Dhanraj Bhosale (dbhosal1@asu.edu)
ASU ID: 1225506620

Q.(1)
[I]

Yes. The strategy to sort the intervals in decreasing order of their start times, will ENSURE finding the optimal solution.

Suppose we have a set S = {a_1, a_2, a_3, …, a_n} of n proposed activities.
Each activity i has a start time s_i and a finish time f_i, where $s_i \leq f_i$.
Assume that the activities are ordered by decreasing order of their start time

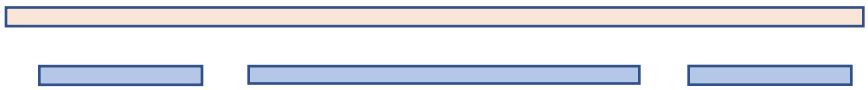
$s_1 \geq s_2 \geq \ldots \geq s_n$.

How do we know that this strategy works?
Suppose {a_5, a_7, a_8, a_{12}, a_{15}} is an optimal solution. Consider the following activity set {a_1, a_7, a_8, a_{12}, a_{15}}. Is this set an optimal solution?
Yes

- Why?
- Since a_5 and a_7 are part of an optimal solution, the end time of a_7 must be before the start time of a_5. Since the start time of a_1 is later than that of the start time of a_5, there will be no conflict between a_1 and a_7. Therefore, the activity set {a_1, a_7, a_8, a_{12}, a_{15}} will also be an optimal solution.

[II]

No. The strategy to sort the intervals in decreasing order of their lengths, will NOT ENSURE finding the optimal solution.
Consider following example:



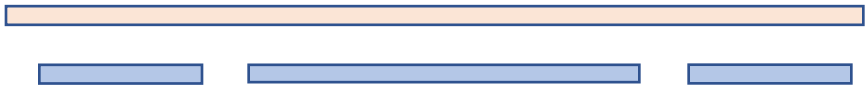
Algorithm will pick **1** Orange job, but we could have picked **3** Blue jobs.

[III]

No. The strategy to sort the intervals in increasing order of the reciprocal values of their lengths, will NOT ENSURE finding the optimal solution.

Note: As we know, as decreasing order of numbers is same as increasing order of their reciprocals.
Example: a>b then 1/a < 1/b

So again, it is same as sort the intervals in decreasing order of their lengths.
Consider following example:



Algorithm will pick **1** Orange job, but we could have picked **3** Blue jobs.

Q.(2)
[I]

Algorithm-

Dhanraj Bhosale (dbhosal1@asu.edu)
ASU ID: 1225506620

Sort jobs in increasing order of their run time and consider them in this order in the algorithm. Return this order

Proof of correctness-
Average waiting time of jobs will be minimized if the total time run by the jobs in the system is minimized.
Let $I = (i_1\ i_2 \dots i_n)$ be any permutation of the integers $\{1, 2, \dots, n\}$. If the jobs are ran in the order I, the total time passed in the system by all jobs is

$$\begin{aligned} T(I) &= t_{i1} + (t_{i1} + t_{i2}) + (t_{i1} + t_{i2} + t_{i3}) + \dots + (t_{i1} + \dots + t_{in}) \\ &= nt_{i1} + (n-1)t_{i2} + (n-2)t_{i3} + \dots + t_{in} \\ &= \Sigma_{k=1 \text{ to } n} (n-k+1)t_{ik} \end{aligned}$$

Suppose now that, I is such that we can find two integers a and b with $a<b$ and $t_{ia} > t_{ib}$
New run order I' is obtained from I by interchanging the run orders of a and b

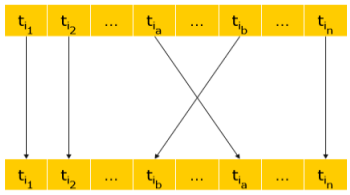
$T(I') = (n-a+1)t_{ib} + (n-b+1)t_{ib} + \Sigma_{k=1 \text{ to } n,\ k\neq a,b}(n-k+1)t_{ik}$

run order	1	2	…	a	…	b	…	n
ran jobs (from I)	i_1	i_2		i_a		i_b		i_n

run duration

After exchange of i_a and i_b

run duration



ran jobs (from I')	i_1	i_2		i_b		i_a		i_n

Thus,

$$\begin{aligned} T(I) - T(I') &= (n-a+1)(t_{ia} - t_{ib}) + (n-b+1)(t_{ib} - t_{ia}) \\ &= (b-a)(t_{ia} - t_{ib}) \\ &> 0 \end{aligned}$$

We can therefore improve any schedule in which a job is run before any other job which requires less run time. The **only schedules that remain are those putting the jobs in non-decreasing order of run time**. ---------- **(a)**
All such schedules are clearly equivalent, and therefore they are all optimal.

[II]

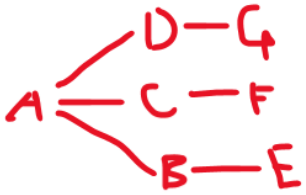
No. There can't be multiple schedules which give minimum average waiting time if the time taken for each of the job is distinct.

Explanation-
As we proved above, from statement (a), **correct order is non-decreasing order** of run time:
If all job will have distinct run time, there exists **ONLY ONE non-decreasing order**.

