CSE 551 – Assignment 4

Submitted By:
Ankit Sharma
1219472813
ashar263@asu.edu

Solution for Question (1)

- <u>Does the minimum spanning tree change?</u>

No, Minimum spanning tree (MST) will not change even if each edge weight in a graph is increased by 1.

Proof:
Assume graph G has 'n' vertices.

Then, all spanning trees of G has $= n - 1$ edges.

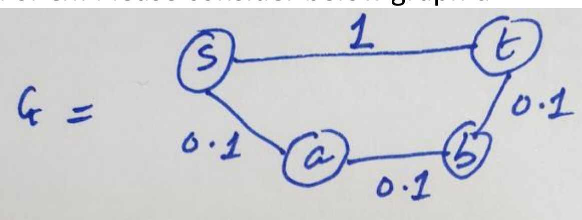
Since weight of each edge is increased by 1 as provided in question. Therefore, weight of all spanning trees will be increased by $= (n-1) * 1 = n - 1$

Since weight of all spanning trees is increased by same value, then **MST in previous graph G will also be MST in new graph**.

- <u>Does the shortest paths change?</u>

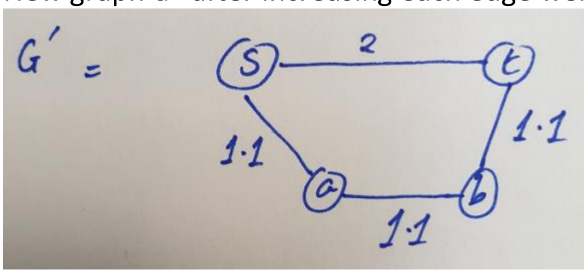
Yes, the shortest paths change even if each edge weight in a graph is increased by 1.

For ex: Please consider below graph G –



Shortest path from s to t is:
$s \to a \to b \to t$ and cost, $C = 0.1 + 0.1 + 0.1 = 0.3$

New graph G' after increasing each edge weight by 1



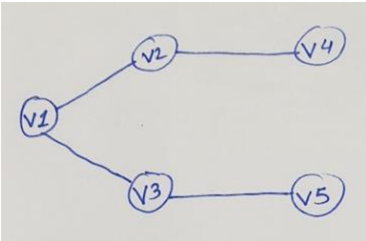
Shortest path now from s to t will be:
$s \to t$ and cost, $C' = 2$
As $s \to a \to b \to t$ has $cost = 1.1 + 1.1 + 1.1 = 3.3$

Hence proved, that **shortest paths between nodes can change**.

Solution for Question (2)

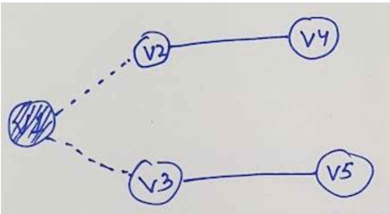
This algorithm will not always return minimum node cover set.

Proof: Let's take following graph G as example –

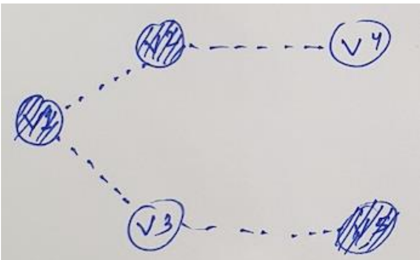


According to the given greedy COVER algorithm, every time we select a vertex of maximum degree.

Suppose $v1$ is selected first amongst $v1, v2$ and $v3$. Then the resultant graph after removing the vertex from set V and its corresponding edges from set E will look like –

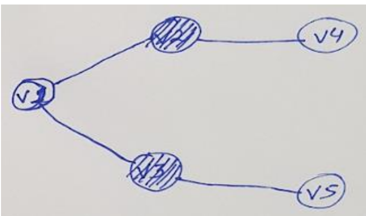


In next two iterations, following greedy approach, if $v2$ and then $v5$ are selected. The graph would look like –



