CMPS 356	Fall 2004
Exam 2: Final	
Chiraz BenAbdelkader	*25/01/05*

Name: .....

Time: 180 MINUTES

Important Notes ...

- This exam is **open-book** and **closed-notes**.
- Make sure there are 9 pages and 8 problems for a total of 100 points.
- Use **pseudo-code** notation to write any algorithms you are asked to write.
- Write as legibly as you can, otherwise your answer will NOT be marked!

Question	Your Grade	Max. Grade
1		10
2		10
3		15
4		15
5		20
6		10
7		10
8		10
Total		100

Question 1: (10 Points) Short-answer questions

Note: for (a) and (b), no explanation is needed, and no partial credit will be given for wrong answers.

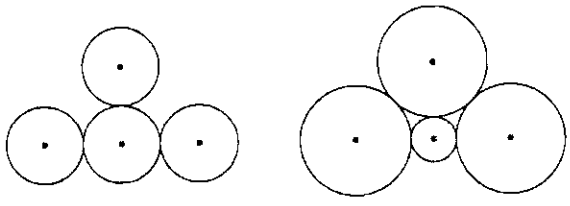
(a) Circle the correct answer: If a problem is NP-complete, it must not be in P. **TRUE FALSE UNKNOWN**

(b) Circle the correct answer: If a problem is not in P, it must be NP-complete. **TRUE FALSE UNKNOWN**

(c) Someone claims to have a polynomial-time algorithm for the HamCycle decision problem, but provided the number of vertices of the input graph is a prime number less than 1,000,000. Can you conclude that P=NP? Why or why not?

Question 2: (10 Points) Circle packing

Consider the following optimization problem. We are given as input **four points in the plane**, A, B, C, and D. We wish to place four circles, centered at each of these points, so that no two circles overlap each other and the average radius is maximized. The Figure below depicts two possible solutions for the points (0,0), (1,0), (-1,0), and (0,1); the solution on the right has larger average radius.

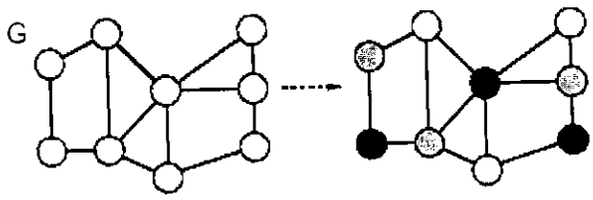


(a) Describe how to set up this average-radius-maximization problem as a linear program, in which the variables a, b, c, d represent the radii of the four circles. You do not need to use standard form or slack form, but do write out the objective function and all the inequalities used for your linear program. You may use $d(p, q)$ to denote the distance between points p and q.

(b) Let NN(x) denote the distance from x to its nearest neighbor. Observe that, in any feasible solution to the circle packing problem, each radius is at most NN(x). On the other hand, setting each radius to NN(x)/2 (as in the left Figure) produces a solution in which no two circles overlap. What can you say about the approximation ratio of this solution?

Question 3: (15 Points) Balanced 3-coloring

Given a graph $G = (V, E)$, where $|V|$ is a multiple of 3, G is said to have a balanced 3-coloring if we can assign a color from three possible colors (for e.g. Red, Green and Blue) to each vertex of G such that no two adjacent vertices have the same color, and such that all three colors are used equally (i.e. the sizes of the three color groups are all all equal to $|V|/3$). An example of such a graph is given in the Figure below.



The B2C decision problem is: given a graph G, determine if it contains a balanced 3-coloring.

Show that B2C is NP-complete using a reduction from the 3COL decision problem we discussed in class. As usual, your proof should consist of two parts: (1) B2C $\in$ NP, and (2) B2C is NP-Hard.

Question 4: (15 Points)

The goal of this problem is to explore the approximability of TSP when the graph's edge weights do NOT satisfy the triangle inequality. Recall that the TSP problem is: given a complete graph with weighted edges, find the cycle of minimum total cost that visits every vertex exactly once.