Dhanraj Bhosale (dbhosal1@asu.edu)
ASU ID: 1225506620

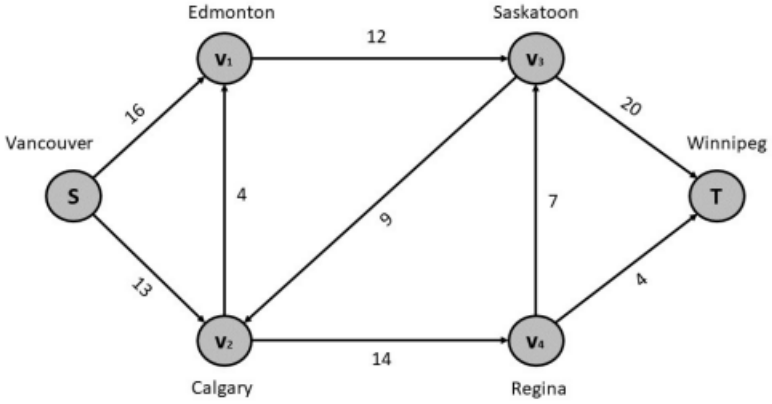
Q.(3)

This algorithm will NOT always generate a minimum node cover.
Let's take following example:



A tree with A as a root node. It has B, C, D as child nodes. B has E, C has F and D has G as child node.
The algorithm above will recognize that node A is the vertex of greatest degree (= 3) and add it to the arrangement set. Then, note that every one of the excess nodes have greatest degree = 1 and there are three edges left in the tree. Thusly, three hubs should be added to the arrangement set to cover every one of the node in the first tree. Consequently, the cardinality of the arrangement set following this calculation will be 4. Nonetheless, the ideal cardinality is 3, when we simply select nodes (B, C, D). Subsequently, this calculation won't necessarily deliver a minimum node cover.

Q. (4)

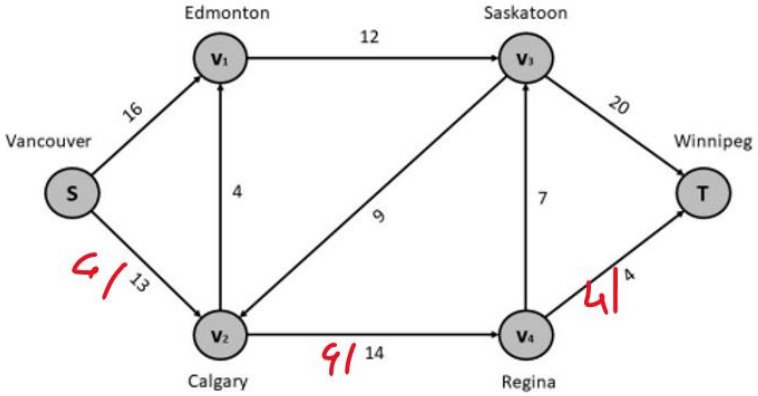


Initially all have 0 used capacity
For forward labeling Delta = Capacity- flow
For backward labeling Delta = flow

Lets consider following augmented paths:

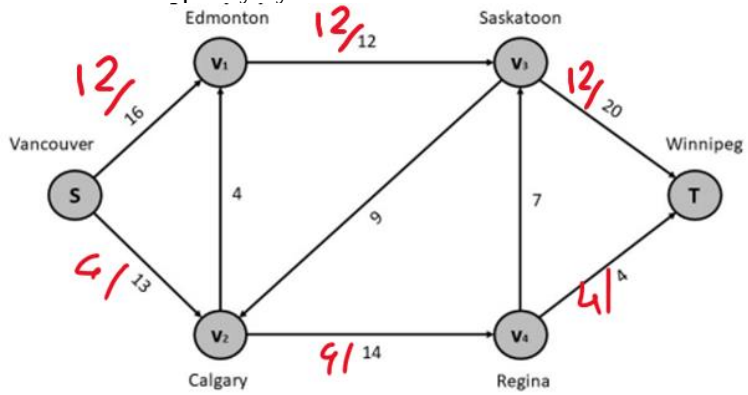
AP1: S->v2->v4->T

Here Delta is 13,14,4
V4->T decides throttling.
So flow for all will increase by 4

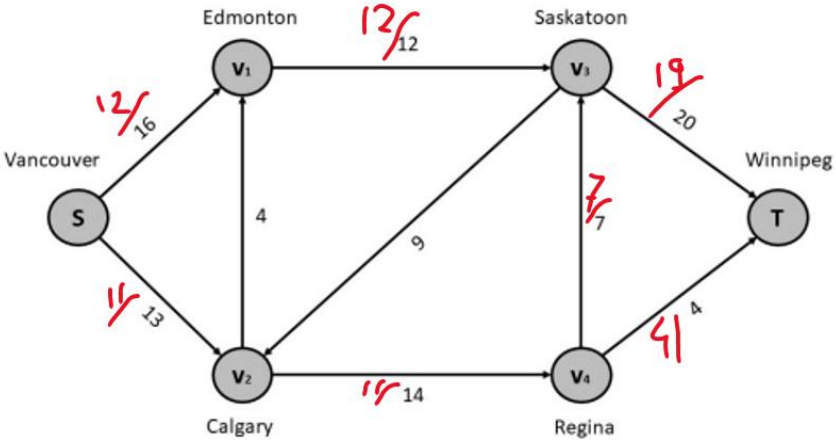


Dhanraj Bhosale (dbhosal1@asu.edu)
ASU ID: 1225506620

AP2: S->v1->v3->T
Delta is 16, 12, 20
V1->v3 decides throttling.
Increase used capacity by 12



AP3: S->v2->v4->v3->T
Delta is 9, 10, 7, 8
V4->v3 decides throttling.
Increase used capacity by 7



No more augmented path cam be found. So we end our search.

So **maximum flow will be 19+4 = 23.**
The minimum cut C(S:S¯) is across S={S,v1,v2,v4} and S¯= {v3,T}.