Finally, $U = \{v1, v2, v5\}$
Then the size of node cover set would be $|U| = 3$

But the minimum node cover for this graph is $U = \{v2, v3\}$ and $|U| = 2$, as shown in below:

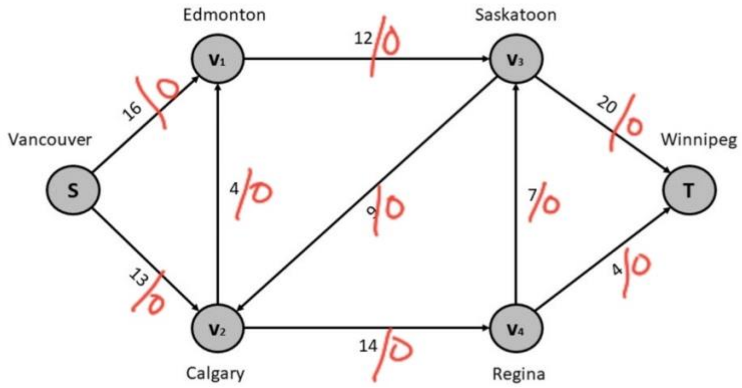


Hence proved that the greedy algorithm **will not always return minimum node cover set**.

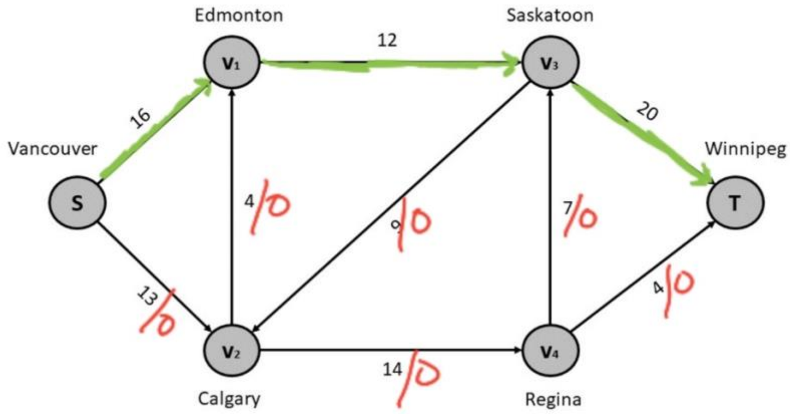
Solution for Question (3)

We can use Ford-Fulkerson algorithm to compute the maximum flow from Vancouver to Winnipeg.

First, we initialize each edge with a flow $f(e) = 0$, where e means all edges in the graph.

