CS 341, Winter 2013
Algorithms
Timothy Chan, Jakub Truszkowski

University of Waterloo Final Examination

Last Name: ____________ First Name: ____________

Signature: ____________

ID number: ____________

- Date: April 17, 2013.
- Start time: 12:30pm. End time: 3:00pm. Duration: 2 hrs and 30 mins.
- Number of pages (including cover and one blank page): 16.
- Exam type: Closed book. No additional materials are allowed.
- Print your initials at the top of each page (in case a page gets detached).
- All answers should be placed in the spaces given. Backs of pages may be used as scratch papers and will not be marked (unless you clearly indicate otherwise). If you need more space to complete an answer, you may use the blank page at the end.
- Cheating is an academic offense. Your signature on this exam indicates that you understand and agree to the University's policies regarding cheating on exams.

Q	Marks	Init.
1	/10	
2(a,b,c)	/10	
2(d,e,f)	/9	
2(g)	/5	
3(a,b)	/14	
3(c)	/6	
4(a)	/6	
4(b)	/6	
5(a,b)	/7	
5(c,d)	/7	
6(a)	/5	
6(b)	/15	
Total	/100	

1. [*10 marks*] Consider the following recurrence:

$$T(n) = \begin{cases} 7\,T(\lfloor n/2 \rfloor) + n^3 & \text{if } n \geq 3 \\ 2013 & \text{if } n \leq 2. \end{cases}$$

Use the guess-and-check (i.e., induction) method to prove the upper bound $T(n) = O(n^3)$.

2. [*24 marks*] *Short questions.*

   (a) [*5 marks*] Arrange the following six functions in increasing order of growth rate. (Justifications are not required.) All logarithms are in base 2.

   $$n2^{\sqrt{\log n}}, \ n\sqrt{\log n}, \quad n^{1.59}, \quad 2^n, \quad n\log\log n, \quad n^{2013}1.99^n.$$

   (b) [*2 marks*] Given the following array of fifteen numbers

   $$\langle 10,\ 3,\ 16,\ 9,\ 2,\ 4,\ 31,\ 12,\ 5,\ 8,\ 0,\ 30,\ 28,\ 16,\ 15\rangle,$$

   what is the pivot chosen for this list by Blum, Floyd, Pratt, Rivest, and Tarjan's deterministic linear-time algorithm for the selection problem? Briefly explain.

   (c) [*3 marks*] Usually in a greedy algorithm, at each iteration we make a decision, and we never change decisions made in the past. Give an example of a greedy algorithm discussed in class which violates this principle.

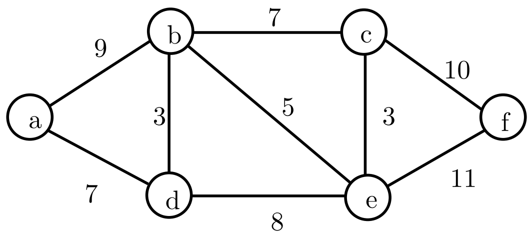
(d) [*3 marks*] Consider the following variant of the "disjoint intervals" (also called "activity selection") problem from class: given n (possibly overlapping) intervals $[a_1, b_1], \ldots, [a_n, b_n]$, chosen a subset of nonoverlapping intervals which maximizes the *total length* of the chosen intervals (not the number of chosen intervals).

Recall the greedy algorithm from class, which at every iteration chooses the interval $[a_i, b_i]$ with the smallest b_i (i.e, the activity with the earliest finish time). Does this algorithm work correctly for this variant of the problem? Justify your answer.

(e) [*3 marks*] Is the dynamic programming algorithm for the longest common subsequence problem (LCS) from class considered a polynomial-time algorithm? Explain.

(f) [*3 marks*] When depth-first search (DFS) and breadth-first search (BFS) are run on a directed graph, one of the two algorithms cannot produce forward edges. Explain which one, and why.

(g) [*5 marks*] Run Dijkstra's shortest path algorithm on the following graph from the source vertex a. Show the order in which vertices are inserted into the set S, the shortest path tree, and the δ values (shortest path distances).



3. [*20 marks*] *Dynamic programming.* Consider the following problem: given two sequences of integers $A = \langle a_1, \ldots, a_m \rangle$ and $B = \langle b_1, \ldots, b_n \rangle$ with $m \leq n$, find a subsequence $\langle b_{j_1}, \ldots, b_{j_m} \rangle$ of B (with $j_1 < j_2 < \cdots < j_m$) that minimizes the *cost* function $|a_1 - b_{j_1}| + \cdots + |a_m - b_{j_m}|$.

   Here is a proposed dynamic programming solution: define $C[i, j]$ to be the minimum cost for the the input sequences $\langle a_1, \ldots, a_i \rangle$ and $\langle b_1, \ldots, b_j \rangle$. For the base cases, $C[0, j] = 0$ for all $j = 0, \ldots, n$, and $C[i, 0] = \infty$ for all $i = 1, \ldots, m$. The recursive formula is as follows (you do not need to justify the correctness of the formula):

$$C[i, j] = \min\{C[i, j-1],\ C[i-1, j-1] + |a_i - b_j|\}.$$

   (a) [*7 marks*] Write pseudocode for computing the optimal cost using the above solution. (Do not use memoization.)

