COMP 5711: Advanced Algorithms
Written Assignment # 1

Fixed Parameter Algorithms (KT Ch 10)

Problem 1 (20pts) We claimed that the Hitting Set Problem was NP-complete. To recap the definitions, consider a set $A = \{a_l.....a_n\}$ and a collection $B_1, B_2, \ldots, B_m$ of subsets of A. We say that a set $H \subseteq A$ is a *hitting set* for the collection $B_1, B_2, \ldots, B_m$ if H contains at least one element from each B_i, that is, if $H \cap B_i$ is not empty for each i. (So H "hits" all the sets B_i.)

Now suppose we are given an instance of this problem, and we'd like to determine whether there is a hitting set for the collection of size at most k. Furthermore suppose that each set B_i has at most c elements. Give an algorithm that solves this problem with a running time of the form $f(c, k) \cdot \text{poly}(n, m)$.

Problem 7 (20pts) The *chromatic number* of a graph is the minimum k such that it has a k-coloring, i.e., each vertex is assigned one of k colors such that no two adjacent vertices share the same color.

(a) Show that the chromatic number of a graph G is at most $w + 1$, where w is the tree-width of G.

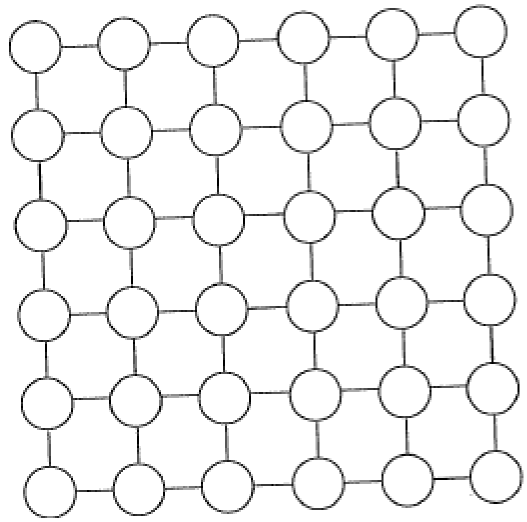
(b) Give an algorithm with running time $f(w) \cdot \text{poly}(n)$ to find the chromatic number of a graph G, where n is the number of vertices in G and w is its tree-width. You may assume that a tree decomposition of G is already given.

Approximation Algorithms (KT Ch 11)

Problem 7 (20pts) You're consulting for an e-commerce site that receives a large number of visitors each day. For each visitor i, $i = 1, 2, \ldots, n$, the site has assigned a value v_i. Each visitor i is shown one of m possible ads $A_1, A_2, \ldots, A_m$ as they enter the site. Given a selection of one ad for each customer, we will define the *spread* of this selection to be the minimum, over $j = 1, 2, \ldots, m$, of the total value of all customers who were shown ad A_j.

Example. Suppose there are six customers with values 3, 4, 12, 2, 4, 6, and there are $m = 3$ ads. Then, in this instance, one could achieve a spread of 9 by showing ad A_1 to customers 1, 2, 4, ad A_2 to customer 3, and ad A_3 to customers 5 and 6.

The ultimate goal is to find a selection of an ad for each customer that maximizes the spread. Unfortunately, this optimization problem is NP-hard (you don't have to prove this). So instead, we will try to approximate it.



a) Give a polynomial-time algorithm that approximates the maximum spread to within a factor of 2. That is, if the maximum spread is s, then your algorithm should produce a selection of one ad for each customer that has spread at least $s/2$. In designing your algorithm, you may assume that no single customer has a value that is significantly above the average; specifically, if $\bar{v} = \sum_{i=1}^{n} v_i$ denotes the total value of all customers, then you may assume that no single customer has a value exceecling $\bar{v}/(2m)$.

b) Give an example of an instance on which the algorithm you designed in part (a) does not find an optimal solution. Say what the optimal solution is in your sample instance, and what your algorithm finds.

Problem 10 (20pts) Suppose you are given an $n \times n$ grid graph G, as in the figure above. Associated with each node v is a weight $w(v)$, which is a nonnegative integer. You may assume that the weights of all nodes are distinct. Your goal is to choose an independent set S of nodes of the grid, so that the sum of the weights of the nodes in S is as large as possible.

Consider the following greedy algorithm for this problem.

> Start with S equal to the empty set;
> **while** *some node remains in* G **do**
>   Pick a node v_i of maximum weight;
>   Add v_i to S;
>   Delete v_i and its neighbors from G;
> **end**
> Return S;

Algorithm 1: Heaviest-first algorithm

a) Let S be the independent set returned by the "heaviest-first" greedy algorithm, and let T be any other independent set in G. Show that, for each node $v \in T$, either $v \in S$, or there is a node $v' \in S$ so that $w(v) \leq w(v')$ and (v, v') is an edge of G.

b) Show that the "heaviest-first" greedy algorithm returns an independent set of total weight at least $\frac{1}{4}$ times the maximum total weight of any independent set in the grid graph G.

Problem 11 (20 pts) Consider the following variant of the knapsack problem. There are n items, each with a weight w_i and a value v_i. Now, you're told that there is a subset O whose total weight is $\sum_{i \in O} w_i \leq W$, where W is the weight limit, and whose total value is $\sum_{i \in O} v_i = V$ for some V. For a given fixed $\epsilon > 0$, design a polynomial-time algorithm that finds a subset of items $\mathcal{A}$ such that $\sum_{i \in \mathcal{A}} w_i \leq (1 + \epsilon)W$ and $\sum_{i \in \mathcal{A}} v_i \geq V$, i.e., your algorithm does at least as good as O in terms of value, but it may exceed the weight limit slightly. Note that your algorithm is given the value of W but not V.