University of British Columbia
Computer Science Department

FINAL EXAMINATION IN CS 500

SPRING 2009

Professor William Evans

This is a closed book examination. You have 2 hours and 30 minutes. Do all 5 problems. Each is worth 20 points. Write your solutions on the examination sheets.

Your name: _____

1. (20 points) **Log Cutting** We run a lumber mill that cuts logs into smaller logs to make (rustic) chopping blocks. Given a log of length n feet, we want to cut it into n logs of length 1 foot each by performing a sequence of cutting operations. The cost of a cutting operation is 1 plus the weight of the log being cut. Let w_i be the weight of the ith foot-length section of our log. So the original log weights $\sum_{i=1}^{n} w_i$. Describe a dynamic programming algorithm that, given $w_1, w_2, \ldots w_n$, finds the minimum cost to cut the log into foot length sections. What is the running time of your solution?

   For example, if $w_1 = 3$, $w_2 = 9$, and $w_3 = 10$, then cutting at 2 and then 1 costs $22 + 12 = 34$, whereas cutting at 1 and then 2 costs $22 + 19 = 41$.

2. Suppose we have n workers numbered 1 to n and m jobs numbered 1 to m. Each job requires exactly one worker and each worker can perform at most c jobs. We are told that worker i can only perform jobs in the set W_i.

   (a) (10 points) Describe an efficient algorithm to assign workers to jobs so that all the jobs are done (or report that they cannot all be done). For example, if $n = 3$, $m = 5$, $c = 2$, $W_1 = \{1, 3\}$, $W_2 = \{2, 4, 5\}$, and $W_3 = \{2, 3, 5\}$, the following assignment satisfies the constraints and gets all the jobs done: $A_1 = \{1, 3\}$, $A_2 = \{4\}$, $A_3 = \{2, 5\}$ where worker i performs jobs in A_i.

   (b) (10 points) Suppose in addition to the above constraints, the jobs are split over k days and each worker can work on at most one job per day. Let $D_1, D_2, \ldots, D_k$ be the partition of the m jobs so that D_i is the set of jobs that happen on day i. (Note: $D_i \cap D_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^{k} D_i = \{1, 2, \ldots, m\}$.) Again, describe an efficient algorithm to assign workers to jobs.

   Hint: Add k nodes for each worker to your flow network.

(blank page)

3. We have n items with positive weights $W = (w_1, w_2, \ldots, w_n)$ and a truck that can carry at most weight B. We want to load our truck with items to achieve maximum total weight, subject to the constraint that the total weight is at most B.

   For example, if $W = (8, 2, 4)$ and $B = 11$, the optimal solution is to load the items with weights 8 and 2.

   (a) (5 points) Here is a greedy algorithm for loading the truck:

   $S = \emptyset$
   $L = 0$
   `for` $i = 1$ `to` n
      `if` $L + w_i \leq B$ `then`
         $S = S \cup \{i\}$
         $L = L + w_i$
   `return` S

   Give an instance in which the total weight of the items returned by the algorithm is less than half the weight of the optimal solution.

   (b) (15 points) Describe an algorithm and prove that it produces a set of items whose total weight is at most B and at least $1/2$ times the weight of the optimal solution. Your algorithm should run in time $O(n \log n)$.

4. (20 points) A **marking algorithm** for page replacement stores only a single mark bit per page in the cache and proceeds in phases. When a page in the cache is requested, it is marked. On a page fault, a marking algorithm evicts an unmarked page, but if all pages are marked, the algorithm first unmarks all pages, beginning a new phase, and then evicts an unmarked page. It then brings in the requested page (and marks it). There are many different marking algorithms based on which of the unmarked pages the algorithm choses to evict. For example, LRU is a marking algorithm.

   The following table shows the contents of a three page cache maintained by a marking algorithm for the sequence of page requests shown in the top row.

A	B	C	D	B	A	B	C	D	A	B
$A^\bullet$	$A^\bullet$	$A^\bullet$	$D^\bullet$	$D^\bullet$	$D^\bullet$	$D^\bullet$	D	$D^\bullet$	$D^\bullet$	D
B	$B^\bullet$	$B^\bullet$	B	$B^\bullet$	$B^\bullet$	$B^\bullet$	$C^\bullet$	$C^\bullet$	$C^\bullet$	C
C	C	$C^\bullet$	C	C	$A^\bullet$	$A^\bullet$	A	A	$A^\bullet$	$B^\bullet$
phase 1			phase 2				phase 3			phase 4

   Prove that any marking algorithm is k-competitive, where k is the cache size in pages.

5. (20 points)

INDEPENDENT SET (IS)
Input: Graph $G = (V, E)$ and a number k.
Question: Does G contain an independent set of size at least k? (An independent set is a subset S of the vertices such that no two vertices in S share an edge.)

SET PACKING (SP)
Input: A set U of m elements, a collection $S_1, S_2, \ldots, S_n$ of subsets of U, and a number k.
Question: Does there exist a collection of at least k of these sets such that no two sets in the collection intersect?

Use the fact that INDEPENDENT SET is NP-complete to show that SET PACKING is NP-complete.