Assignment No. 3

Siddharaj Nandkumar Pujari
Ninad Parikh
Harsh Vora
Nihar Anant Darji

Problem - 1

Q.1. Choose the job that starts last.

→ We are given a set $S = \{a_1, a_2 \dots a_n\}$ of activities, where $a_i = \{s_i, f_i\}$. We assume to use strategy that choose the job that finishes first. We know that this yields an optimal solution. So we create another set S', $S' = \{a'_n, a'_{n-1} \dots a'_1\}$ and $a_i = \{f_i, s_i\}$ that means we begin in the last and end in the first so S and S' are of same format. According to chapter-16 from CLRS we see that if we choose a job that finishes first is optimal it is equal to choosing the job that starts last. Hence we can conclude that S' i.e. choosing the job that starts last is optimal.

PROOF:

In order to prove the job that starts last is optimal, we prove that a_m is an activity in S_k with the last start time, then a_m is included in some maximum-size subset of mutually compatible activities of S_K.

Let A_K be a maximum size subset of mutually compatible activities S_K and let a_i be the activity A_K with the last start time.

If $a_i = a_m$ it proves on basis of time.
If $a_i \neq a_m$, let set $A'_K = (A_K - \{a_j\}) \cup (a_m)$ be A_K Substituting a_m for a_i

The activities in A'_K are disjoint which follows our proof, because the activities in A_K are disjoint. Since $S_m \geq S_i$. Since $|A'_K| = |A_K|$ we conclude that A'_K is a maximum size subset of mutually compatible activities of S_K and includes a_m.

Hence, selecting the job that starts last is exactly same as the original problem. Imagining time running in reverse which produces the optimal solution for the same reason.
It is greedy because we make the best looking choice ~~ate~~ each step.

PROBLEM 1

Q.2. Choose the job that conflicts with the fewest other jobs

→ Suppose that our activities times are

i	1	2	3	4	5	6	7	8	9	10	11
Start S_i	-1	0	0	0	2	4	6	8	8	8	10
Finish F_i	1	3	3	3	5	7	9	11	11	11	12

Consider the following example above.

'a_6' is the activity with least number of conflicts

$a_6 = (4,7)$ has only 2 conflicts.

If we pick a_6 we will not get the optimal solution, because the optimal solution for the above problem is

$\{a_1, a_5, a_7, a_{11}\}$:- $\{(-1,1), (2,5), (6,9), (10,12)\}$

Hence it does not constructs an optimal schedule

PROBLEM 1.

Q.3 Choose the job of longest duration.

→ Consider anexample with following 3 activites:-

$\{(1,11), (4,6), (7,10)\}$

1) So if we pick the activity of longest duration amongst this, we will only have one activity i.e (1,11).

2) Whereas the optimal solution will pick 2 activities of timings $\{(4,6), (7,10)\}$

3) Hence, it does not constricts an optimal schedule.

* SUMMARY

PROBLEM 1

1) If we choose the job that starts last

→ Greedy choice of activities selection problem gives optimal schedule.

~~2~~

2) Choose the job that conflicts with the fewest other jobs.

→ Gives the non-optimal schedule.

3) Choose the job of longest duration.

→ Gives ~~the~~ ~~non-optin~~ non-optimal schedule.

PROBLEM 2.

Q. 1.

We have $c_1 = 1$, $c_2 = 5$, $c_3 = 10$, $c_4 = 25$, $c_5 = 50$, & $c_6 = 100$.

1. The greedy algorithm through recursive iterations always gives the highest denomination coin that you can without going over.
2. This process is repeated until the amount of remaining change drops to 0.

Proof:

We assume the best non-greedy solution for a given instance of problem is $(d_{100}, d_{50}, d_{25}, d_{10}, d_5, d_1)$

where, d_{100} = number of coins of 100 cents

d_{50} = number of coins of 50 cents.

$\vdots$

d_1 = number of coins of 1 cents.

Suppose, we are given N \$ in change with minimum number of coins.

$\therefore N = 100d_{100} + 50d_{50} + 25d_{25} + 10d_{10} + 5d_5 + d_1$

We need to show that greedy solution is good or better than the best solution.

If $c_{100}, c_{50}, c_{25}, c_{10}, c_5, c_1$ are no. of coins with those denominations.

To show,

$$C_{100} + C_{50} + C_{25} + C_{10} + C_5 + C_1 \leq d_{100} + d_{50} + d_{25} + d_{10} + d_5 + d_1$$

Since, the best solution is not greedy at some point there will be fewer coins of some denominations in the best solution as compared to greedy solution. ~~Now~~ Hence any combination of coins with lower denominations which make for the difference could be replaced with fewer coins.

