Solutions to Exam 2020

Exercise 1

1.1.1 (a)
$9^{\log_3 n} = n^{\log_3 9} \sim \Theta(n^2)$
$0.003 \cdot \lg^n 27 = 0.003 \cdot (lg27)^n \sim \Theta((lg27)^n)$
$7727n^{9.2} \sim \Theta(n^{9.2})$
$57lgn^5 = 57 \cdot 5lgn \sim \Theta(lgn)$

1.1.2. (b)
$n^2 \lg n$ is $\Theta(n^2 \lg n)$, and thus $\Omega(n^2)$

2.1. (d) This can be solved by the master method, the third case.
2.2. (b) $a = 1$, $b = \frac{4}{3}$, and $f(n) = \sqrt{n}$. Since $f(n) = \sqrt{n} = n^{0.5} = \Omega(n^{\log_{\frac{4}{3}} 1+\epsilon}) = \Omega(n^{0+\epsilon})$ for some constant $\epsilon > 0$, e.g., $\epsilon = 0.5$.

The regularity condition: $a \times f(n/b) \leqslant c \times f(n) \rightarrow 1 \times \sqrt{3n/4} \leqslant c \times \sqrt{n} \rightarrow 1 \times \sqrt{3/4} \times \sqrt{n} \leqslant c \times \sqrt{n}$, where some constant $c < 1$ exists. We should consider case 3.
2.3. (c) Case 3: the solution is $T(n) = \Theta(f(n)) = \Theta(\sqrt{n})$.

3.1. (b)
(a) The worst case time complexity of merge sort is $\Theta(n \lg n)$.
(b) The average case time complexity of quick sort is $\Theta(n \lg n)$.
(c) The best case time complexity of selection sort is $\Theta(n^2)$
(d) Quick sort is an in place sorting algorithm.

3.2. (a) selection sort
After $i = 1 : [2, 54, 18, 26, 9, 17, 45]$
After $i = 2 : [2, 9, 18, 26, 54, 17, 45]$
After $i = 3 : [2, 9, 17, 26, 54, 18, 45]$

4.1. (d) See Figure 1.
4.2. (c) See Figure 2.

5.1. (c)
Level 1: put a to the tree.
Level 2: put b and c to the tree (the order does not matter).
Level 3: put d and e to the tree (the order does not matter).
Level 4: must visit d's adjacent vertex g (such that d can be marked as black), and then visit e's adjacent vertices. Therefore, g is d's child, but not e's. Therefore, **(a)** and **(b)** are wrong.
(d) is wrong, because h, whose color is white, should not be included in the breath first tree.

5.2. (a)
h will be visited as e's adjacent vertex.

