Design and Analysis of Algorithms: Problem 1-1

Ben Chaplin

Contents

1 Problem statement

Drunken Donuts, a new wine-and-donuts restaurant chain, wants to build 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 $p(u)$, which describes 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 set $U \subseteq V$ of sites that maximizes the total profit $\sum_{u \in U} p(u)$.

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

(b) Give an efficient algorithm to determine a placement with maximum profit.

(c) Suppose that, in the absence of good market research, DD decides that all sites are equally good, so the goal is simply to design a restaurant placement with the largest number of locations. Give a simple greedy algorithm for this case, and prove its correctness.

2 Varying-profitable locations

2.1 Profit-maximizing algorithm

Algorithm 1 DP algorithm for maximizing profit

Input
G an acyclic graph with weighted vertices describing profit

Output
p the maximum profit over non-adjacent vertices

```
1: procedure MAXPROFIT(v)
2:     if v has no children then
3:         return p(v)
4:     else
5:         a ← p(v) + Σ_{v∈v.grandchildren} MAXPROFIT(v)
6:         b ← Σ_{v∈v.children} MAXPROFIT(v)
7:         return max(a, b)
```

8: **end if**
9: **end procedure**

2.2 Correctness

Definition. Let v be a vertex in a tree, and H be the tree it roots. The **maxprofit of v** is the maximum profit over all subsets $U \subseteq H$, where no two vertices in U are adjacent.

Proof. (Induction on the height of G)
Base case 1: assume G has height 1, and thus only one vertex v. Then **Algorithm 1** will return $p(v)$, which is the maxprofit of G.
Base case 2: assume G has has height 2. Let h be the root of G.
Case 1: $p(h) > \sum_{v \in h.children} p(v)$
Algorithm 1 will return $p(v)$, as $a > b$.
Case 2: $p(h) \leq \sum_{v \in h.children} p(v)$
Algorithm 1 will return $\sum_{v \in h.children} p(v)$, as $b \geq a$.

Inductive hypothesis: Assume **Algorithm 1** is correct when run on graphs of height $n-1$ and $n-2$.

Suppose $|G| = n > 2$ and **Algorithm 1** is run on G. Let h be the root of G.

Case 1: suppose h is included in the profit-maximizing subset of vertices.

Then, **Algorithm 1** returns the sum of $p(h)$ with recursive calls on the grandchildren of h. The trees under the grandchildren of h have height $n-2$, so by our hypothesis, we can be sure that the algorithm returns the maxprofit of each grandchild.

Furthermore, no grandchild is adjacent to another, so no member of either's tree is adjacent to a member of another's tree.

Therefore, **Algorithm 1** returns the maxprofit of h.

Case 2: suppose otherwise.

Then, **Algorithm 1** returns the sum $\sum_{v \in h.children} p(v)$. The trees under each child v have height $n-1$, so by our hypothesis, we can be sure that the algorithm returns the maxprofit under each child.

Furthermore, no child is adjacent to another, so no member of either's tree is adjacent to a member of another's tree.

Therefore, **Algorithm 1** returns the maxprofit of h. □

2.3 Runtime analysis

Algorithm 1 performs a lookup on each vertex in the G exactly once, thus the runtime is $O(|V|)$.