Therefore, the best solution must be equal to the greedy solution.

* Method 2

Let N be the total amount to be paid

1. Let optimal solution =

$$P = A * 100 + B * 50 + C * 25 + D * 10 + E * 5 + F * 1$$

The optimal solution would have least number of coins.

So, while selecting coins in order to give optimal solution. Following cases can be considered:

PROBLEM 2

Q.1. 1. Instead of taking five coins of 1 we can always take one coin of 5.
2. Instead of taking two coins of 5 we can take one coin of 10.
3. Instead of taking three coins of 10 we can take one coin of 25 & one coin of 5.
4. Instead of 2 coins of 25 we can have one coin of 50.
5. Instead of two coins of 50 we will one coin of 100.

Inferring from 1 to 5
$F \leq 4$, $E \leq 2$, $D \leq 3$, $C \leq 2$, $B \leq 2$, $A \leq 1$ (Assuming optimal Solution)

2. Let Greedy Solution =
$P = a*100 + b*50 + c*25 + d*10 + e*5 + f*1$

From $0 \leq F \leq 4$ and $P = (A*100 + B*50 + C*25 + D*10 + E*5 + F*1)$ —— (1)
Similarly f, $0 \leq f \leq 4$
Therefore P (for greedy solution) $= a*100 + b*50 + c*25 + d*10 + e*5 + f*1$ —— (2)

From (1) & (2), $f = F$

Similarly $a = A$, $b = B$, $c = C$, $d = D$, $e = E$.
Hence solution given by greedy algorithm is same as solution given by optimal method.
$\therefore$ it is proved

For these denominations, greedy algorithms always gives the change with fewest coins.

PROBLEM 2.

Q.2.

1. Given an optimal solution $(x_0, x_1, \ldots x_k)$ where x_i indicates the number of coins of denominations b^i.

2. We will first show that we must have $x_i < b$ for every $i < k$.

3. Suppose that we had some $x_i \geq b$, then we could decrease x_i by b and increase ~~x_{i+1}~~ x_{i+1} by 1.

4. This collections of coins has same value and has $b-1$ coins less.

5. Therefore the original solution must of been non optimal

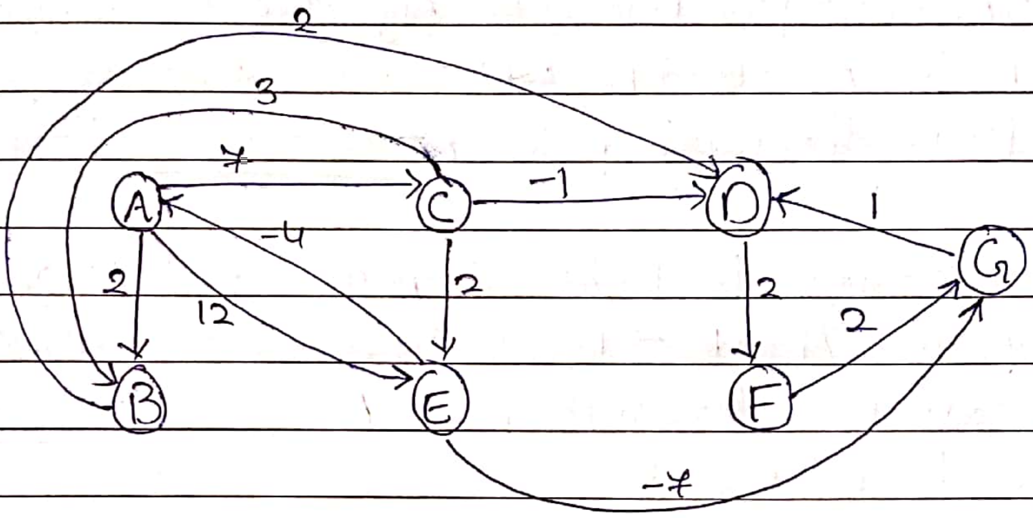
6. This configuration of coins is exactly same we will get if we did greedy picking algorithm of selecting the largest coin possible

7. Total value of coins V $x_K = \lfloor V_b^{-K} \rfloor$ and for $i < K$.

8. This is the only solution that satisfies the property of $V \bmod b^K$ and is achieved only through greedy algorithm.

9. Hence greedy algorithm always give the change with fewest coins.

Problem 3



Q.1.

Vertex	Known	Distance	Path
A	A	0	
B	B	~~∞~~ 2	A
C	D	~~∞~~ 7	A
D	F	~~∞~~ ~~2~~ 4	B
E	C	~~∞~~ ~~12~~, 9	A, C
F	G	~~∞~~ 6	D
G	E	~~∞~~ 8	F

The known column is written in the order in which it is marked known.

