DAA Mid Sem Exam Solution

1: $\frac{p_1}{w_1} = \frac{20}{2} = 10$, $\frac{p_2}{w_2} = \frac{10}{3}$, $\frac{p_3}{w_3} = \frac{30}{5} = 6$,

$\frac{p_4}{w_4} = \frac{14}{7} = 2$, $\frac{p_5}{w_5} = \frac{12}{1} = 12$, $\frac{p_6}{w_6} = \frac{36}{4} = 9$, $\frac{p_7}{w_7} = \frac{6}{1} = 6$,

After Sorting $\frac{p_i}{w_i}$, we get:

$\frac{p_5}{w_5} = 12$, $\frac{p_1}{w_1} = 10$, $\frac{p_6}{w_6} = 9$, $\frac{p_3}{w_3} = 6$, $\frac{p_7}{w_7} = 6$,

$\frac{p_2}{w_2} = \frac{10}{3}$, $\frac{p_4}{w_4} = 2$ (3)

Taking the items in order of $\frac{p_i}{w_i}$: (1)

$w_5 = 1$, $w_1 = 2$, $w_6 = 4$, $w_3 = 5$, $w_7 = 1$, $w_2 = 3$, ...

$p_5 = 12$, $p_1 = 20$, $p_6 = 36$, $p_3 = 30$, $p_7 = 6$, $p_2 = 10$, ...

$\{5, 1, 6, 3, 7\}$ gives $\sum w_i = 13$ and $\sum p_i = 104$.

For the 2nd item, we have to take a fraction $\frac{2}{3}$ so that total weight becomes $3 \times \frac{2}{3} + 13 = 15$.

The corresponding profit is given by: $104 + \frac{2}{3} \times 10$

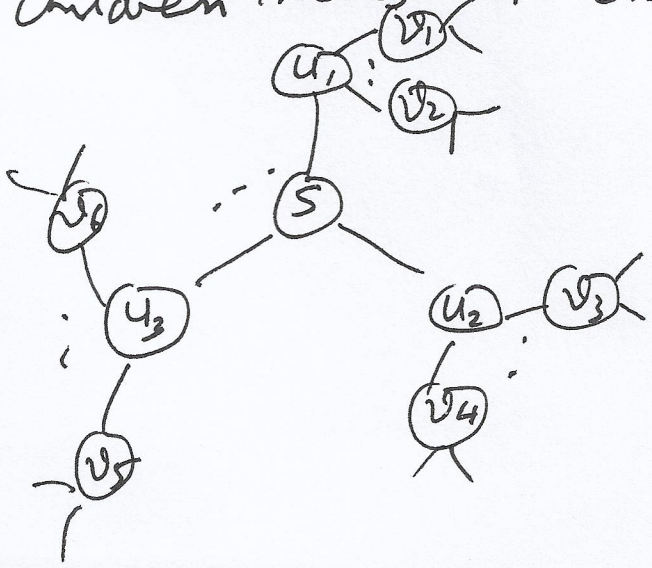
$= 110\frac{2}{3} = 110.66$. (3)

The items selected are: 5, 1, 6, 3, 7 and $\frac{2}{3}$ of 2. (3)

2: We will make use of the Breadth First Search (BFS) algorithm on the given tree $T = (V, E)$ to convert it into a rooted tree with parent-child relationships in $O(|V| + |E|)$ time. Let r be the root node of the original tree T (it is the node from which we start the BFS algorithm). Let $VC(r)$ be the original ~~size~~ size of the minimum vertex cover of the tree T with root r.

Subproblems: For any node $s \in V$, let $VC(s)$ be the size of the minimum vertex cover of the subtree rooted at s.

Optimal Substructure: Consider the subtree rooted at s with children nodes u_i's and grandchildren nodes v_j's:



We have 2 possibilities for s:

Case 1 : S is in the minimum vertex cover.

S covers all edges to the children nodes.

We should optimally solve the subproblems $VC(u_i)$.

In this case we have the size of VC given as

$$VC_1 = 1 + \sum_{\substack{u \text{ is child} \\ \text{of } S}} VC(u) \longrightarrow (1)$$

Case 2 : S is not in the minimum vertex cover.

To cover the edges from S to the children nodes, we should include all the children in the vertex cover. This will also cover all the edges from children to grandchildren nodes. We should optimally solve the subproblems $VC(v_i)$.

