Computer Algorithms I

Spring 2020

Midterm-2-Practice

1. Coin changing

denominations $c_1 < \ldots < c_k$ (e.g., 1, 5, 10, 25), assume $c_1 = 1$

make change for n cents using the minimal number of coins!

idea: let $coins(i)$ be the minimal number of coins needed to make change for i cents

greedy is not always optimal! e.g., $c_1 = 1, c_2 = 3, c_3 = 4$ and $n = 6$

$coins(n) = \min(1 + coins(n - c_i) \, : \, i = 1, \ldots, k, c_i \leq n)$

running time $O(n \cdot k)$

Midterm-2-Practice

2. Average completion time

activities $a_1, \ldots, a_n$ with processing times $p_1, \ldots, p_n$

schedule: ordering of the activities, a_i completed in time t_i

find schedule minimizing average completion time

idea: here a greedy algorithm works

for ordering $a_1, a_2, \ldots, a_n$ the total completion time is

$$n \cdot p_1 + (n-1) \cdot p_2 + \ldots + p_n$$

non-decreasing order: assume numbering is such that $p_1 \leq p_2 \ldots \leq p_n$

This is optimal: in any other order there is an i such that $p_i > p_{i+1}$. Switching these activities the average completion time decreases.

3. Let $G = (V, E)$ be a connected graph with edge weights such that the edge weights are all different. Show that there is a unique minimum spanning tree.

Note: the argument that then Prim's algorithm always has a unique choice is not correct. Why?

Let T_1, T_2 be two different minimum spanning trees. Consider the smallest weight edge which is in only one of the trees, say e is in T_1. Adding e to T_2 a cycle is formed. Some edge f on the cycle is not in T_1. Therefore $w(f) > w(e)$. But then $T_2 + e - f$ is better than T_2, contradiction.

Midterm-2-Practice

4. True or false? Let G be a directed graph. If there is a directed path from u to v, and in a DFS $u.d < v.d$ then v is a descendant of u.

False. Let the edges be $(r, u), (r, v), (u, r)$, r be the root and let the adjacency list of r be u, v.

Midterm-2-Practice

5. Given an acyclic graph and two vertices s and t, find the number of directed paths from s to t!
Give a $O(|V| + |E|)$ time algorithm for this problem.

apply topological sorting to G and return the linked list of vertices between s and t

let $path(u, t)$ be the number of paths from u to t

$$path(u, t) = \sum_{v \in G.Adj[u]} path(v, t)$$

compute $path(u, t)$ backwards from t

running time is the same as DFS (array size + sum of out-degrees)