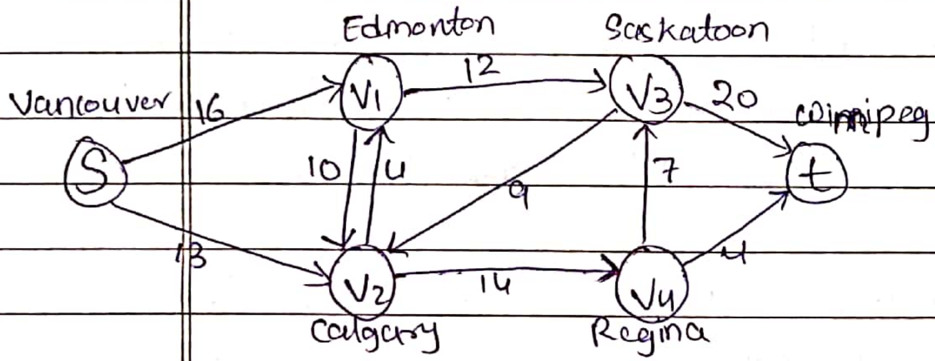
Q.2.

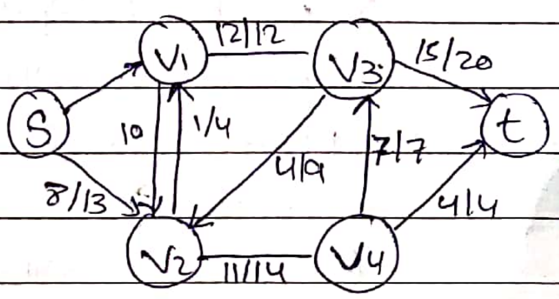
1. According to Dijkstra's Algorithm the computed path to G is A, B, D, F G whereas the shortest path to node G is through path A→C→E→G whose distance is 2.

2. According to Dijkstra's Algorithm the computed path to D is A → B → D whereas the shortest path to node D is throught path A → C → E → G → D with distance 3.

3. According to Dijkstra's Algorithm the computed path to F is A → B → D → F whereas the shortest path to node F is through path A → C → E → G → D → F with distance 5.

Problem 4





(a)

(b)

A flow network G=(V,E) for the Lucky puck Company's trucking problem. The vancouver factory is the source, and the winnipeg warehouse is the sink t. pucks are shipped through intermediate cities, but only c(u,v) crates per day can go from city u to city v. Each edge is labelled with it's capacity.

A flow f in 'G' with value |f|=19. Only positive flows are shown. If (u,v)>0, edge (u,v) is labelled by f(u,v)/C(u,v) 'slash' represents the separation of flow & capacity. if (u,v)≤0 edge (u,v) is labeled only by its capacity.

Flow network 'G' with source 's' & sink 't', and we wish to find a flow of maximum value.

1. The capacity constraint simply says that the flow from one vertex to another must not exceed the given capacity.

2. Skew symmetry is a notational convenience that says that the flow from a vertex u to a vertex

v is the negative flow in the reverse direction.

3. The flow conservation property says that the total flow out of a vertex other then the source of sink is 0.

By skew symmentry, we can rewrite the flow conservation property as.

$$\sum_{u \in V} f(u,v) = 0$$

$\forall u \in V - \{s,t\}$ i.e. total flow in a vertex is '0'.

When neither (u,v) nor (v,u) in E, there can be no flow between u and v and $f(u,v) = f(v,u) = 0$

Formal Proof:

Consider a graph $G = (V,E)$ with source $= S$ and sink $= t$ and capacity function $= C$

We know a flow network satisfies the capacity Constraint:

$$0 \leq f(u,v) \leq C(u,v)$$

where f is a flow in G and $u,v \in V$.

Now, if f is a flow in G, then the residual network Gf only has edges that can admit more flow. The residual capacity $C_f(u,v)$ for an edge (u,v) is defined as.

$$C_f(u,v) = \begin{cases} c(u,v) - f(u,v) & \text{if } (u,v) \in E. \\ f(v,u) & \text{if } (v,u) \in E. \\ 0 & \text{otherwise.} \end{cases}$$

Now, if f is the maximum flow in G, then a residual network Gf must not exist.

This means that there is at least one edge (u, v) in G where f(u,v) = c(u,v) i.e the flow through the edge is equal to its total capacity. It is only if we have such an edge that s and t would get disconnected. Otherwise a residual network would be present from s to t and the flow f would not be maximum.

Therefore, if $f(u,v) = c(u,v)$
We have,
$C_f(u,v) = 0$ i.e there cannot be any additional flow from u to v in any cases which means: $f(u,v) = 0$, which is the required proof.