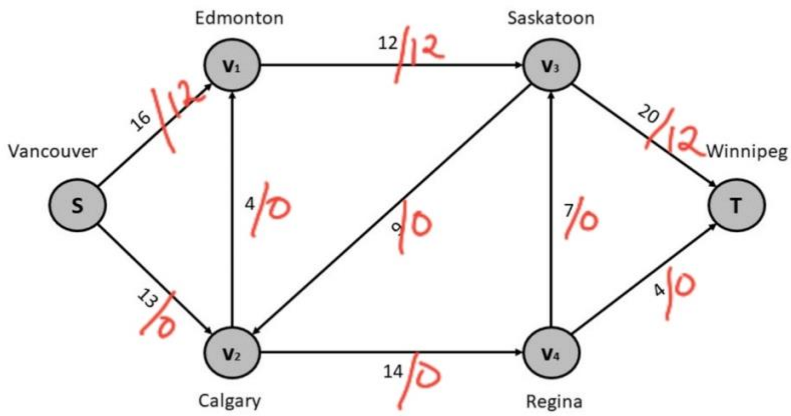


Let, **first augmented path $\boldsymbol{P1}$** $= S \to V_1 \to V_3 \to T$

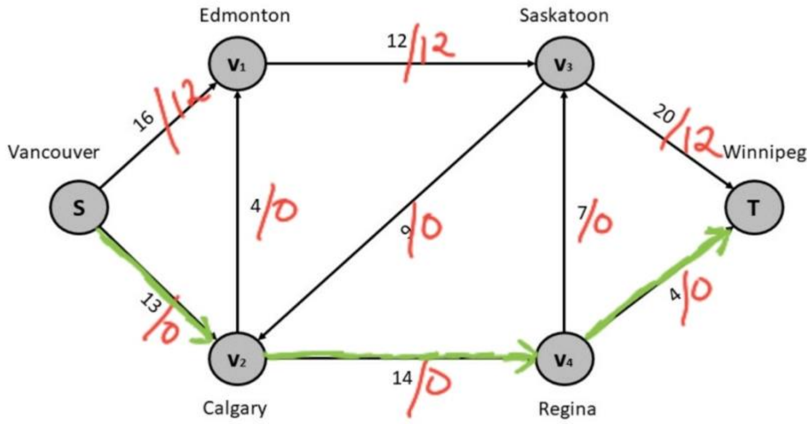


Now, $\Delta = \min(16-0, 12-0, 20-0) = 12$

Since all are forward labels in the augmented path. The augmented flow will be:

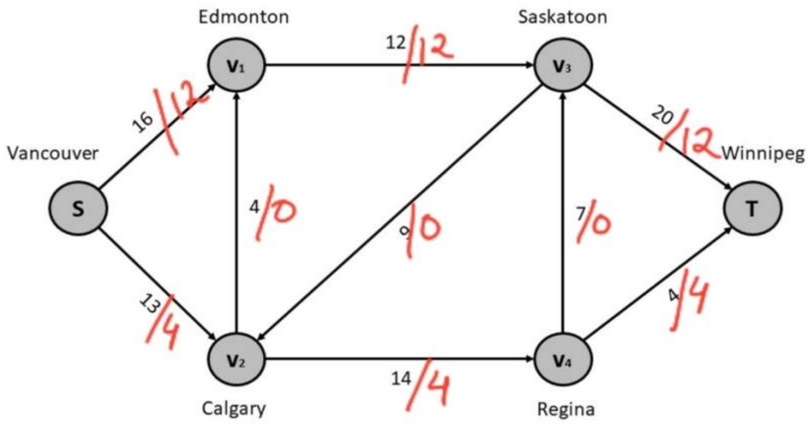


Let, **second augmented path $P2$** $= S \rightarrow V_2 \rightarrow V_4 \rightarrow T$

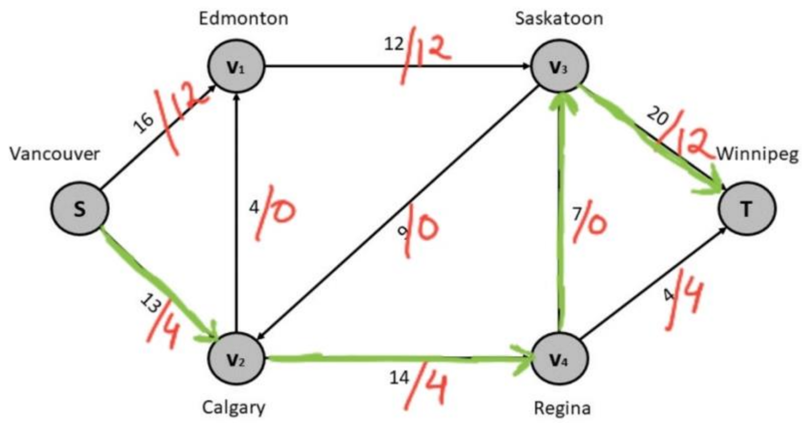


Now, $\Delta = \min(13 - 0, 14 - 0, 4 - 0) = 4$

Since all are forward labels in the augmented path. The augmented flow will be:

