CSIE 2136: Algorithm Design and Analysis

Final Exam

Time: 10:20-13:20 (180 minutes) January 16, 2015

Instructions

- This is a 3-hour close-book exam.
- There are 7 questions worth of 100 points plus 35 bonus points. The maximum of your score is 100, so you can allocate your time and select problems based on certain optimal strategies of your own.
- For any of the questions or sub-questions except bonus ones, you get 1/5 of the credit when leaving it blank and get 0 when the answer is completely wrong. You get partial credit when the answer is partially correct.
- Please write clearly and concisely; avoid giving irrelevant information.
- You have access to the appendix on the last page.
- Tips: read every question first before start writing. Exam questions are not all of equal difficulty, move on if you get stuck.
- Please print **your name, student ID, and page number on every answer page**.

Problem 1: Short Answer Questions (30 points)

Answer the following questions and briefly justify your answer to receive full credit. Clear and concise (e.g., in one or two sentences) explanations are appreciated.

(a) (3 points) True or False: If P=NP, every NP-complete problem can be solved in polynomial time.

(b) (3 points) True or False: If A $\leq_p$ B and there is an $O(n^3)$ algorithm for B, then there is an $O(n^3)$ algorithm for A.

(c) (3 points) True or False: If A can be reduced to B in $O(n^2)$ time and there is an $O(n^3)$ algorithm for B, then there is an $O(n^3)$ algorithm for A.

(d) (3 points) True or False: It is more efficient to represent a sparse graph using an adjacency matrix than adjacency lists.

(e) (3 points) True or False: Kruskal's algorithm and Prim's algorithm always output the same minimum spanning tree when all edge weights are distinct.

(f) (3 points) True or False: A weakly connected directed graph with n vertices and n edges must have a cycle. (A directed graph is weakly connected if the corresponding undirected graph is connected.)

(g) (4 points) Describe a computational problem that is not in NP, that is, cannot be verified in polynomial time.

(h) (4 points) Bloom filters support worst-case $O(1)$ time membership testing at the cost of false positives. Explain the meaning of false positives here.

(i) (4 points) Construct an example demonstrating that plurality is not strategyproof. Recall that plurality is a voting rule in which each voter awards one point to top candidate, and the candidate with the most points wins, and a voting rule is strategyproof if a voter can never benefit from lying about his or her preferences.

Problem 2: Maximum Independent Sets (15 points)

An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V'. The independent-set problem is to find a maximum-size independent set in G. We denote by A_{opt} and A_{dec} the optimization problem and the decision problem of the independent-set problem, respectively.

(a) (3 points) Describe the corresponding decision problem, A_{dec}.

(b) (3 points) Show that $A_{dec} \leq_p A_{opt}$.

(c) (3 points) Show that $A_{opt} \leq_p A_{dec}$.

(d) (6 points) Although the related decision problem is NP-complete, finding an independent set of maximum size on certain special graphs can be done in polynomial time. Suppose G is an n-by-m grid, express the maixmum size of independent sets as a function of n and m.

Problem 3: Approximation Algorithms (20 points)

in Homework 5, we learned how to reduce 3-CNF-SAT to k-CNF-SAT for any integer $k > 3$ in polynomial time. Answer the following related questions:

(a) (10 points) The MAX-k-CNF-SAT problem is similar to the k-CNF-SAT problem, except that it aims to return an assignment of the variables that maximize the number of clauses evaluating to 1. Design a randomized $(2^k/(2^k - 1))$-approximation algorithm for k-CNF-SAT ($k > 3$). Your algorithm should run in polynomial time. To simplify this question, you can assume that no clause contains both a variable and its negation.

(b) (10 points) Your friend Ada claimed that, for any k, she can design a $(2^k/(2^k-1))$-approximation algorithm for the MAX-3-CNF-SAT problem. Given a k and a formula in the 3-CNF form, her algorithm first reduces the formula into the k-CNF form, and then solves the MAX-k-CNF-SAT problem (as described in part (a)).

Do you agree with her? Justify your answer.

Problem 4: Negative-Weight Edges (20 points)

(a) (10 points) Explain why Dijkstra's algorithm (Figure 2) may return incorrect results when the graph contains negative-weight edges. Provide a small (no more than 4 vertices) counterexample as well.

(b) (10 points) Johnson has learned a powerful technique called *reweighting* in Homework 4. Given a graph $G = (V, E)$ and a weight function w, this reweighting technique constructs a new weight function $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ such that $\hat{w}(u, v) \geq 0$ for all $(u, v) \in E$.

Prove that the following statement is true for any function h that maps vertices to real numbers: For any pair of vertices $u, v \in V$, a path p is a shortest path from u to v using the weight function w if and only if p is a shortest path from u to v using weight function $\hat{w}$.

