DATA STRUCTURES AND ALGORITHMS DESIGN

SE ZG519/SS ZG519

EC-1: Assignment Weightage: 20% Marks: 40

Submission Dead Line: 04th May 2025 (Sunday)12:00 Noon

Mode of Submission: Write down (preferably typed in a doc file) your answer in the space provided and submit a scan copy of your document.

Name: MALLIDI AKHIL REDDY **ID:** 2024TM93056

A company wants to build a chain of restaurants on many street corners with the goal of maximizing their total profit. The street network is described as an undirected graph G = (V, E), where the potential restaurant sites are the vertices of the graph. Each vertex u has a nonnegative integer value p_u indicating the potential profit of site u. Two restaurants cannot be built on adjacent vertices (to avoid self-competition). You are supposed to design an algorithm that outputs the chosen subset $U \subseteq V$ of sites that maximizes the total profit $\sum_{u \in U} p_u$.

(a) [Marks: 5] Suppose that the street network G is acyclic, i.e., a tree. Consider the following "greedy" restaurant-placement algorithm: Choose the highest- profit vertex u_0 in the tree (breaking ties according to some order on vertex names) and put it into U. Remove u_0 from further considerations, along with all of its neighbors in G. Repeat until no further vertices remain. Produce two example graphs, i.e., trees, where for one graph the algorithm produces maximum profit and for the other graph the algorithm does not produce the maximum profit. Consider having exactly five nodes in each of the graphs.

Let G be a city graph such that G = (V, E)

V be Vertices (City corners)

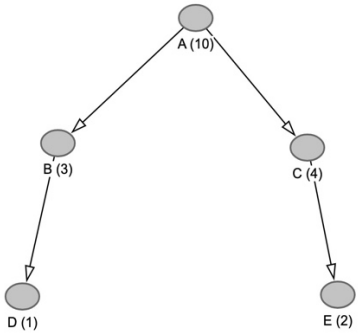
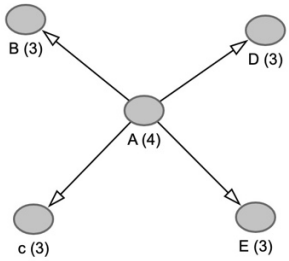
E be Edges (Streets)

As per greedy approach, the steps are:

1) Pick the highest profit node

2) Remove that node and all its neigbors (as the restaurants should be on adjacent corners)

3) Repeat steps 1 & 2

Graph with maximum profit using Greedy	**Graph which doesn't give maximum profit using Greedy**
	
Pick the highest node A having profit 10. Removing that node and its adjacent nodes (B & C), as they are neighbors. Nodes left are D and E. Pick E, as its has highest profit among two withr profit 2. As there are no adjacent nodes, pick D which has profit 1. Picked Nodes – A, E, D Total profit – 10 + 2 + 1 = 13 No other combination gives profit more than 13, Hence Greedy works perfectly here.	As per Greedy, Picking the highest node A with profit 4. Removing A along with all adjacent nodes (B, E, D, C), results in no nodes left. Picked Nodes – A Total profit – 4 But alternative path, by selecting B, C, D, E as they are not neighbors yields total profit 12. So here, Greedy approach does not yield Maximum profit.

(b) [Marks: 10] Suppose that the street network G is acyclic. Give an efficient algorithm to determine a placement with maximum profit. Note: your algorithm should be written in plain English and in sequence of steps.

Algorithm: Maximum Profit Restaurant Placement in a Tree

Step 1: Pick any node as the root of the tree.

Step 2: For each node 'u' in the tree, we will calculate two values:

- **MaxProfitWithNode(u):** The maximum profit that can be achieved in a given subtree rooted at 'u' including the profit of node 'u'.
- **MaxProfitWithoutNode(u):** The maximum profit that can be achieved in a given subtree rooted at 'u' excluding the profit of node 'u'.

