figure 2: Breadth First Search with the traversal order 0, 1, 2, 3, 4.

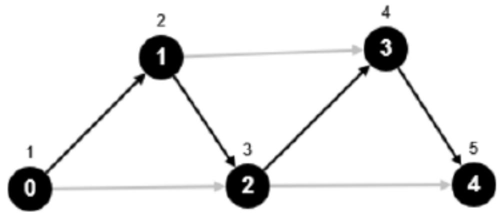

Figure 3: Depth First Search with the traversal order 0, 1, 2, 3, 4.

Problem 2

When all arc length are the same in the examined graph, the shortest path problem becomes suitable for solving with straightforward approaches like Breadth First Search and Depth First Search. The only extra thing to be done is recording the traversed path. BFS modified in such sense would be as follows:

Algorithm 1 BFS for shortest path in unweighted network.

```
Unmark all nodes
Mark s
LIST {s}
$d(i) = \infty, \forall i \in N$
$d(s) = 0$
while LIST $\neq \emptyset$ do;
    pick $i$, the very first node in LIST
    if $i$ is incident to an admissible arc $(i, j)$ then
        mark node $j$
        add node $j$ to LIST
        $d(j) = d(i) + 1$
    else
        delete $i$ from LIST
    end if
end while
```

After running the algorithm, d is going to include the shortest path distances from the source node to all the other nodes. The time complexity of the algorithm is $O(n + m)$, as in the standard BFS.

Problem 3

Intuitively, the problem can be adapted to a network as in Figure 4. However, this setting does not include all possible solutions such as (9-13, 12-15, 14-17) which also covers every single hour in the time horizon. Thus, the network can be modified such that it allows backtracing to the previous hours with no cost. Note that going back to 9 and backtracing after reaching 17 is not necessary when we set the source node to 9 and the sink node to 17. Therefore, these arcs can be ignored. The resulting network is presented in Figure 5.

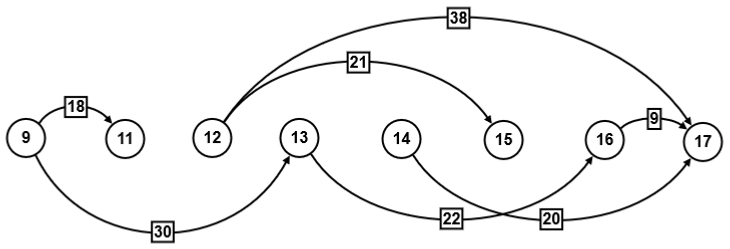

Figure 4: The network with no backtracing options.

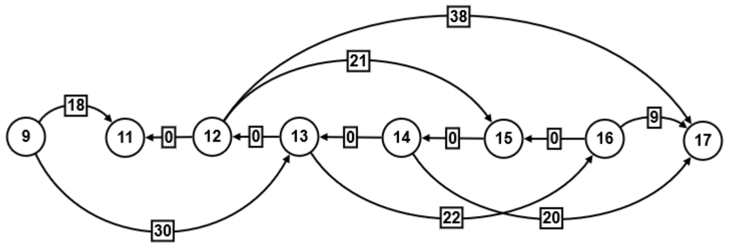

Figure 5: The network with backtracing options.

Iteration	Sets	d(9)	d(11)	d(12)	d(13)	d(14)	d(15)	d(16)	d(17)
0	S={} $\bar{S}$={9, 11, 12, 13, 14, 15, 16, 17}	0	∞	∞	∞	∞	∞	∞	∞
1	S={9} $\bar{S}$={11, 12, 13, 14, 15, 16, 17}	0	18	∞	30	∞	∞	∞	∞
2	S={9, 11} $\bar{S}$={11, 12, 13, 14, 15, 16, 17}	0	18	∞	30	∞	∞	∞	∞
3	S={9, 11, 13} $\bar{S}$={12, 14, 15, 16, 17}	0	18	30	30	∞	∞	52	∞
4	S={9, 11, 13, 12} $\bar{S}$={14, 15, 16, 17}	0	18	30	30	∞	51	52	68
5	S={9, 11, 13, 12, 15} $\bar{S}$={14, 16, 17}	0	18	30	30	51	51	52	68
6	S={9, 11, 13, 12, 15, 14} $\bar{S}$={16, 17}	0	18	30	30	51	51	52	68
7	S={9, 11, 13, 12, 15, 14, 16} $\bar{S}$={17}	0	18	30	30	51	51	52	61

Table 1: Dijkstra iterations for the bus scheduling problem.

The problem can be solved with Dijkstra Algorithm since it includes cycles but no negative arc costs. The Dijkstra iterations are depicted in Table 1.

