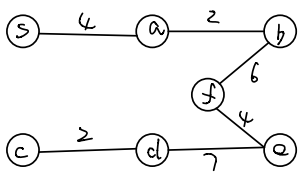
Exam II

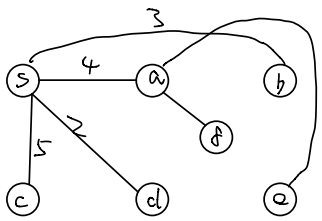
Yaoben Wu

1.

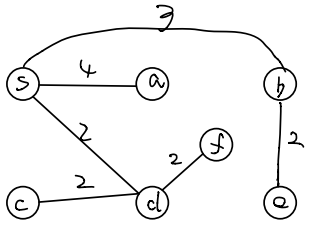
a).



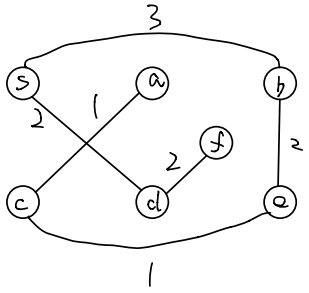
b).



c).



d).



2.

a. Coin set : 1, 5, 8

C : 11

b1. suppose that an optimal solution for change C is given such that

$$n_0 b^0 + n_1 b^1 + n_2 b^2 + \dots + n_k b^k = C$$

if all the coin are given is the largest the remaining can contain then it already optimal, trivial.

suppose an optimal solution is given when out the remaining change C', it skip c[i].

where $C' \geq c[i]$ where c[i] is the largest among c[0] ... c[i]

The since c[i] can be composed of some coins that smaller that c[i] and

$1 = \text{cost}[c[i]] < \text{cost}[c[j]] + \text{cost}[c[k]]$

where $c[j] + c[k] = c[i]$

If it doesn't use c[i] to reach the optimal solution, let cost[C] be the total number of coins when it doesn't include c[i]. Then by replacing number of coins that sums up to c[i] with a single c[i] coin. The new number is smaller than original that doesn't include c[i]. Therefore, the original one is not optimal, it is a contradiction.

3. a). We can modify Kruskal's algorithm such that the edges are sorted in decreasing order and each time we can get an edge with the highest weight. if it doesn't result in a cycle, we add the edge to F. and repeat.

Time complexity: $O(E \log V)$ if we use priority queue. since initially All edge are inserted into priority queue and takes $O(E \log E)$ time, then extracting required $O(\log E)$ time each and detecting cycle with disjoin set requires $O(\log V)$ time. each.

So $O(E \log E) + O(E \cdot (\log E + \log V)) = O(E \log E)$

$= O(E \log V^2) =$

$O(E \log V)$

b). MinimumFeedback(V, E, w)

$E' \leftarrow$ PartA(V, E, w)

for all edge uv

if uv is not in E'

add uv to F

return F

Time complexity: $O(E)$

4.

a).

$$\text{MaxLength}(v,r) = \begin{cases} 0 & \text{if } v=t, r=0 \\ -\infty & \text{if } v=t, r>0 \\ \max\limits_{v\to w} \{\max(\text{MaxLength}(w,r), \text{MaxLength}(w,r-1)) + l(v\to w)\} & \text{if } v \text{ is important} \\ \max\limits_{v\to w} \{\text{MaxLength}(w,r) + l(v\to w)\} & \text{if } v \text{ is not important.} \end{cases}$$

b). Assume part a is correct

Set of v: All vertices in the Graph.

Set of r: 0 to k

Evaluation order: post order

Return: MaxLength(s, k)

Time complexity: $O(\max(V+E, kE))$ because $O(V+E)$ to get post order, and for each i from 0 to k, it needs to traversal all edge once.