Solutions to Exam 2019 Exercises

Exercise 1

1.1.1 d)

1.1.2. b & d)

2 b, c)

3 d
Master method, 3rd case.

4.1 bcd

5 ab

6 e

7 d
`empty, 1, 10, 17, 13, 5, empty, empty, 8`
8 ad There is a bug in the question. So if you only choose a, or only choose d, or choose both a& d, you all get full points.

Ex. 2.1
3 points, $\Theta(2^n)$
2 points, $\Theta(2^5)$
Ex. 2.2
3 points Basic ideas: we define points(k, s) as the maximum points the student can get when solving questions $\{Q_1, Q_2, \ldots, Q_k\}$ within time s. The original problem is to solve points(N, X).
5 points Recurrence:

$$points(k, s) = max(points(k-1, s-t_k) + v_k, points(k-1, s))$$

the first item denotes the case where we solve question Q_k. since it takes t_k time, we need to substract the time, i.e., $s - t_k$, but we get v_k points already. the second item denotes the case where we do not solve question Q_k.
2 points pseudo code
5 points with DP, run time $\Theta(NX)$. Fill in a table of $N \times X$.

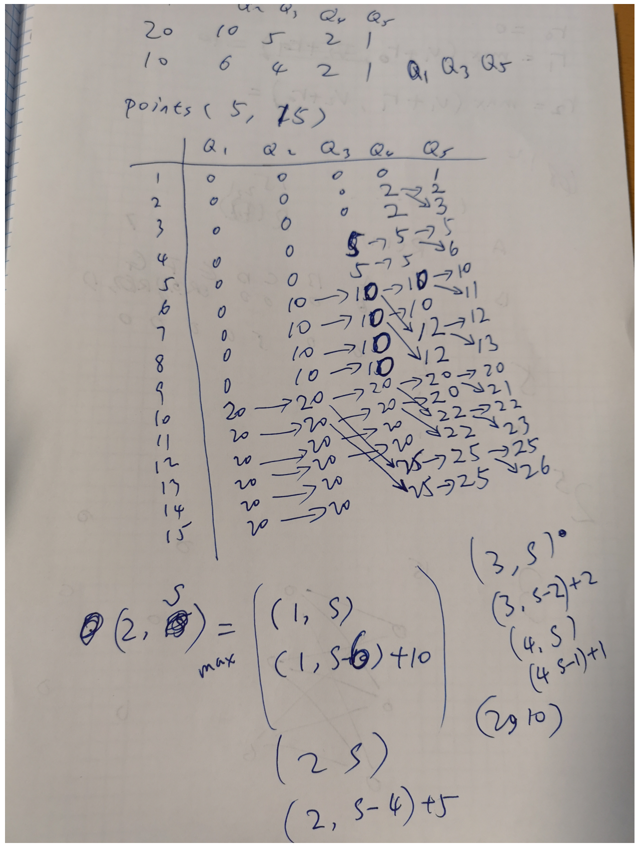

Figure 1: solutions to 2.1

Exercise 3

3.1 (*6 points*)

3.1.1 (*3 points*) The minimum cost is 9.

3.1.2 (*3 points*)

1. Choose village v_4 that has the minimum cost for building a power station, i.e., $c_4 = 3$.

2. Connect v_1 with v_4 and the connection cost is $t_{41} = t_{14} = 2$, which is smaller than $c_1 = 5$ if build a power station on v_1.

3. Connect v_2 with v_1 and the connection cost is $t_{12} = t_{21} = 2$, which is smaller than $c_2 = 4$ if build a power station on v_2.

4. Connect v_3 with v_1 and the connection cost is $t_{13} = t_{31} = 2$, which is smaller than $c_3 = 4$ if build a power station on v_3.

3.2 (*6 points*)

3.2.1 (*4 points*) The input is a connected, undirected weighted graph $G = (V, E, W, C)$, where

1. $V = \{v_1, \ldots, v_n\}$ is a vertex set,

MST-Prim-Modification (G)

```
1   for each vertex v_i ∈ G.V do
2       v_i.setKey(c(v_i))
3       v_i.setParent(NIL)
4   Q.init(G.V)
5   while not Q.isEmpty()
6       u ← Q.extractMin()
7       for each vertex v ∈ u.adjacent() do
8           if v ∈ Q and G.w(u, v) < v.key() then
9               v.setKey(G.w(u, v))
10              Q.modifyKey(v)
11              v.setParent(u)
```

2. $E = \{(v_i, v_j)\}$ is an edge set, with $i, j \in [1, n]$ and $i \neq j$,

3. W is a function that maps each edge (v_i, v_j) to a real value, denoted as $w(v_i, v_j)$, in order to model the connection cost, and

4. C is a function that maps each vertex v_i to a real value, denoted as $c(v_i)$, to model the cost of building a power station.

3.2.2 (*2 points*) The output is a minimum spanning tree (MST), where each vertex v_i has two attributes:

1. $v_i.parent$: v_i's parent in the MST to show how vertices are connected, and

2. $v_i.key$: the least weight that connects v_i to MST.

The sum of the key values of all vertices in MST is the minimum cost.

3.3 (*8 points*) The algorithm is a modification of Prim's algorithm, but differs in that (1) this algorithm does not have a random vertex to start with, but a vertex that has the least cost for building a power station to start with; and (2) the key value of a vertex v_i is not initialized with positive infinity, but with $c(v_i)$. The intuition is that if the connecting cost is smaller than building a power station, we choose Option 2 to connect two villages. If the connecting cost is bigger than building a power station, we choose Option1 to build a power station. However, we need to choose a vertex v_i that has the least $c(v_i)$ to start with.

3.3.1 (*6 points*) See MST-Prim-Modification (G).

3.3.2 (*2 points*) $O(lg|V| \times |E|)$, if the implementation of min-priority queue is binary min-heap.

1. Initialization, lines 1-3, takes $\Theta(|V|)$

2. Line 4 takes $O(V)$

3. Line 6 takes $O(lg|V|)$, a total of $O(lg|V|) \times |V|$ for $|V|$ iterations

4. Line 10 takes $O(1)$ or $O(lg|V|)$, a total of $O(1) \times |E|$ or $O(lg|V|) \times |E|$ for $|E|$ iterations

5. Summing up: $\Theta(|V|) + O(V) + O(lg|V| \times |V|) + O(lg|V) \times |E|)$, so $O(lg|V| \times |E|)$.