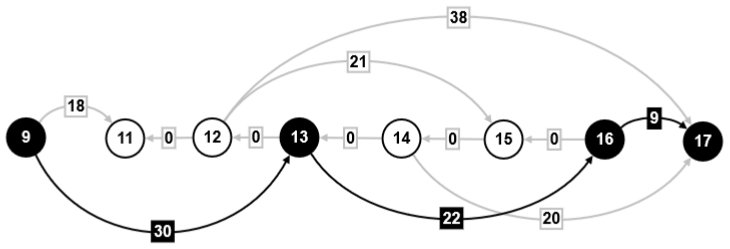

Figure 6: Shortest path from 9 to 17.

Problem 4

Output of the program:

```
> Iteration 0
> Permanent Set:
> []
> Temporary Set:
> [9, 11, 12, 13, 14, 15, 16, 17]
> d:
> {9: 0, 11: None, 12: None, 13: None, 14: None, 15: None, 16: None, 17: None}
>
>
> Iteration 1
> Permanent Set:
> [9]
> Temporary Set:
> [11, 12, 13, 14, 15, 16, 17]
> d:
> {9: 0, 11: 18.0, 12: None, 13: 30.0, 14: None, 15: None, 16: None, 17: None}
>
>
> Iteration 2
> Permanent Set:
> [9, 11]
> Temporary Set:
> [12, 13, 14, 15, 16, 17]
> d:
> {9: 0, 11: 18.0, 12: None, 13: 30.0, 14: None, 15: None, 16: None, 17: None}
>
>
> Iteration 3
> Permanent Set:
> [9, 11, 13]
> Temporary Set:
> [12, 14, 15, 16, 17]
> d:
> {9: 0, 11: 18.0, 12: 30.0, 13: 30.0, 14: None, 15: None, 16: 52.0, 17: None}
>
>
> Iteration 4
> Permanent Set:
> [9, 11, 13, 12]
> Temporary Set:
> [14, 15, 16, 17]
> d:
> {9: 0, 11: 18.0, 12: 30.0, 13: 30.0, 14: None, 15: 51.0, 16: 52.0, 17: 68.0}
>
>
> Iteration 5
> Permanent Set:
> [9, 11, 13, 12, 15]
> Temporary Set:
> [14, 16, 17]
> d:
> {9: 0, 11: 18.0, 12: 30.0, 13: 30.0, 14: 51.0, 15: 51.0, 16: 52.0, 17: 68.0}
>
>
> Iteration 6
> Permanent Set:
> [9, 11, 13, 12, 15, 14]
> Temporary Set:
> [16, 17]
> d:
> {9: 0, 11: 18.0, 12: 30.0, 13: 30.0, 14: 51.0, 15: 51.0, 16: 52.0, 17: 68.0}
>
>
> Iteration 7
> Permanent Set:
> [9, 11, 13, 12, 15, 14, 16]
> Temporary Set:
> [17]
> d:
> {9: 0, 11: 18.0, 12: 30.0, 13: 30.0, 14: 51.0, 15: 51.0, 16: 52.0, 17: 61.0}
>
>
> The shortest path from 9 to 17 is 9 -> 13 -> 16 -> 17 with the cost 61.0.
>
```

Problem 5

The problem requires three basic heap operations: building, removing the minimum, decreasing label

(a) To create a heap from a given array, each element should be appended to an empty heap one by one and meanwhile, siftup procedure should be applied to maintain the heap property:

```
procedure siftup($i$);
begin
while $i$ is not a root node and key($i$) $<$ key(pred($i$)) do;
    swap($i$, pred($i$));
end while
end;
```

Figure 7: Procedure siftup(i).

(b) To remove the minimum item of a min heap, the root node should be swapped with the rightmost item of the bottom layer. Then it should be dropped. To maintain the heap property, siftdown procedure should be applied to the new root node:

```
procedure siftdown($i$);
begin
while $i$ is not a leaf node and key($i$) $>$ key(minchild($i$)) do;
    swap($i$, minchild($i$));
end while
end;
```

Figure 8: Procedure siftdown(i).

(c) Decreasing a label also requires the siftup procedure to be applied to the same label after the update.

