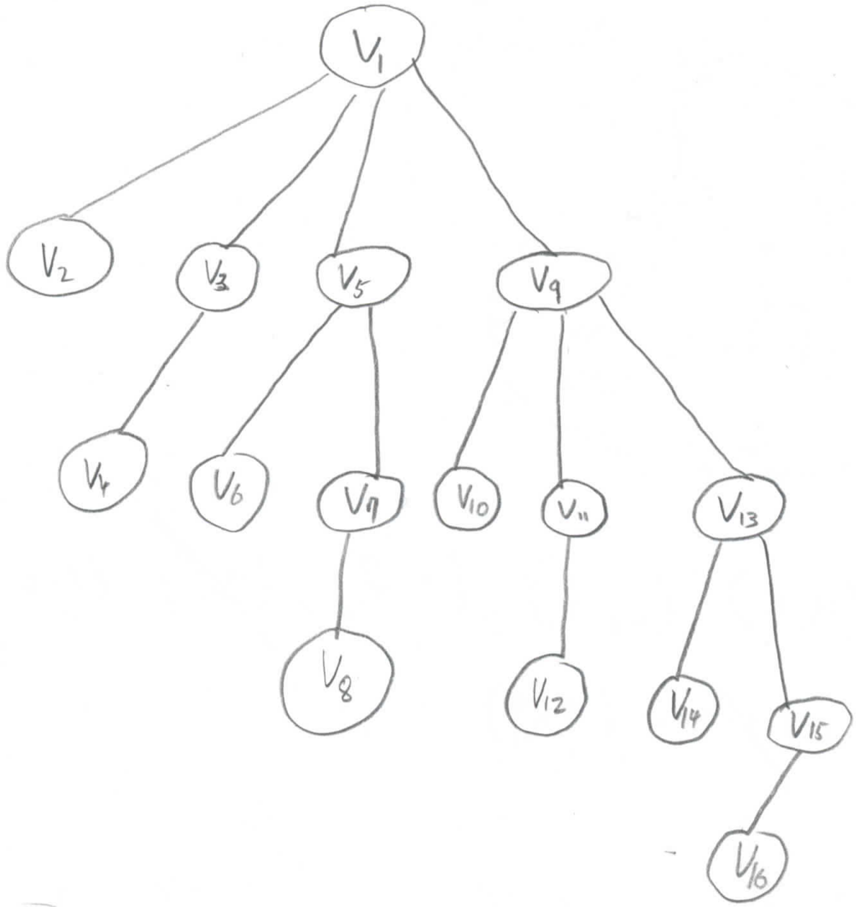
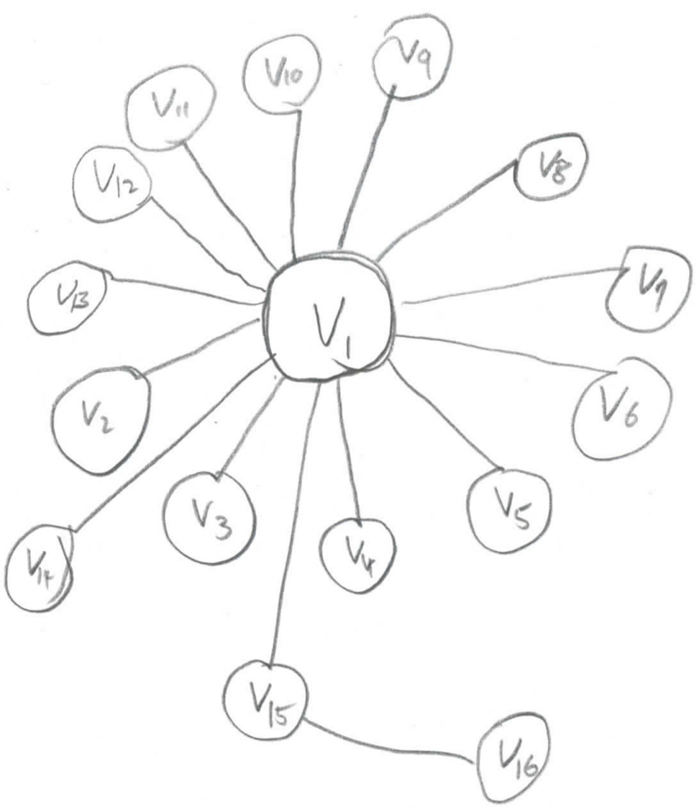
P. 2

