NATIONAL UNIVERSITY OF SINGAPORE

School of Computing

Final Examination for
Semester 1 AY2013/14

CS5234—Combinatorial and Graph Algorithms

November 19 2013 Time Allowed: 2 Hour

INSTRUCTIONS TO CANDIDATES

1. This exam contains **FOUR (4)** questions and comprises **FOURTEEN (14)** printed pages, including this page and four pages of scratch paper.
2. The exam is closed book. You may bring two double-sided sheet of A4 paper to the quiz. (You may not bring any magnification equipment!) You may not use a calculator, your mobile phone, or any other electronic device.
3. Write your solutions in the space provided. If you need more space, then use the scratch paper at the end of the exam.
4. Read through the problems before starting. Do not spend too much time on any one problem.
5. Show your work. Partial credit will be given.
6. Be neat. If I cannot read your solution, then I cannot give you credit for it.
7. You may use (unchanged) any algorithm or data structure given in class. You do not need to restate the algorithm. If you modify the algorithm, you must explain exactly how your version works.
8. Draw pictures frequently to illustrate your ideas.
9. Good luck!

MATRICULATION NUMBER: _____

Problem #	**Name**	**Possible Points**	**Achieved Points**
1	Warehouse Management	20	
2	Linear Programming	30	
3	MaxCut	24	
4	SecretNets	26	
Total:		100	

Problem 1. Warehouse Management [20 points]

You have been placed in charge of the delivery infrastructure for an on-line grocery store. Your job is to decide where to situate warehouses in order to minimize the costs of operating the warehouse and shipping the groceries to the customers. You have information on all your customers, including their location and the amount of goods that they want. Each customer must receive all the required groceries, and may receive shipments from one or more warehouses. In more detail, for this problem, you are given:

- A set of customers $c_1, \ldots, c_n$, where each c_i represents the location of customer c_i. (Each location is a set of coordinates in Euclidean space.)
- The demand d_i for each customer c_i. That is, customer c_i requires d_i kilograms of groceries.
- A set of possible locations to open warehouses $w_1, \ldots, w_k$. Each w_j repesents the location of warehouse site w_j.
- The cost v_j of operating each warehouse w_j. That is, to operate a warehouse at location w_j costs w_j dollars.
- The cost Z for shipping 1 kilogram of groceries 1 kilometer. That is, if you are shipping x kilograms of groceries to a customer that is y kilometers from the warehouse, then it costs xyZ dollars.

Your job is to formulate this problem as an integer linear program. (You do not have to solve the problem. You simply have to write the appropriate integer linear program.) Your integer linear program does not have to be in standard form, however it should only use constructs that we have already shown can be transformed into standard form.

Please write your answer on the next page.

Variables: *(For each variable, explain its intuitive meaning.)*

Objective:

Constraints: *(For each constraint, first describe in English the goal of the constraint. Then provide the mathematically precise constraint.)*

Problem 2. Linear Programs [30 points]

Problem 2.a. [15 points] Give all the vertices of the polytope associated with the following linear program, and find the point that maximizes the objective function.

$$\begin{array}{lrl} \text{maximize} & x_1 + 2x_2 & \\ \text{subject to} & x_1 + 3x_2 & \leq 6 \\ & -6x_1 - 3x_2 & \leq -3 \\ & x_1 + x_2 & \leq 4 \\ & x_1 & \geq 0 \\ & x_2 & \geq 0 \end{array}$$

Problem 2.b. [15 points] Find the dual of the following linear program:

$$\begin{array}{lll} \text{maximize} & 2x_1 + x_2 + 4x_3 + 12x_4 & \\ \text{subject to} & x_1 + 3x_2 & \leq 6 \\ & x_1 + 7x_3 + 4x_4 & \geq 9 \\ & x_1 + x_2 + x_3 + 4x_4 & = 13 \\ & x_1 & \geq 0 \\ & x_2 & \geq 0 \\ & x_3 & \geq 0 \\ & x_4 & \geq 0 \end{array}$$

Problem 3. Approximating MaxCut [24 points]

The problem of finding a minimum cut in a graph $G = (V, E)$ is easy to solve (e.g., using Ford-Fulkerson). By contrast, the problem of finding a maximum cut is NP-hard! In this problem, we will develop a 2-approximation algorithm for this problem. (A famous result by Goemans and Williamson uses a more complicated technique—semidefinite programming—to find a 0.878 approximation.)