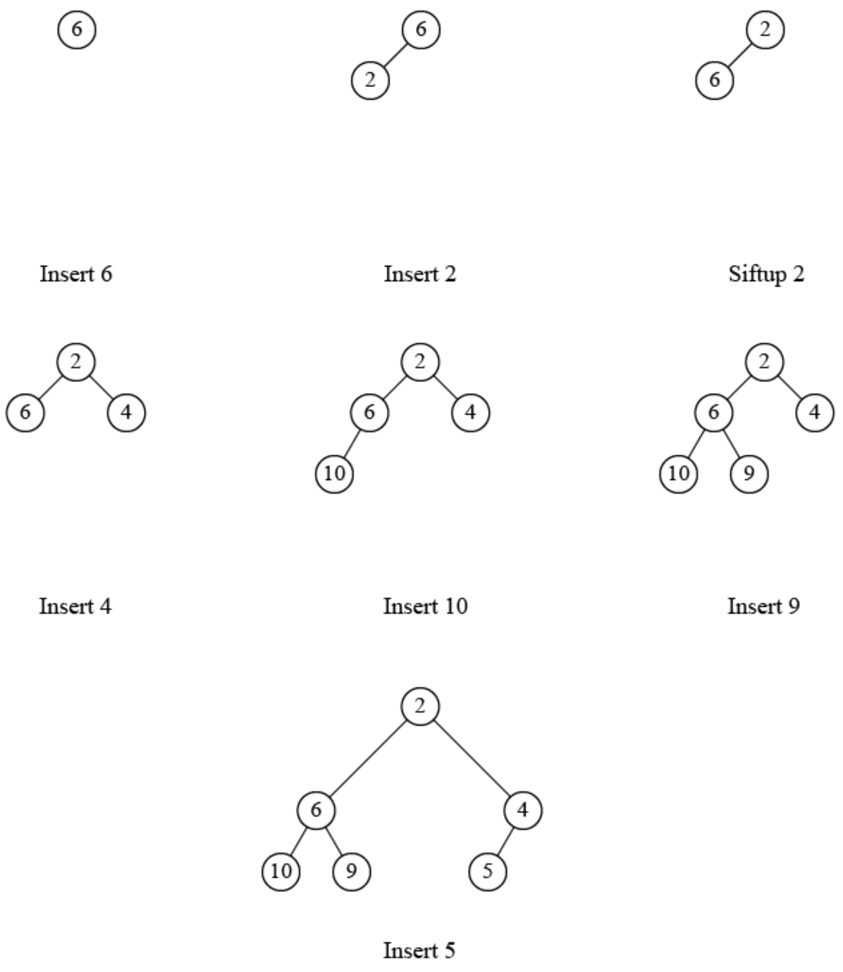

Figure 9: Steps of building a heap from the array {6, 2, 4, 10, 9, 5}.

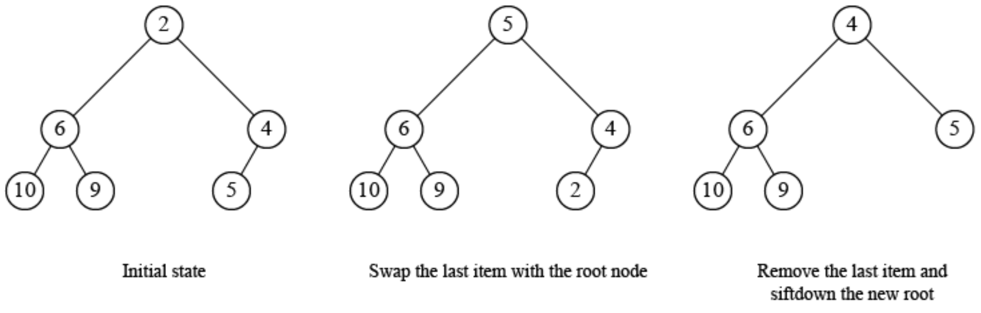
Initial state
Swap the last item with the root node
Remove the last item and siftdown the new root

Figure 10: Steps of the first remove min operation.