1)



V_1
V_2
V_3
V_5
V_9
V_4
V_6
V_7
V_{10}
V_{11}
V_{13}
V_8
V_{12}
V_{14}
V_{15}
V_{16}

2)



V_{10}
V_9
V_{11}
V_{12}
V_8
V_{13}
V_1
V_7
V_2
V_6
V_3
V_5
V_{14}
V_4
V_{15}
V_{16}

3) $V_{12} \rightarrow V_{11} \rightarrow V_1$

4) The find operation in Union-Find distinguishes Union-Find from Union-by-rank since path-compression occur in the find operation.

Now, as shown in class the running time for Union-by-Rank is

$O(m \log n)$.

P. 3

See below.

Problem 3: The changes I made are underlined.

Let e_1, e_2, e_3, . . . denote the sequence of edges taken by the algorithm.
Assume that all edges have distinct edges as given in the hypothesis of the problem.
Claim: for each i, there exists an unique optimal spanning tree that contains e_1, e_2, . . . , e_i.
We use induction on i.
For i = 0, the claim is obvious.
Assume that the claim holds for i and let us prove it for i + 1.
By induction hypothesis, there exists an unique optimal spanning tree that contains e_1, e_2, . . . , e_i.
Let T_ denote such a tree. If T_ also contains e_{i+1}, then the claim holds for T_ and i + 1. Else, adding e_{i+1} to T_ creates a cycle C. There is at least one other edge e of C that is not e_1, e_2, . . . , e_i (because by construction, e_1, e_2, . . . , e_{i+1} does not have any cycle). Then T_0 = T_ − e + e_{i+1} is still a spanning tree, and its cost differs from the cost of T_ by $w_{e_{i+1}} - w_e$.
Since T_ is optimal, T_0 has weight greater than or equal to the weight of T_, so $w_{e_{i+1}}$ >= w_e. There are two cases: either $w_e < w_{e_{i+1}}$, or $w_e = w_{e_{i+1}}$.
In the first case, e would have been considered before e_{i+1} by the algorithm, and it does not create a cycle with e_1, . . . , e_i, so it would have been taken by the algorithm: contradiction.
In the second case, T_0 has cost equal to that of T_, so it is an optimal tree containing e_1, . . . , e_i, e_{i+1}, as desired.
However, now note that by assumption that all edges have distinct weights $w_{e_{i+1}} > w_{e\ i}$, and e_{i+1} is unique since any other edges not in the tree must have weight greater than (“not equal to”) e_{i+1}.
This proves the claim, hence the tree output by Kruskal’s algorithm has minimum weight and is unique if all edges have distinct weights.

P.4

The idea is that

1) Take the activities that starts closest to, but also before the previous end time.

2) Take the longest (one that ends the latest) among the activities selected by 1

3) repeat.

Here's implementation.

Let $a_i = (s_i, t_i)$ the start time and the end time

$a_i(s) = s_i$, $a_i(t) = t_i$.

S = the output set

Sort $\{a_i\}$ by the ascending order of the start time.

i.e) $a_1(s) \leq a_2(s) \leq \cdots \leq a_n(s)$.

$S := \emptyset$

$a := a_1$

For $i \leftarrow 1$ to $(n-1)$

if $a(s) = a_i(s)$ & $a(t) < a_i(t)$

$a \leftarrow a_i$

if $a(s) \neq a_i(s)$ & $a_{i+1}(s) > a(t)$

add a to S

$a \leftarrow a_i$

if $a_n(t) > a(t)$

add a_n to S