University of New Mexico
Department of Computer Science
Final Examination
CS362: Data Structure and Algorithms

Name:_____
Email:_____

Instructions:

1. Please silence your cell phones.

2. Write your name and email address legibly in the space provided above.

3. Write your name legibly at the upper right hand corner on each page.

4. This is a close-book exam. You must not discuss the questions with anyone except the professor in charge.

5. You are allowed to use a one page double-sided "cheating sheet" that you have brought to the exam and a calculator. Nothing else permitted.

6. Write your answers legibly.

7. There are **6 problems** in this exam, including 1 **optional extra credit problem**.

8. For algorithmic problems, **do not give detailed pseudo-code**. Describe the key ideas and key steps of your solutions, correctness argument, and provide running time analysis. In the case, where you are using a well-known algorithm (e.g., merge sort or linear time selection) as a subroutine **without modification**, just cite the well-known algorithm and its running time. Do not repeat the steps of the well-known algorithms. Don't spend too much time on any single problem. All questions are weighted equally. If you get stuck, move on to something else and get back later.

9. You may assume the following problems are NP-C and use them without proof:

- Hamiltonian path (i.e., a path that goes through every vertex exactly once) on general graph.
- Traveling salesman tour (i.e., a cycle with minimum total edge weights that goes through every vertex exactly once) on general weighted graph.
- Vertex cover problem on general graph.
- Independent set problem on general graph.
- Clique problem on general graph.

10. In the problems where you are asked to design an efficient algorithm, your points earned will depend on the efficiency of your algorithm. However, any solution is better than nothing.

11. Good luck!

1. (10pt) Suppose we roll two six-sided dice. The dice are fair, and one of them is red and the other is blue. Answer the following questions:

(a) (3pt) What is the probability that the sum of the two dice is even?

(b) (3pt) What is the probability that the red dice is larger than the blue dice?

(c) (4pt) What is the expectation of the difference between the red dice and the blue dice? What is the expectation of the absolute value of the difference of the two dice?

Solutions:

(a) 0.5

(b) $Pr\left(\left\{\begin{matrix}(2,1),(3,1),(4,1),(5,1),(6,1),\\(3,2),(4,2),(5,2),(6,2),\\(4,3),(5,3),(6,3),\\(5,4),(6,4),\\(6,5)\end{matrix}\right\}\right) = \frac{15}{36}$

(c) 0 and $\frac{70}{36}$

2.(10pt) Suppose you are given an unsorted array *A* of n integers.

(a) (5pt) Design an algorithm to remove all duplicates in terms of worst-case running time.

(b) (5pt) Design an algorithm to remove all duplicates in terms of average or expected running time.

Solutions:

(a) Heap-sort, Merge-sort. Note Quicksort has a worst case running time of $O(n^2)$ and will not work.

(b) Hashing.

It's also OK to answer bucket sort with a sampling analysis performed to control the probability of buckets to be uniform.

3. (10pt) Recall that given an undirected graph $G(V, E)$:

An ***independent set*** is a subset of vertices *U* such that no two vertices in *U* are adjacent (i.e., joined by an edge in *E*). The independent set problem is given a graph *G* and an integer *k*, determine if *G* has an independent set of *k* vertices.

A ***vertex cover*** is a subset of vertices *C*, such that every edge in *E* is incident on at least one vertex in *C*. The vertex cover problem is given a graph *G* and an integer *k*, determine if *G* has a vertex cover of *k* vertices.

An ***independent vertex cover*** is a subset of vertices that is both an independent set and a vertex cover of *G*. The independent vertex cover problem is to determine if a graph *G* has an independent vertex cover.

From the class we learned that both the ***independent set problem*** and the ***vertex cover problem*** are NP-Complete.

Answer the following questions:

(1) (4pt) Is the independent vertex cover problem in P?

(2) (3pt) Is the independent vertex cover problem in NP? Explain why?

(3) (3pt) Is the independent vertex cover problem in NP-C?

Solutions:

(1) Yes. Observe that the independent vertex cover is simly a 2-coloring, which can be done in polynomial time with BFS.

(2) Yes.

(3) No, unless P = NP.

.

4. (10pt) The low-degree spanning tree problem is as follows: Given a graph G and an integer k, does G contain a spanning tree such that all vertices in the tree have degree at most k (obviously, only the tree edges count towards the degree)?

Determine if the low-degree spanning tree problem is NP-C.

Solution:

Yes. Reduce the Hamiltonian path problem to the low-degree spanning tree problem.

Grading will emphasize on the direction of the reduction.

5. (10pt) Recall that given a graph G = (V, E), a vertex cover of G is a subset of vertices $C \subseteq V$ such that each edge of G has at least one endpoint in *C*. The goal of the vertex cover problem is to find the optimal vertex cover C^* with the minimum number of vertices.
Consider the following algorithm for vertex cover.

Step 1: Start with $C = \phi$.

Step 2: Pick an edge e uniformly at random from the edges that are not covered by *C* (i.e., if *e* has endpoints *u* and *v*, then $\{u, v\} \cap C = \phi$), and add a random endpoint of *e* to *C*.

Step 3: If *C* is a vertex cover, terminate and output *C*; else go to *Step 2*.

Answer the following questions:

(a) (3pt) Consider the very first iteration of the algorithm. What is the probability that a vertex from the smallest vertex cover C^* is added to C?

(Hint: for each edge $e \in E$, at least one endpoint of *e* must be in C^*.)

(b) (3pt) Consider the second iteration of the algorithm. What is the probability that a vertex from the smallest vertex cover C^* is added to C?

(Hint: you should discuss the two scenarios of whether a vertex from C^* is added to C in the first iteration or not.)

(c) (4pt) Let k be the number of vertices in the smallest vertex cover C^*. Show that the expectation of the number of vertices of *C* is $2k$.

Solution:

(a) 0.5, since for each edge, one endpoint will be in the optimal cover.

(b) Still at least 0.5.

(c) We expected to get half of a vertex in the optimal cover in each iteration. Therefore in 2k iterations, we expect to have obtained the vertex cover.

6. **(Extra Credit)** (10pt)

In the class, we discussed the computational complexity classes of P and NP. In this problem, you will be exposed to computational complexity classes for randomized algorithms.

A decision problem belongs to the class of randomized polynomial time (RP) if it has an algorithmic solution satisfying the following properties:

(1) The algorithm runs in polynomial time in the input size.

(2) If the correct answer is NO, the algorithm always returns No.

(3) If the correct answer if YES, the algorithm returns Yes with a probability of at least ½, otherwise, it returns NO.

Answer the following questions:

(a) (4pt) Does the class of P belong to the class of RP? Explain why?

(b) (3pt) Suppose you have a decision problem belonging to the class RP. How to increase the probability of getting a correct YES to 0.99?

(c) (3pt) Recall in the class, we have developed a randomized algorithm for solving the minimum cut problem in a multi-graph. From that algorithm, can you conclude that the min-cut problem belongs to the class of RP?

Solutions:

(a) Yes. Because the deterministic algorithm will return a Yes with a probability of 1.

(b) Run the algorithm k times, where k is calculated with the equation: $1 - \left(\frac{1}{2}\right)^k \geq 0.99$

(c) Yes. Repeating the randomized min-cut n^2 times and improves it probability to be more than ½. Alternatively, it is also OK is say that the min-cut is in P and therefore in RP from (a).

Have a nice Summer Break!