|  |  |
| --- | --- |
|  | **[Design & Analysis of Algorithm]**  **[BSCS – 5 A]**  **Department of Computer Science**  **Bahria University, Lahore Campus** |

**Assignment: 3**

Name: \_Affan Ahmad \_\_\_ Roll No: \_03-134221-003\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation of CLO** | **Question Number** | **Marks** | **Obtained Marks** |
| **CLO statement**   * **CLO: Demonstrate an understanding of algorithm design process and different problem solving techniques** | 1 | 2.5 |  |
| **Total Marks** | | **2.5** |  |

**Problem Title: Cafe Management [2.5 points]**

**Sweet Donuts, a new Coffee-and-donuts Café chain, wants to build cafes on many street corners of Lahore with the goal of maximizing their total profit. The street network is described as an undirected graph *G = (V, E)*, where the potential cafe sites are the vertices of the graph. Each vertex u has a nonnegative integer value *Pu*, which describes the potential profit of site u. Two cafes cannot be built on adjacent vertices (to avoid self-competition). You are supposed to design an algorithm that outputs the chosen set *U ⊆ V* of sites that maximizes the total profit ⅀u∈U Pu.**

**First, for parts (a)–(c), suppose that the street network G is acyclic, i.e., a tree.**

1. [0.5 points] Consider the following “greedy” cafe-placement algorithm: Choose the highest-profit vertex u0 in the tree (breaking ties according to some order on vertex names) and put it into U. Remove u0 from further consideration, along with all of its neighbors in G. Repeat until no further vertices remain. Give a counterexample to show that this algorithm does not always give a cafe placement with the maximum profit.

Profits: P1=4, P2**=3, P3=2,** P4=5, P5=1, P6=1, P7=1

Using the greedy algorithm:

1. Choose vertex 1 with profit 4.
2. Remove vertex 1 and its neighbors (vertices 2 and 3).

Remaining vertices: 4, 5, 6, 7.

1. Choose vertex 4 with profit 5.
2. Remove vertex 4 and its neighbors (none left).

Total profit: 4+5=94 + 5 = 94+5=9.

Optimal solution:

1. Choose vertices 2, 4, and 6.
2. Total profit: 3+5+1=93 + 5 + 1 = 93+5+1=9.
3. [0.9 points] Give an efficient algorithm to determine a placement with maximum profit.

For each vertex uuu, we define two states:

* include[u]\text{include}[u]include[u]: Maximum profit when vertex u is included.
* exclude[u]\text{exclude}[u]exclude[u]: Maximum profit when vertex u is not included.

**Algorithm:**

1. Perform a depth-first search (DFS) to traverse the tree.
2. For each vertex u:
   * If u is included, then its children cannot be included.
   * If u is excluded, then its children can be either included or excluded.

The recurrence relations are:

include[u]=Pu​+∑v∈children(u)​exclude[v]

exclude[u]=∑v∈children(u)​max(include[v],exclude[v])

1. [0.6 points] Suppose that, in the absence of good market research, owner decides that all sites are equally good, so the goal is simply to design a cafe placement with the largest number of locations. Give a simple greedy algorithm for this case, and prove its correctness.

**Greedy Algorithm for Largest Number of Locations**:

Start with an empty set U.

While there are vertices remaining:

Choose a leaf vertex u (a vertex with degree 1).

Add uuu to U.

Remove uuu and its neighbor from the tree.

1. [0.5 points] Now suppose that the graph is arbitrary, not necessarily acyclic. Give the fastest correct algorithm you can for solving the problem.

**Reduction to Maximum Weighted Independent Set on General Graphs:**

The problem is equivalent to finding the maximum weighted independent set in the graph. However, this problem is NP-hard. We can use a heuristic or approximation algorithms for practical purposes, or exact algorithms for small instances.

**Approximation Algorithm:**

1. Use a greedy algorithm to approximate the solution:
   1. Sort vertices by their profits.
   2. Iteratively add the highest remaining profit vertex to the solution, removing it and its neighbors from the graph.

**Exact Algorithm for Small Instances:**

1. Use dynamic programming with memoization on tree decompositions or branch and bound techniques.
2. For small graphs, exact algorithms like branch and bound or ILP formulations can be used.