In this case, the size of the VC is given by:

$$VC_2 = \sum_{\substack{u \text{ is child} \\ \text{of } S}} \left(1 + \sum_{\substack{v \text{ is child} \\ \text{of } u}} VC(v)\right) \longrightarrow (2)$$

We should choose minimum of (1) and (2):

$$VC(S) = \min\left\{ 1 + \sum_{\substack{u \text{ is child} \\ \text{of } S}} VC(u), \sum_{\substack{u \text{ is child} \\ \text{of } S}} \left(1 + \sum_{\substack{v \text{ is child} \\ \text{of } u}} VC(v)\right) \right\} \quad (6)$$

Algorithm TVC (T, r)

```
1  for each vertex s ∈ T.V − {r}
2      s.color ← white
3      s.parent ← NIL, VC(s) ← 0
4  r.color ← gray
5  r.parent ← NIL, VC(r) ← 0
6  Queue Q ← ϕ
7  Stack P ← ϕ
8  Enqueue(Q, r)
9  while (Q ≠ ϕ)
10     s ← Dequeue(Q)
11     for each u ∈ T.Adj[s]
12         if u.color = white
13             u.color ← gray
14             u.parent ← s
15             Enqueue(Q, u)
16     s.color ← black
17     Push(P, s)
```

```
18 while (P ≠ ϕ)
19     s ← Pop(P), sum1 ← 1, sum2 ← 0
20     for each u ∈ T.Adj[s]
21         if u.parent = s
               sum1 ← sum1 + VC(u)
22
               sum2 ← sum2 + 1
23
               for each v ∈ T.Adj(u)
24                 if v.parent = u
25
                       sum2 ← sum2 + VC(v)
26
27     if sum1 ≤ sum2
28         s.cover ← 1
29         VC(s) ← sum1
30     else
           s.cover ← 0
31         VC(s) ← sum2
32         for each u ∈ T.Adj[s]
33             if u.parent = s
34                 u.cover ← 1
35
```

```
36  output UC(r)
37  for each s ∈ T.V
38      if s.cover = 1
39          output s
```

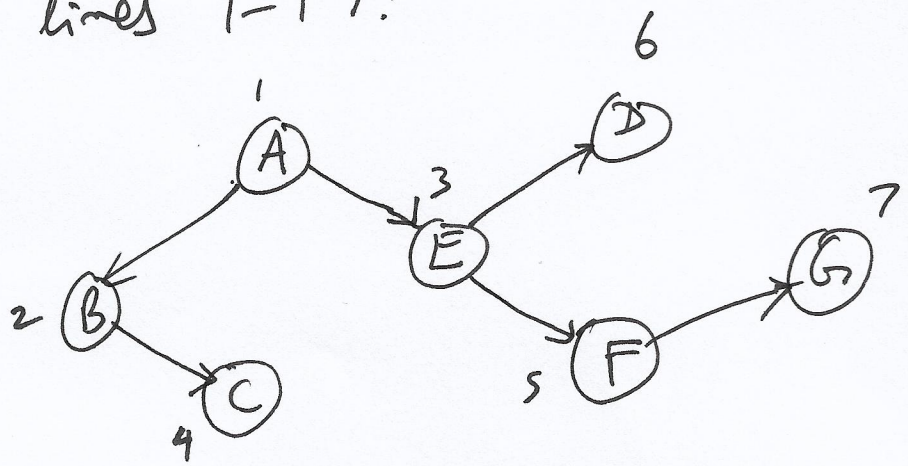
(8)

Complexity of TVC: Lines 1-17 are simply the BFS algorithm having complexity $O(|V|+|E|)$. The while loop of lines 18-35, pops the vertices $|V|$ times. For each vertex, its adjacent vertices are visited two times: line 20 as a child; and line 24 as a grandchild. The total contribution will be $2|E|$. Lines 36-39 have complexity $O(|V|)$.

Total complexity we get is $O(|V|+|E|) = O(|V|)$ which is linear in the input size (adjacency list representation of T). (2)

(For a tree $|V| = |E| + 1$).