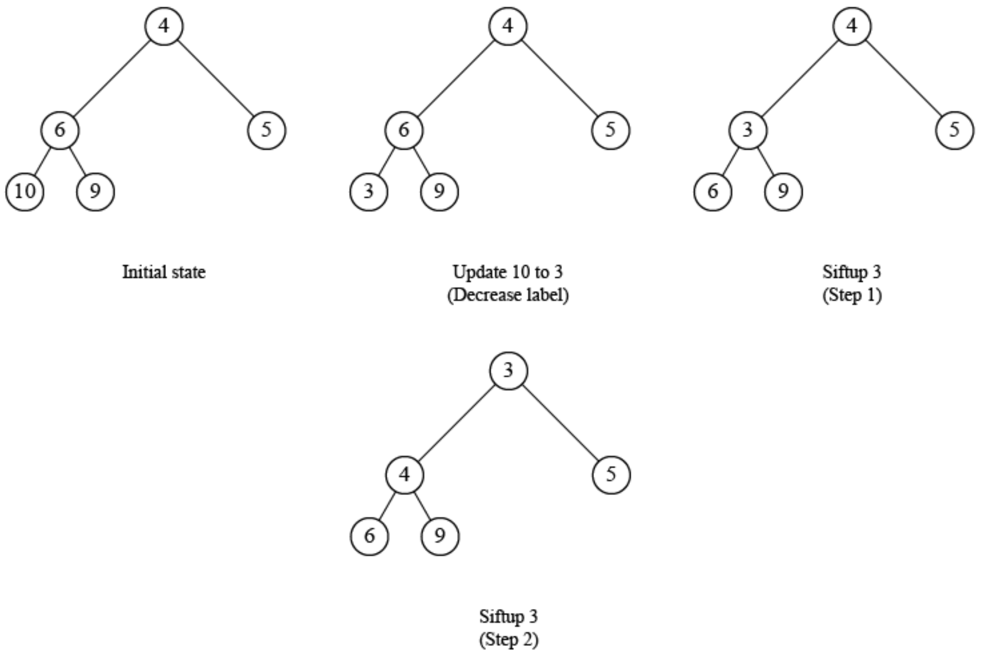

Figure 11: Steps of the first label update.

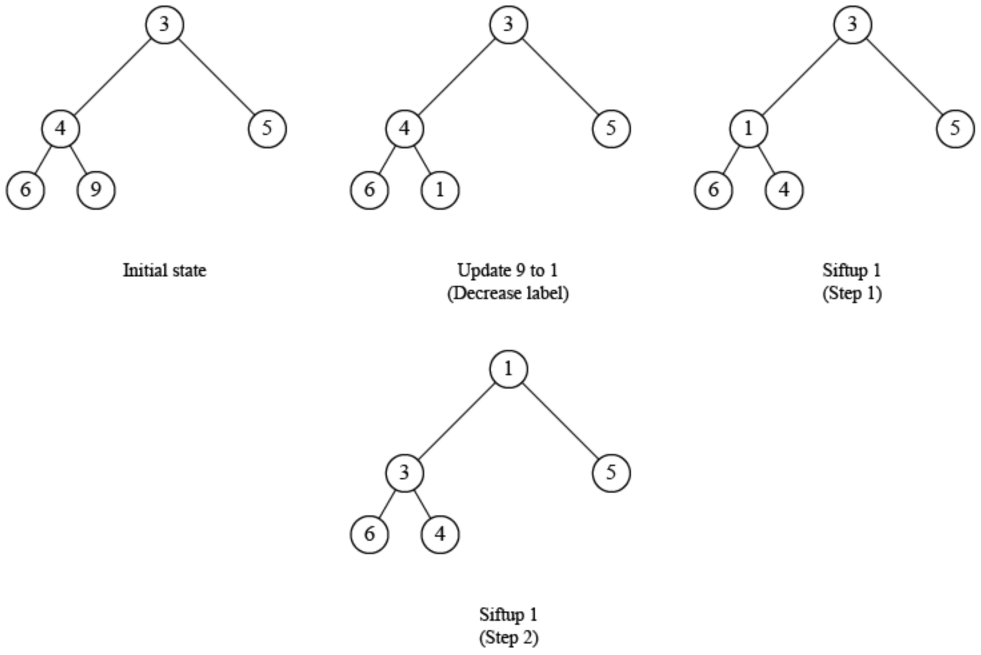

Figure 12: Steps of the second label update.

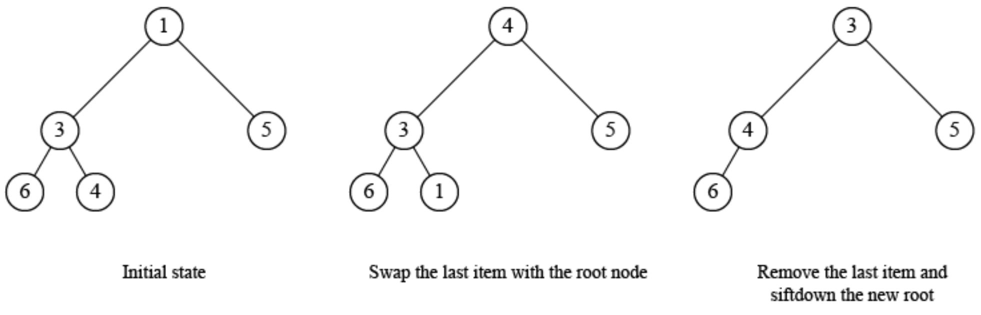
1
3
5
6
4
Initial state
4
3
5
6
1
Swap the last item with the root node
3
4
5
6
Remove the last item and siftdown the new root

Figure 13: Steps of the second remove min operation.