Step 3: If 'u' is a leaf node

- **MaxProfitWithNode(u):** profit(u)
- **MaxProfitWithoutNode(u):** 0

Step 4: If 'u' is an internal node (a node with children), let 'v1', 'v2', ..., 'vk' be its children

- **MaxProfitWithNode(u):** profit(u) + MaxProfitWithNode(v1) + MaxProfitWithNode(v2) + …….. + MaxProfitWithNode(vk)
- **MaxProfitWithoutNode(u):** max(MaxProfitWithNode (v1), MaxProfitWithoutNode(v1)) + max(MaxProfitWithNode (v2), MaxProfitWithoutNode (v2)) + ... + max(MaxProfitWithNode (vk), MaxProfitWithoutNode (vk))

Step 5: The maximum profit for the entire tree is the larger of the two values calculated for the root node: max(MaxProfitWithNode (root), MaxProfitWithoutNode (root)).

(c) [Marks: 5+5] Apply your algorithm designed in b) on the graphs in part a) and show the results. Your answer is expected in an appropriate graphical representation of the graph.

Based on the above graph, applying the algorithm

Selecting node A as root.

As D and E are leaf nodes, calculating the max profit for them.

- MaxProfitWithNode(D) = 1 (profit of D)
- MaxProfitWithoutNode(D) = 0
- MaxProfitWithNode(E) = 2 (profit of E)
- MaxProfitWithoutNode(E) = 0

As B and C are internal nodes, calculating the max profit for them.

For Node B:

- MaxProfitWithNode(B) = profit(B) + MaxProfitWithoutNode(D) = 3 + 0 = 3
- MaxProfitWithoutNode(B) = max(MaxProfitWithNode(D), MaxProfitWithoutNode(D)) = max(1, 0) = 1

For Node C:

- MaxProfitWithNode(C) = profit(C) + MaxProfitWithoutNode(E) = 4 + 0 = 4
- MaxProfitWithoutNode(C) = max(MaxProfitWithNode(E), MaxProfitWithoutNode(E)) = max(2, 0) = 2

Based on the above graph, applying the algorithm

Selecting node A as root.

As B, C, D and E are leaf nodes, calculating the max profit for them.

- MaxProfitWithNode(B) = 3 (profit of B)
- MaxProfitWithoutNode(B) = 0
- MaxProfitWithNode(C) = 3 (profit of C)
- MaxProfitWithoutNode(C) = 0
- MaxProfitWithNode(D) = 3 (profit of D)
- MaxProfitWithoutNode(D) = 0
- MaxProfitWithNode(E) = 3 (profit of E)
- MaxProfitWithoutNode(E) = 0

For Node A which is root, the child nodes are B, C, D & E.

- MaxProfitWithNode(A) = profit(A) + sum of MaxProfitWithoutNode(child)

  = 4 + (0 + 0 + 0 + 0)

  = 4

- MaxProfitWithoutNode(A) = sum of max(MaxProfitWithNode(child),

For Node A which is root, the child nodes are B & C.

- MaxProfitWithNode(A) = profit(A) + MaxProfitWithoutNode(B) + MaxProfitWithoutNode(C)

  = 10 + 1 + 2 = 13

- MaxProfitWithoutNode(A) = max(MaxProfitWithNode(B), MaxProfitWithoutNode(B)) + max(MaxProfitWithNode(C), MaxProfitWithoutNode(C))

  = max(3,1) + max(4,2)

  = 3 + 4 = 7

Maximum Profit =

max(MaxProfitWithNode(A), MaxProfitWithoutNode(A))

= max(13, 7)

= 13

Selected Nodes: A, E, D

MaxProfitWithoutNode(child)) over all children

= (max(3,0) + max(3,0) + max(3,0) + max(3,0))

= (3 + 3 + 3 + 3)