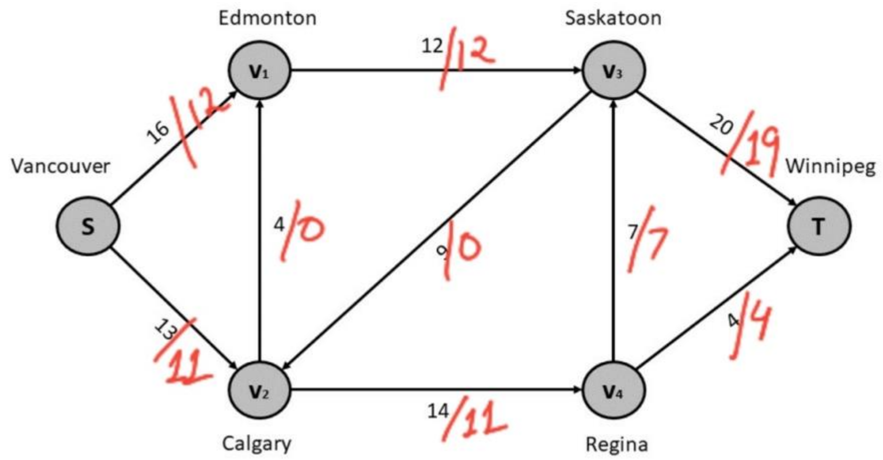


Let, **third augmented path $P3$** $= S \rightarrow V_2 \rightarrow V_4 \rightarrow V_3 \rightarrow T$



Now, $\Delta = \min(13 - 4, 14 - 4, 7 - 0, 20 - 12) = 7$

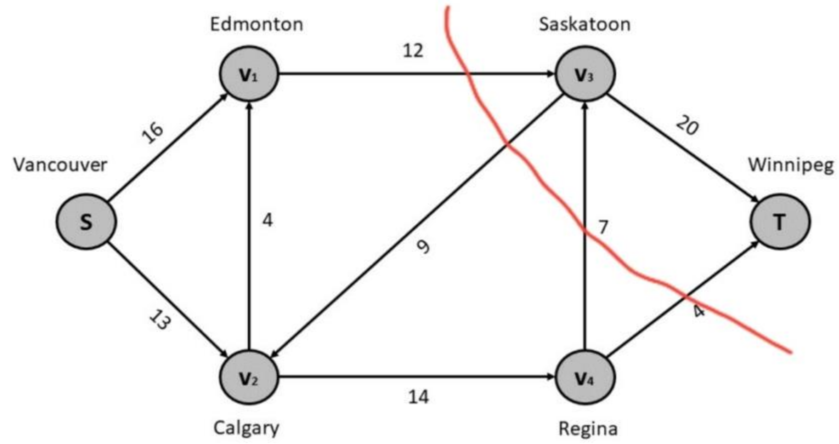
Since all are forward labels in the augmented path. The augmented flow will be:



Now, there are no possible augmented paths. So, the maximum flow from Vancouver to Winnipeg is:

$\boldsymbol{F_{max}} = 19 + 4 = \mathbf{23}$

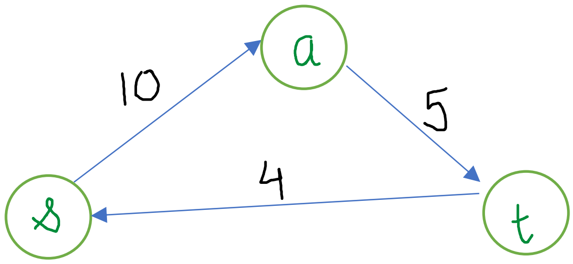
We can also see that this value of $\boldsymbol{F_{max}} = Mincut$



$Mincut\ (S:S') = 12 + 7 + 4 = 23$, where $S = \{s, v_1, v_2, v_4\}$ and $S' = \{v_3, T\}$

Solution for Question (4)

Let's consider a network, given below:



Now, there are two possible cuts:

<u>*Case 1*</u>:

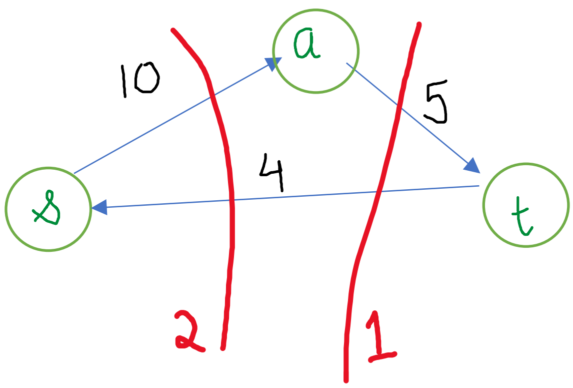
$S = \{s, a\}$ and $S' = \{t\}$

$\sum_{e \in (S:S')} c(e) = 5$ and $\sum_{e \in (S':S\ )} c(e) = 4$

<u>*Case 2*</u>:

$S = \{s\}$ and $S' = \{a, t\}$

$\sum_{e \in (S:S')} c(e) = 10$ and $\sum_{e \in (S':S\ )} c(e) = 4$



In the network above, Max Flow from s to t, $\boldsymbol{F_{max} = 5}$

We need to check if $F_{max}\ = Mincut$ if $C(S:S')$ is defined as-

(a) $\boldsymbol{C(S:S') = \sum_{e \in (S:S'\ )} c(e) + \sum_{e \in (S':S\ )} c(e)}$

Using this formula-

For <u>case 1</u>, capacity of the cut is $=\ 5 + 4 = 9$

For <u>case 2</u>, capacity of the cut is $= 10 + 4 = 14$

Now, $\boldsymbol{Min\ cut} = \min(9{,}14) = 9 \neq \boldsymbol{F_{max}}$

Hence, the **above definition of C(S:S') is FALSE** and cannot be used to calculate Maximum flow.

(b) $\boldsymbol{C(S{:}S') = \sum_{e\in(S:S')} c(e) - \sum_{e\in(S':S)} c(e)}$
Using this formula-
For <u>case 1</u>, capacity of the cut is $= 5 - 4 = 1$
For <u>case 2</u>, capacity of the cut is $= 10 - 4 = 6$

Now, $\boldsymbol{Min\ cut} = \min(1,6) = 1 \neq \boldsymbol{F_{max}}$

Hence, the above definition of C(S:S') is **FALSE** and cannot be used to calculate Maximum flow.

(c) $\boldsymbol{C(S{:}S') = \sum_{e\in(S:S')} c(e)}$
We know that Total Flow from (S:S') is-

As discussed in class,
Total Flow, $F = Flow\ from\ (S{:}S') - Flow\ from\ (S'{:}S)$
=> $F \leq Flow\ from\ (S{:}S')$

We also know that $flow \leq capacity$, So-
=> $F \leq Flow\ from\ (S{:}S') \leq Capacity\ of\ (S{:}S')$
=> $F \leq C(S{:}S')$

Now $C(S{:}S') = \sum_{e\in(S:S')} c(e)$, So-
=> $F \leq \sum_{e\in(S:S')} c(e)$

From the above relation we can see that value of F cannot be higher than minimum value of $\sum_{e\in(S:S')} c(e)$. Therefore-
=> $F \leq \min\ (\sum_{e\in(S:S')} c(e))$
=> $F \leq Min\ cut$
=> $F_{max} = Min\ cut$

Hence, the **above definition of C(S:S') is TRUE** and can be used to calculate Maximum flow.
We can also check validity of definition in the above network.
For case 1, capacity of the cut is $= 5$
For case 2, capacity of the cut is $= 10$
Capacity of $\boldsymbol{Min\ cut}$ = $\min(5,10) = 5 = \boldsymbol{F_{max}}$

(d) $\boldsymbol{C(S{:}S') = \min\{\sum_{e\in(S:S')} c(e),\ \sum_{e\in(S':S)} c(e)\}}$
Using this formula-
For case 1, capacity of the cut is $= \min\{5,4\} = 4 \neq \boldsymbol{F_{max}}$
For case 2, capacity of the cut is $= \min\{10,4\} = 4 \neq \boldsymbol{F_{max}}$

Now, $\boldsymbol{Mincut} = \min(9,14) = 9 \neq \boldsymbol{F_{max}}$

Hence, the above **definition of C(S:S') is FALSE** and cannot be used to calculate Maximum flow.