Taking $r = A$, we get the following tree after lines 1-17: (7)



7 G: G.cover = 0, VC(G) = 0

6 D: D.cover = 0, VC(D) = 0

5 F: F.cover = 1, VC(F) = 1

4 C: C.cover = 0, VC(C) = 0

3 E: E.cover = 1, VC(E) = 2

2 B: B.cover = 1, VC(B) = 1

1 A: A.cover = 0, VC(A) = 3

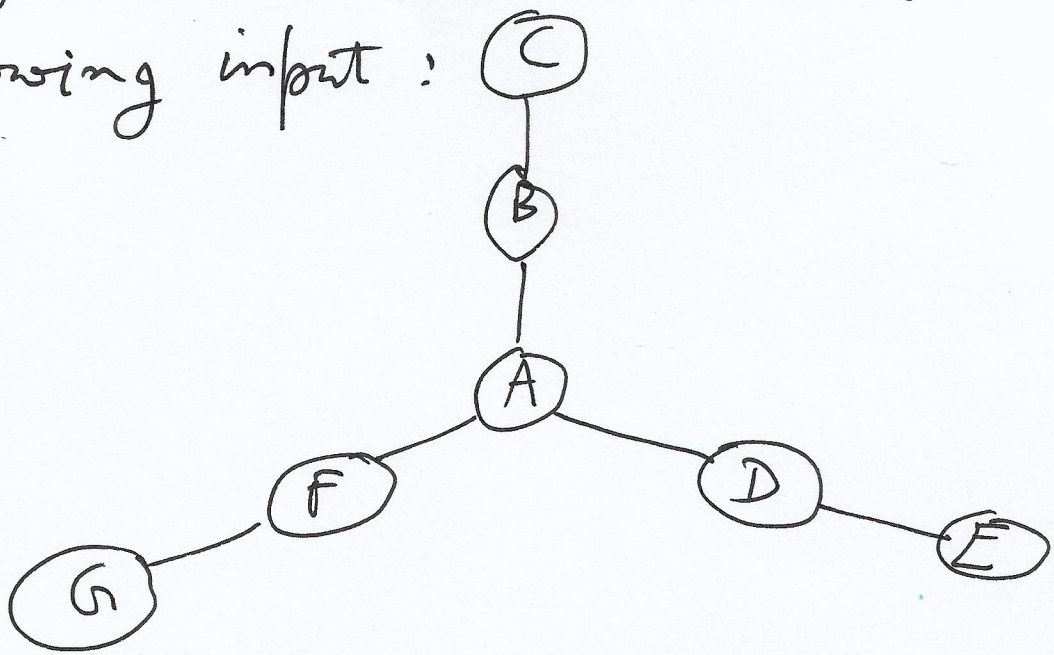
The minimum vertex cover is $\{B, E, F\}$ of size 3.

(4)

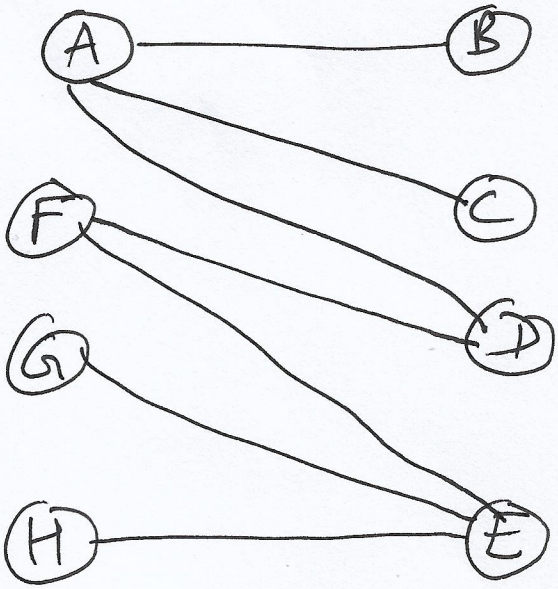
Incorrect Approaches for Q2 :

(1) Take any node which is adjacent to a leaf node in the vertex cover and remove the edges incident on it. Repeat the same procedure again untill no edges remain. This is a Greedy Algorithm with $O(|V|)$ time complexity (Not a DP Algorithm).