Consider an unweighted, undirected graph $G = (V, E)$. The goal is to find a maximum cut, i.e., to partition the vertices V into two (disjoint) sets A and B where the number of edges crossing the cut is maximized.

Assume you are given a cut (A, B) where $A \subseteq V$ and $B \subseteq V$. For every node v, we define: $edges(v, A)$ to be the number of edges connecting v to a node in A, and $edges(v, B)$ to be the number of edges connecting v to a node in B.

The idea of our algorithm is to start with an arbitrary cut, and incrementally (greedily) improve it: (i) we search for a node in A with more edges connected to nodes in A than nodes in B, and move it to B; or (ii) we search for a node in B with more edges connected to nodes in B than nodes in A, and move it to A. We continue until we cannot improve the cut any more, and then return the resulting sets. The algorithm is defined as follows:

```
FindCut(V, E)
  A ← V                          ▷ Assign all the nodes to set A.
  B ← ∅                          ▷ B is empty.

  ▷ Search for a node v to move across the cut.
  while ∃v ∈ A where edges(v, A) > edges(v, B) or
        ∃v ∈ B where edges(v, B) > edges(v, A)
    do
       if v ∈ A                  ▷ Then move v to B.
          then  A ← A \ {v}
                B ← B ∪ {v}
       if v ∈ B                  ▷ Then move v to A.
          then  B ← B \ {v}
                A ← A ∪ {v}
  ▷ Return the cut.
  return (A, B)
```

The problem continues on the next page.

Problem 3.a. [12 points] Prove that the algorithm FindCut, described above, terminates in polynomial time.

(Hint: what is the largest possible cut that the algorithm discovers?)

Problem 3.b. [12 points] Prove that the algorithm FindCut, described above, finds a 2-approximation of the maximum cut.

(Hint: For each node, what is the fraction of edges that cross the cut?)

Problem 4. SecretNets Corporation Incorporated, Ltd. [26 points]

SecretNets Corp. has built an overlay network for tranporting data across the internet. The SecretNet consists of a directed graph $G = (V, E)$ with a set of entry nodes $e_1, e_2, \ldots, e_k$ and a set of exit nodes $x_1, x_2, \ldots, x_k$. SecretNets makes the following guarantee to its clients: "Your packets will be routed securely through the network, and they will never share a node or an edge in the network with a competitor." This ensures that there is no way that anyone can spy on their packets!

Problem 4.a. [8 points] The engineer who originally designed the SecretNet just quit. Your job, as the replacement, is to find the maximum number of clients that the SecretNet can support.

- Each client is assigned some entrypoint e_i and some exit point x_j. (Any combination of entry point and exit point is allowed.)
- Each client is assigned a path through the network from e_i to x_j.
- No two client paths intersect at any **nodes or edges**.

Give an efficient algorithm for finding the maximum number of clients that the SecretNetwork can support.

Problem 4.b. [18 points] It turns out that there are only d clients, at the moment, fewer than the maximum number that your network can support. Each link in the network has a cost to use it, and you want to find the d disjoint paths that are cheapest. That is:

- Assume each edge in the SecretNetwork $e \in E$ has a bandwidth cost $b(e)$.
- Your algorithm should output d paths $p_1, p_2, \ldots, p_d$ that are entirely disjoint, that is, they do not intersect at either the nodes or edges.
- The cost of a path p_j is the sum of the edge costs, i.e., $\sum_{(v,w)\in p_j} b(v,w)$.
- Your algorithm should minimize the total cost, i.e., $\sum_{j=1,\ldots,d} cost(p_j)$.

Give an efficient algorithm, and prove that it satisfies the requirements.

(For partial credit, you may instead give an algorithm that finds client paths that do not intersect at any ***edge*** *in the SecretNetwork.)*

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

End of Exam