Homework 5
Com S 311
Due: April 24, 11:59PM

Late Submissions are NOT accepted.

There are 4 problems and each problem is worth 60 points. When asked to design and algorithm, please write *pseudo code*. If you write Java/C code, it will not be graded.

You are NOT allowed to use hashing/hash tables in this HW

Unless stated otherwise, graphs are always represented in the Adj. list representation

1. Given an undirected graph $G = (V, E)$, with $V = \{1, 2, \cdots, n\}$. A set $S \subseteq V$ is a dominating set if every vertex $v \in V$ either belongs to S or is adjacent to a vertex in S. Given a graph, we would like to compute a dominating set of smallest size. Consider the following natural greedy algorithm that attempts to compute a dominating set. Given $v \in V$, let

$$N(v) = \{y \in V \mid \langle v, y \rangle \in E\}$$

   (a) Input $G = (V, E)$.

   (b) Set $D = \emptyset$

   (c) While V is not empty do

   - let $v \in V$ be a vertex with largest degree.
   - Add v to D.
   - Remove v and all vertices in $N(v)$ from G (and thus from V).

   (d) Output D.

   Note: When we remove a vertex x from G, we remove x from V, and all edges that are incident on x from E.

   Using appropriate data structures, design an efficient implementation of the above algorithm and analyze/derive the runtime of your implementation. Express the run-time as a function of number of edges (m) and number of vertices (n). Your grade partly depends on efficiency.

   Remark. The above algorithm is a heuristic. It will not always produce a smallest dominating set. Think about graphs on which the above algorithm fails to produce a smallest dominating set.

2. Let $G = (V, E)$ be an undirected graph with $V = \{1, 2, \cdots, n\}$. Given a vertex x, let

$$N(x) = \{y \in V \mid \langle x, y\rangle \in E\}$$

A subset $S \subseteq V$ is called *independent* if for every pair of distinct vertices x, y from S, the following holds:

$$N(x) \cap N(y) = \emptyset$$

A set $S \subseteq V$ is called *maximal independent* if S is independent and for every $y \notin S$, $S \cup \{y\}$ is not independent.

- Design an algorithm that gets an undirected graph $G = (V, E)$ and a subset $S \subseteq V$ as input and determines if S is independent or not. You may assume $V = \{1, 2, \cdots, n\}$ Derive the run-time of your algorithm. Part of your grade depends on the run-time.
- Design an algorithm that gets a graph $G = (V, E)$ as input and outputs a maximal independent set of G. Derive the run-time of your algorithm. Your grade partly depends on the run-time. You may assume $V = \{1, 2, \cdots, n\}$.

3. Let M be a matrix of integers with n rows and m columns and let $M[i, j]$ denote the entry in ith row and jth column. Both rows and columns are indexed from 1. A *horizontal cut* in M is a sequence $[c_1, c_2, \cdots, c_m]$ such that

- For every i, $1 \leq c_i \leq n$
- For $1 \leq i \leq m - 1$, $c_{i+1} \in \{c_i - 1, c_i, c_i + 1\}$

Given a horizontal cut $[c_1, c_2, \cdots, c_m]$, its cost is $M[c_1, 1] + M[c_2, 2] + \cdots + M[c_m, m]$. A horizontal cut is a *max-cost cut* if its cost is at least the cost of any other horizontal cut.

Give a dynamic programing algorithm, that gets a matrix M as input, and outputs the *cost of the* max-cost cut. Let $C[i, j]$ be the cost of a max-cost horizontal cut that ends at $M[i, j]$. State the recurrence relation for $C[i, j]$. Based on this recurrence relation arrive at an iterative algorithm. Derive the run time of your algorithm. Note that your algorithm need not output a max-cost cut, it suffices to output only its cost. Part of your grade depends on the run-time.

4. Let $S = \{x_1, x_2, \cdots, x_n\}$ be a set of distinct non-negative integers. Give a dynamic programming algorithm that gets S as input and determines of there is a subset $S' \subseteq S$ such that

$$\sum_{x \in S'} x = \sum_{x \in S - S'} x$$

Your algorithm must be based on an appropriately defined recurrence relation and must be iterative. Derive the run-time of your algorithm. Express run-time as a function of n and N, where $N = x_1 + x_2 + \cdots + x_n$. Part of your grade depends on the run-time.

GUIDE LINES: Same as before.