(a) Give an example of a complete graph (with as few as 4 vertices if you want, but no less) such that the 2-approximation algorithm given in class for the TSP problem results in a tour whose cost is **strictly more than** twice the cost of the optimum TSP tour. Show **all steps** of the approximation algorithm on your graph.

(b) Suppose that for some constant $q > 1$, there exists a polynomial-time, factor-q approximation algorithm for the TSP problem for graphs with **arbitrary** edge weights, i.e. not necessarily satisfying the triangle inequality. Give a polynomial-time algorithm that uses/calls this algorithm to solve the HamCycle decision problem we discussed in class. Briefly explain the running time of your algorithm.

Question 5: (20 Points) Fractional Knapsack vs. 0-1 Knapsack

The *0-1 Knapsack problem* is defined as follows: given a set of n items, where the weight of the ith item is w_i and its profit is p_i, find a maximum-profit subset of items whose total weight is **at most** K. Note that in this problem no fractions of items allowed into the selected subset; an item is in or out.

(a) Recall that the *Fractional Knapsack* problem can be solved using a greedy strategy. Give an example to demonstrate that this same strategy does not always yield an optimal solution for the 0-1 Knapsack (i.e. with actual values for the p_i's, w_i's and K).

(b) Show that 0-1 Knapsack is NP-complete.

Hint#1: Your first step should be to define a related decision problem and prove it is "at least not (polynomially) harder" to solve than 0-1 Knapsack. There are (at least) two decision problems related to 0-1 Knapsack. One of them is significantly easier to prove it is NP-complete (so choose wisely...).

Hint#2: Use reduction from SUBSET-SUM and consider relating the set $\{p_1, p_2, \cdots, p_n\}$ to the set X of integers in the SUBSET-SUM problem.

(c) Design an $O(nK)$ algorithm for the 0-1 knapsack problem. Your algorithm should output an *optimal* subset as well (there might be many optimal subsets, but we just want one of them).

Question 6: (10 Points)

Consider the following *decision* problem: given a set of n points in the plane: $P = \{p_1, p_2, \cdots, p_n\}$, we want to determine if there exists any 3 collinear points among them.

(a) Give a brute-force algorithm for solving this problem, and derive its running time. Hint: This should be almost trivial.

(b) Give a $O(n^2 \log n)$ time algorithm for this problem.

Hint#1: the running time should provide a partial hint on what the algorithm involves.

Hint#2: make use of polar angles...

Question 7: (10 Points)

Recall that the two-dimensional orientation primitive (operation) can be generalized to three-space as follows:

$$\text{Orient}(p,q,r,s) = \text{sgn}(\det \begin{pmatrix} 1 & p_x & p_y & p_z \\ 1 & q_x & q_y & q_z \\ 1 & r_x & r_y & r_z \\ 1 & s_x & s_y & s_z \end{pmatrix})$$

where sgn(.) is a unary function that returns the sign of its argument (which is assumed to be a number of course).

Consider now the following query in the plane: given three noncollinear points p, q, and r, determine whether a point s lies inside the circle determined by p, q, and r.

(a) Describe one way to solve this problem geometrically, but without using the `Orient` primitive. Then give a reason(s) for why this solution might not be desirable for implementation on a computer.

(b) Explain how to use the 3D `Orient` primitive to answer this same query. ***Hint:*** the trick is to determine the right value for the z-coordinates, such that the nonlinear planar circle condition is transformed to a linear condition in 3-space.

Question 8: (10 Points)

Suppose you are given a convex polygon P as a counterclockwise list of its n vertices. You are also given a point q in the plane. You may assume for simplicity that q does not lie on the boundary of P.

(a) Give a $O(n)$-time algorithm that determines whether q lies inside P.

(b) Give a $O(\log n)$-time algorithm that determines whether q lies inside P.

It always helps to illustrate your answer with a meaningful diagram (at least to get some partial-credit).

Hint: use orientation around point q... (in both (a) and (b)).