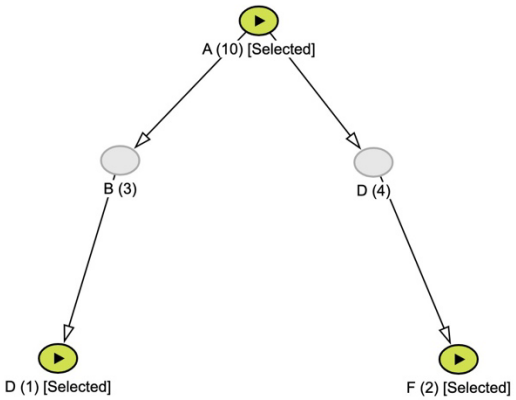
= 12

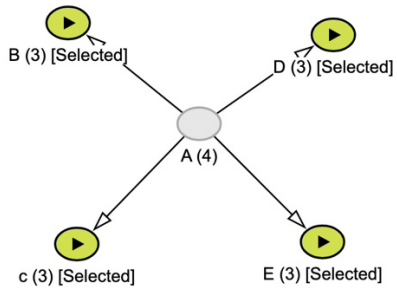
Maximum profit = max(MaxProfitWithNode(A), MaxProfitWithoutNode(A))

= max(4, 12)

= **12**

Selected Nodes: B, C, D, E





(d) [Marks: 5 points] Suppose that the street network G is acyclic. In the absence of a good market research, the company decides that all sites are equally good, and the goal is simply to design a restaurant placement with the largest number of locations. Give a simple greedy algorithm to solve the problem. Algorithm should be written in plain English and in sequence of steps.

Algorithm: Acyclic Graph Greedy Restaurant Placement for Maximum Locations

Step 1: Initialize U = {} Creating an empty set to store the chosen restaurant locations. Mark all nodes in the graph as "unvisited."

Step 2: While there are any "unvisited" nodes remaining in the graph

- Choose any "unvisited" node 'u'
- Add node 'u' to the set of chosen locations: $U = U \cup \{u\}$.
- Mark node 'u' and all of its neighbors as "visited."

Step 3: The set 'U' now contains the maximum number of non-adjacent restaurant locations.

(e) [Marks: 5] Now suppose that the graph is arbitrary, not necessarily acyclic. Give the fastest and correct algorithm you can for solving the problem. What is the time complexity of your algorithm?

Algorithm: Maximum Profit Restaurant Placement (Arbitrary Graph)

// The goal is to find the independent set with the maximum total profit.

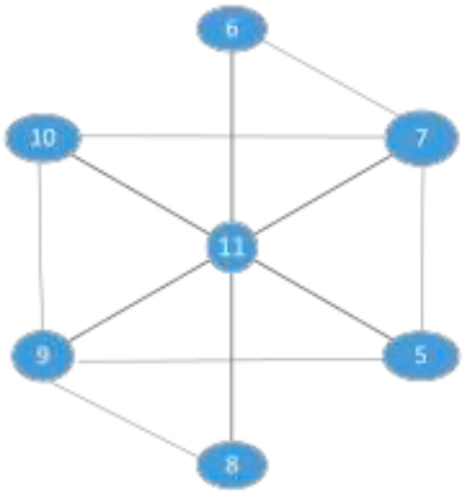
Step 1: Generate all subsets in a given graph.

Step 2: For each subset check whether the given subset is independent or not. For every pair of vertices (u, v) in U, verify that there is no edge between u and v in the graph G.

Step 3: If U is an independent set, calculate the total profit of the vertices in U by summing their individual profits.

Step 4: After considering all subsets, return the independent set U with the highest total profit.

(f) [Marks: 5] Apply algorithm in part e) to find the solution in the following graph. Draw only the final graph keeping the relative positions of the nodes unchanged.



From the graph:

6 is connected to 7 and 11.

10 is connected to 7, 9, and 11.

11 is connected to 5, 6, 7, 8, 9, 10.

9 is connected to 5, 8, 10, 11.

5 is connected to 7, 9, 11.

8 is connected to 9, 11.

7 is connected to 5, 6, 10, 11.

There are 2^7 = 128 subsets of these vertices (including the empty set).

By computing the all possible subsets, The independent set with the maximum total profit has the nodes **5, 6, 8 and 10** with a total profit of **29**.