(b) [*4 marks*] Write pseudocode for retrieving the optimal subsequence.

(c) [*3 marks*] Analyze the total running time and space usage of the whole algorithm as a function of m and n.

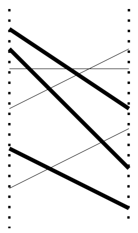
(d) [*6 marks*] Consider a new variant of the problem: given two sequences of integers $A = \langle a_1, \ldots, a_m \rangle$ and $B = \langle b_1, \ldots, b_n \rangle$, and given an extra number K (call it the "length parameter"), with $K \leq m \leq n$, find a subsequence $\langle a_{i_1}, \ldots, a_{i_K} \rangle$ of A and a subsequence $\langle b_{j_1}, \ldots, b_{j_K} \rangle$ of B that minimize the cost function $|a_{i_1} - b_{j_1}| + \cdots + |a_{i_K} - b_{j_K}|$.

To solve this modified problem, redefine $C[i, j, k]$ to be the minimum cost for the input sequences $\langle a_1, \ldots, a_i \rangle$ and $\langle b_1, \ldots, b_j \rangle$ with given length parameter k.

Write a new recursive formula for $C[i, j, k]$. Justify your answer. (You don't have to give base cases or pseudocode.)

4. [*12 marks*] *Graph algorithms.*

   (a) [*6 marks*] We are given a set S of n (possibly intersecting) line segments where all the left endpoints are on a common vertical line and all the right endpoints are on another common vertical line. We want to find a subset T of S with the largest number of line segments, such that no two line segments in T intersect.



   Show that this problem can be reduced to finding a longest path in a directed acyclic graph (dag) with n vertices and $O(n^2)$ edges (hence, the problem can be solved in $O(n^2)$ time).

(b) [*6 marks*] We are given a weighted connected undirected graph $G = (V, E)$ with m edges. Suppose that we have already precomputed the minimum spanning tree T^* of G. Show that given an edge $e \in T^*$, we can compute the minimum spanning tree of $G - \{e\}$ (the graph obtained by deleting the edge e from G) in $O(m)$ time.

[Hint: use a lemma from class.]

5. [*14 marks*] *Short questions on* NP.

   (a) [*4 marks*] Convert the following optimization problem into a decision problem in NP (you do not need to prove that the decision problem is in NP).

      Input: a set P of n points in 2D and a number r.

      Output: a subset $Q \subseteq P$ with the largest size (number of points) such that the closest pair in Q has distance at least r.

   (b) [*3 marks*] Show that if your decision problem in (a) can be solved in polynomial time, then we can compute the size of the optimal solution in polynomial time as well.

(c) [*4 marks*] Which of the two problems below is in NP? Justify your answer.

PROBLEM X:

Input: undirected graph $G = (V, E)$ and an integer K.

Output: "yes" iff every clique in G has size at most K.

PROBLEM Y:

Input: complete weighted undirected graph $G = (V, E)$ with weight function w, and a number W.

Output: "yes" iff

$$\min_{S \subseteq V,\ |S| = n/2} \sum_{u \in S,\ v \in V - S} w(uv) \ \leq\ W.$$

(d) [*3 marks*] True or False: The halting problem is in NP. Briefly justify your answer.

6. [*20 marks*] *Proving NP-completeness.* Consider the following problem PRECISE-SET-COVER:

   Input: a set U of M elements, where each element e has an integer weight w_e, and a collection of N subsets $S_1, \ldots, S_N$ of U.

   Output: "yes" iff there exists a subcollection $T \subseteq \{S_1, \ldots, S_N\}$ such that each element $e \in U$ appears in precisely w_e subsets of T.

   For example, if input has $U = \{1, 2, 3, 4, 5\}$, $S_1 = \{1, 3, 5\}$, $S_2 = \{3, 5\}$, $S_3 = \{2, 4, 5\}$, $S_4 = \{2, 5\}$, and $w_1 = w_2 = w_3 = w_4 = 1$, $w_5 = 2$, then the output is "yes", by choosing the subcollection $T = \{S_1, S_3\}$.

   (a) [*5 marks*] Prove that PRECISE-SET-COVER is in NP.

(b) [*15 marks*] Prove that PRECISE-SET-COVER is NP-complete, using the known fact that VERTEX-COVER (VC) is NP-complete. (Remember that partial marks may be given if you can fill in at least the outline of the proof correctly.)

- Give a polynomial-time reduction from ________________ to ________________.

  Given:

  To construct:

  The construction:
  (Hint: The elements are the edges. For each vertex u, create a set containing the edges incident to u. For each edge e, make $w(e) = 2$. You may need to create extra sets of size 1, and an extra element whose purpose is to ensure that exactly K vertices are chosen.)

- Prove the correctness of your reduction.

  To show:

  Proof (two directions):

(Extra space)