CS 473: Fundamental Algorithms, Fall 2011

Discussion 11

November 1, 2011

11.1 FIND YOUR GRAPH.

Let k be a relatively small number (say, equal to 10 or 20). A k-graph is an undirected graph having k vertices (numbered explicitly as 1 to k). You want to build a data-structure such that you can do the following two operations:

(A) Store an incoming k-graph in the data-structure.
(B) Given a k-graph G, the data-structure retrieves all the k-graphs stored in the data-structures that are identical (as far as the edges are concerned).

Describe how to build such a data-structure that performs the insertion operation in $O(k^2)$ time, and can answer a query in $O(k^2 + m)$ time, where m is the number of graphs returned.

(A) Describe such a data-structure that does not use hashing.
(B) Describe such a data-structure that uses hashing. Describe the hashing function explicitly. What is the probability for two distinct graphs to collide under your hash function?

11.2 REARRANGEABLE MATRICES.

Let M be an $n \times n$ matrix with each entry equal to either 0 or 1. Let m_{ij} denote the entry in row i and column j. A *diagonal entry* is one of the form m_{ii} for some i.

Swapping rows i and j of the matrix M denotes the following action: we swap the values of m_{ik} and m_{jk}, for $k = 1, \ldots, n$. Swapping two columns is defined analogously.

We say that M is ***rearrangeable*** if it is possible to swap some of the pairs of rows and some of the pairs of columns (in nay sequence) so that after all the swapping, all the diagonal entries of M are equal to 1.

(a) Give an example of a matrix M that is not rearrangeable, but for which at least one entry in each row and each column is equal to 1.

(b) Give a polynomial-time algorithm that determines whether a matrix M with 0-1 entries is rearrangeable.

11.3 THE PROBLEM WITH CHANGE.

Consider the problem of making change for n cents using the least number of coins.

(a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

(b) Suppose that the available coins have the values $c^0, c^1, ..., c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.

(c) Give a set of 4 coin values for which the greedy algorithm does not yield an optimal solution, show why.

(d) Give a dynamic programming algorithm that yields an optimal solution for an arbitrary set of coin values.

11.4 Small Changes to MST.

Let G be a connected, undirected graph where each edge e has weight $w(e)$. You may assume all edge weights are positive and distinct. Consider a Minimum Spanning Tree T of G. Suppose that we decrease one of the edges not in T to a new distinct, positive value. How could you find the MST in the modified graph?

11.5 Flow Facts?

Which of the following statements are true and which are false? Justify your answer.

(a) If all directed edges in a network have distinct capacities, then there is a unique max flow.

(b) Consider a graph $G = (V, E)$. Now, for each edge $e = (u, v)$ with capacity $c(e)$, we will add an edge $e' = (v, u)$ in the opposite direction with the same capacity $c(e)$. This alteration to G will not change the value of the max flow.

11.6 Randomized Max Cut.

Consider the *Max Cut* problem: given an undirected graph $G = (V, E)$ and weight function $w : E \to Z^+$, find a cut (A, B) such that the value of the weights across the cut is *maximized.* We will now analyze a simple randomized algorithm for this problem:

For each v, independently put it in A with probability $1/2$. Output the cut $(A, V \setminus A)$.

(a) What is the probability of edge (u, v) being in the cut?

(b) What is the expected weight of the edges in the cut?

(c) What is the maximum weight of any cut?