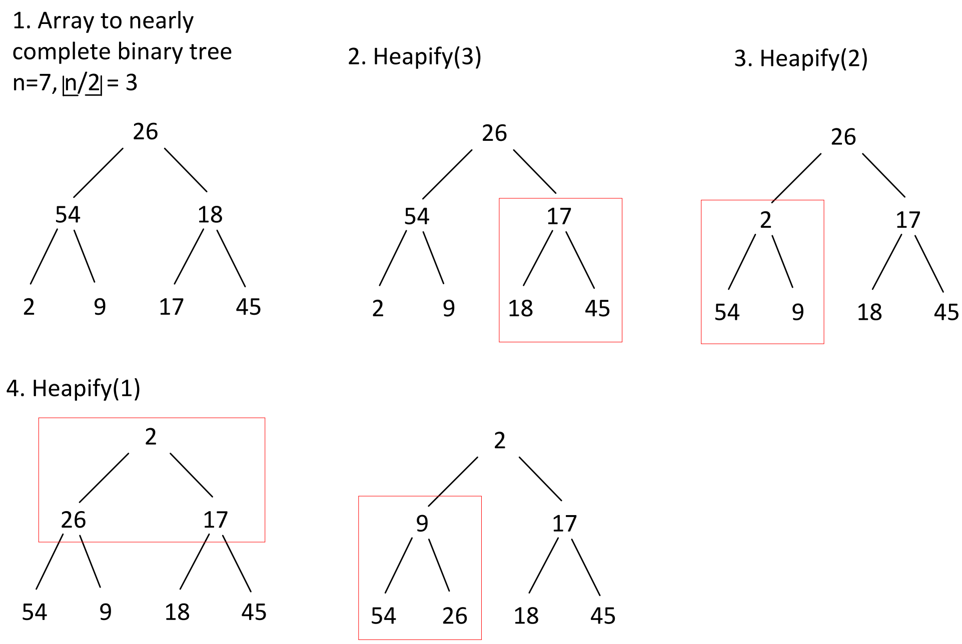

Figure 1: Solution for 4.1

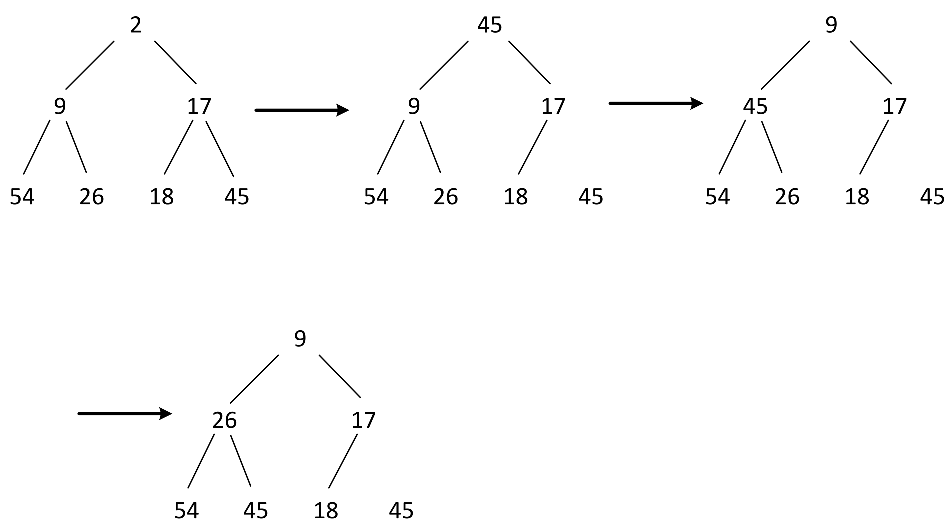

Figure 2: Solution for 4.2

6.1 (c)
6.2 (d) The adjacency matrix represents the graph, as shown in Figure 3.

(a) f should not appear before h.
(b) b should not appear before a.
(c) h should not appear before a.

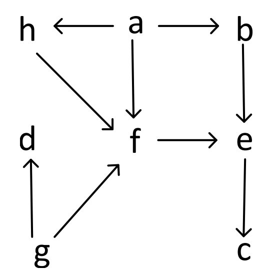

Figure 3: Solution for 6.2

Ex. 2

1. (*7 points*) $m[1] = 0$
   $m[i] = \min_{k<i\leqslant n}(m[k] + f_{k,i})$, $1 < i \leqslant n$.

2. (*7 points*) Pseudo code:

   F is the fee table, and we denote the value at row i and column j as $F[i, j]$, meaning the travel fee from station i to station j, with $1 \leqslant i < j \leqslant n$.

```
BOTTOM-UP(F)
m[1] ← 0
for i = 2 to n
    m[i] ← ∞
    for k = 1 to i − 1
        if (m[i] < m[k] + F[k, i])
            m[i] ← m[k] + F[k, i]
Return m[n]
```

3. (*6 points*) Run time $\Theta(n^2)$. 2 for loops.

Exercise 3

Question set A:

1. (*4 points*)

   a: b, 2; c, 13;
   b: e, 5; d, 3;
   c: $\emptyset$
   d: a, 2; c, 4; e, 1
   e: c, 4

2. (*16 points*)

   (a) (*4 points*) Dijkstra's alg. Because Dijkstra's alg. is more efficient than Bellman ford; and Alg. for DAG cannot deal with circles.

   (b) (*6 points*) abdc, 9.

   (c) (*6 points*) Number of edge relaxations that help to improve existing distances: 6.

   Relaxing edges ab, ac, bd, be, da (but not improving distance), dc, de, ec (but not improving distance).

Question set B:

1. (*4 points*)

   0 2 13 0 0
   0 0 0 3 5
   0 0 0 0 0
   2 0 4 0 1
   0 0 -6 0 0

2. (*16 points*)

   (a) (*4 points*) Bellman ford alg. Because Bellman ford alg. can deal with negative weights, but Dijkstra's algorithm cannot. And Alg. for DAG cannot deal with circles.

   (b) (*6 points*) abdec, 0.

   (c) (*6 points*) Number of edge relaxations: $(|V|-1)*|E|$= 4*8=32.