Problem 5: Degree-Constrained Minimum Spanning Tree (15 points)

In class we have learned how to find a minimum spanning tree on a graph in polynomial time.

Now let's consider a variant of minimum spanning trees called *degree-constrained minimum spanning trees*. The decision version of the degree-constrained minimum spanning tree problem is defined like this: Given a graph G, a weight function w over edges, a weight limit W, and a degree constraint b, is there a spanning tree whose total weight is at most W and the degree of each vertex in the spanning tree is at most b?

(a) (5 points) Recall that in Prim's algorithm (Figure 3), we greedily grow a spanning tree from an arbitrary root vertex by adding an edge with the minimal weight such that one end of the edge is connected to the tree and the other end is not. Your friend proposed a slightly modified greedy algorithm to solve the degree-constrained minimum spanning tree problem. In this modified algorithm, an additional check $degree(u) < b$ is conducted at Line 9 in Figure 3.

Provide a counterexample to show that such a greedy algorithm is incorrect.

(b) (10 points) Prove that this problem is NP-complete.

Hint: The Hamiltonian Path problem is a known NP-complete problem.

Problem 6: Bitonic TSP (15 bonus points)

In the euclidean traveling-salesman problem, we are given a set of n points in the plane, and we wish to find the shortest closed tour that connects all n points. Figure 1(a) shows the solution to a 7-point problem. The general problem is NP-hard, and its solution is therefore believed to require more than polynomial time.

J. L. Bentley has suggested that we simplify the problem by restricting our attention to bitonic tours, that is, tours that start at the leftmost point, go strictly rightward to the rightmost point, and then go strictly leftward back to the starting point. Figure 1(b) shows the shortest bitonic tour of the same 7 points. In this case, a polynomial-time algorithm is possible.

Describe an $O(n^2)$-time algorithm for determining an optimal bitonic tour. You may assume that no two points have the same x-coordinate and that all operations on real numbers take unit time.

Hint: Scan left to right, maintaining optimal possibilities for the two parts of the tour using the concept of dynamic programming.

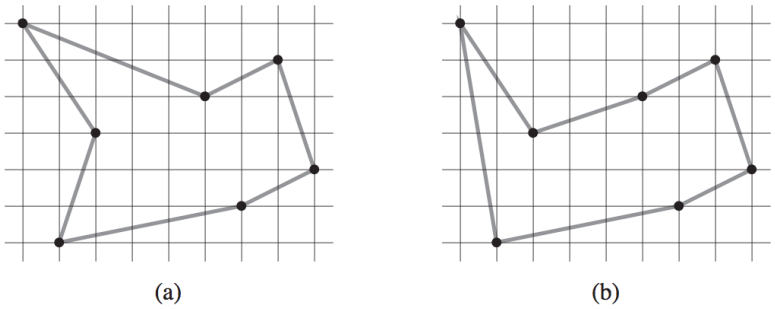

Figure 1: (a) A TSP tour, (b) a bitonic TSP tour.

Problem 7: Dynamic Binary Search (bonus 20 points)

Binary search of a sorted array takes logarithmic search time, but the time to insert a new element is linear in the size of the array. We can improve the time for insertion by keeping several sorted arrays.

Specifically, suppose that we wish to support SEARCH and INSERT on a set of n elements. Let $k = \lceil \log_2(n+1) \rceil$, and let the binary representation of n be $\langle n_{k-1}, n_{k-2}, \cdots, n_0 \rangle$. We have k sorted arrays $A_0, A_1, \cdots, A_{k-1}$, where for $i = 0, 1, \cdots, k-1$, the length of array A_i is 2^i. Each array is either full or empty, depending on whether $n_i = 1$ or $n_i = 0$, respectively. The total number of elements held in all k arrays is therefore $\sum_{i=0}^{k-1} n_i 2^i = n$. Although each individual array is sorted, elements in different arrays bear no particular relationship to each other.

Hint: To simplify the setting, you can compute the running time based on the number of comparisons only, ignoring memory allocation, copy, and free.

(a) (5 points) Describe how to perform the SEARCH operation for this data structure. Analyze its worst-case running time.

(b) (5 points) Describe how to perform the INSERT operation. Analyze its worst-case running time.

(c) (10 points) Analyze the amortized running time of INSERT.

Appendix

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5      u = EXTRACT-MIN(Q)
6      S = S ∪ {u}
7      for each vertex v ∈ G.Adj[u]
8          RELAX(u, v, w)
```

Figure 2: Dijkstra's algorithm.

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Figure 3: Prim's algorithm.