(2) Take the node with highest degree in the vertex cover and remove the edges incident on it. Repeat the same procedure again until no edges remain. This is an incorrect Greedy Algorithm. The algorithm will fail to give the minimum vertex cover on the following input :



(3) use the BFS algorithm to convert the given tree into a Bipartite graph, and perform 2-coloring of the graph. Take the vertices of the same color with minimum number of vertices in the Vertex Cover. This is an incorrect algorithm. It will fail to give the minimum vertex cover on the following input:



(4) Let $DP[v][0]$ be the size of minimum vertex cover in which v is not selected (→ in the vertex cover). Let $DP[v][1]$ be the size of minimum vertex cover in which v is selected in the vertex cover.

$$DP[v][0] = \sum_{u \text{ is child of } v} DP[u][1]$$

$$DP[v][1] = \sum_{u \text{ is child of } v} DP[u][0]$$

This is equivalent to the algorithm (3) above which is incorrect.

3/ Decision version of the Load Balancing Problem:

$LBP = \{ (t_i)_{i=1}^{n}, m, B \mid \exists$ an assignment of tasks to the m machines such that the makespan is upper bounded by $B \}$ [4]

To prove that $LBP \in NP\text{-C}$, we have to prove that $LBP \in NP$ and $LBP \in NP\text{-H}$.

$LBP \in NP$: We design an NTM N such that on input $((t_i)_{i=1}^{n}, m, B)$, N will guess an assignment of the n tasks on the m machines in polynomial time, using its non-deterministic moves. Then it will find the makespan, again in polynomial time. If makespan $\leq B$, then accept the input, otherwise, reject the input.

$\Rightarrow LBP \in NP$. [4]

$LBP \in NP\text{-H}$: We will reduce the Subset Sum Problem (SSP) to LBP in polynomial time.

We know that $SSP \in NP\text{-C}$. $SSP \leq_p LBP$ will prove that $LBP \in NP\text{-H}$.

$$SSP = \left\{ \{a_i\}_{i=1}^{n}, S \;\middle|\; \exists \text{ a subset } A \subseteq \{a_i\}_{i=1}^{n} \text{ such that } \sum_{a_j \in A} a_j = S \right\}$$

Consider the following Polynomial-Time Reduction ($SSP \leq_p \angle BP$): (11)

$R(\{a_i\}_{i=1}^{n}, S)$

{ $S_0 \leftarrow \sum_{i=1}^{n} a_i$

$S_1 \leftarrow S_0 - S$

$B \leftarrow \max\{S_1, S\}$

$a_{n+1} \leftarrow |S_1 - S|$

if ($a_{n+1} = 0$)

{ return ($\{a_i\}_{i=1}^{n}, 2, B$) }

else

{ return ($\{a_i\}_{i=1}^{n+1}, 2, B$) } (4)

}

$(\{a_i\}_{i=1}^{n}, S) \in SSP \Rightarrow \exists A \subseteq \{a_i\}_{i=1}^{n}$ such that $\sum_{a_j \in A} a_j = S$. Let $C = \{a_i\}_{i=1}^{n} - A$

$\sum_{a_j \in C} a_j = S_0 - S = S_1$

if $S_1 < S$, then put a_{n+1} in C otherwise put a_{n+1} in A (if it exists). Then both A and C will have elements with total sum B. Allocate all tasks in A to M_1, and allocate all tasks in C to M_2. $\Rightarrow$ makespan = B $\Rightarrow$

$(a_i)_{i=1,2}^{n+1}, B) \in LBP$. [4]

To prove

$(\{a_i\}_{i=1}^{n}, S) \notin SSP \Rightarrow ((a_i)_{i=1}^{n+1}, 2, B) \notin LBP$

assume that $(\{a_i\}_{i=1}^{n}, S) \notin SSP$ and

$((a_i)_{i=1}^{n+1}, 2, B) \in LBP$

$\Rightarrow \exists$ assignment of tasks on M_1 and M_2 such that

$$\sum_{a_i \in M_1} a_i \leq B \text{ and } \sum_{a_j \in M_2} a_j \leq B$$

Let $a_{n+1} \in M_1$ (if it exists), then either $M_1 - \{a_{n+1}\}$ or M_2 will have elements with sum = S $\Rightarrow$ $(\{a_i\}_{i=1}^{n}, S) \in SSP$ (a contradiction).

$\Rightarrow$ $((a_i)_{i=1}^{n+1}, 2, B) \notin LBP$. [4]