This page is intentionally left blank.
Please do not open this question set before you are allowed to do so.

4810-1183 Approximation and Online Algorithms with Application (Spring 2017) #1

Final Problem 1

Consider the following situation.

> In k-center problem, we have a set of positions for n houses. Out of that n houses, we will pick to install ward offices at k houses, and minimize the longest commute time from houses to ward offices. In our discussion, we assume that the commute time is proportional to the Euclidean distance, i.e. if the Euclidean distance is d the commute time is $c \cdot d$ for some constant c. For this problem, we will consider the setting where we have some disabilities in our city. Instead of $c \cdot d$ like others, the disabilities will take $2 \cdot c \cdot d$ to get to a ward office with Euclidean distance d. With those disabilities, we still want to minimize the longest commute time from houses to ward offices.

We will call this problem as *k-center with disabilities*.

Question 1.1: State inputs of this problem by a mathematical formulation.

Question 1.2: State outputs of this problem by a mathematical formulation.

Question 1.3: State constraints of this problem by a mathematical formulation.

Question 1.4: State objective functions of this problem by a mathematical formulation.

Question 1.5: State constraints of *1.999-approx k-center with disabilities*.

Question 1.6: Write a program for solving *1.999-approx k-center* based on the assumption that an efficient algorithm for solving the *1.999-approx k-center with disabilities* problem is given in a library.

```
[AnswerOf1_2] 1.999ApproxKCenterwithDisabilities([AnswerOf1_1]);

Set 1.999ApproxKCenter(Point[] p){
     //write your code for knapsack here
}
```

Question 1.7: Based on your answer of Question 1.6, discuss why it is not possible to find a 1.999-approximation algorithm for the *k-center with disabilities* problem.

4810-1183 Approximation and Online Algorithms with Application (Spring 2017) #1

Final Problem 2

We will continue working on your optimization model. In this problem, we will devise an approximation algorithm for the k-*center with disabilities*.

Question 2.1: Construct an input where an optimal output for k-*center* problem is different from an optimal output for k-*center with disabilities* problem.

Let assume the following notation.

S: an output of 2-approximation algorithm for k-*center* problem
SOL: an objective value obtained from applying S to the objective function of k-*center* problem
SOL_D: an objective value obtained from applying S the objective function of k-*center with disabilities* problem
OPT: an optimal value of k-*center* problem
OPT_D: an optimal value of k-*center with disabilities* problem

Question 2.2: Discuss why $SOL_D \geq SOL$

Question 2.3: Discuss why $OPT_D \leq 2 \cdot OPT$

Question 2.4: Discuss why a 2-approximation algorithm for the k-*center* problem is a 4-approximation algorithm for the k-*center with disabilities* problem.

Question 2.5: Give a brief idea how to have a 2-approximation algorithm for k-*center with disabilities* problem.

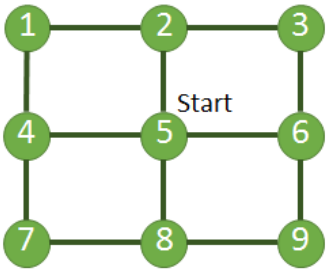
Question 2.6: Discuss how the solution for k-*center with disabilities* problem can be used in machine learning application.

4810-1183 Approximation and Online Algorithms with Application (Spring 2017) #1

Final Problem 3

In this problem, we will derive an online algorithm for a path finding problem.

Suppose that each dot in the following map is a place that you can try finding a treasure. Two places are linked together by a line, if you can move between those places. We begin finding a treasure at the middle of the map, and we want to minimize the number of moves here.



Question 3.1: If we know that the treasure is at Point 3, how many number of moves we need to get the treasure?

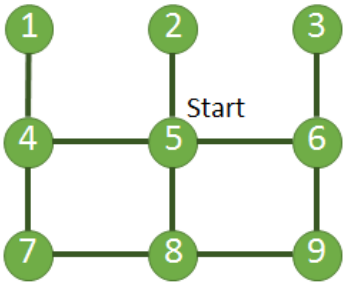
From next question, assume that there is only one treasure in the map. We know the map, but we do not know where the treasure is.

Question 3.2: How many moves do we need to find the treasure at all points? How do we move to have that number of moves? How many moves until we can find a treasure at Points 2, 4, 6, 8?

Question 3.3: What is the competitive ratio of your strategy in Question 3.2?

Question 3.4: Discuss why there is no strategy of which the competitive ratio is less than 7.

Next, consider the following map, where one link is missing.



Question 3.5: Find a 7-competitive move strategy for the above map.

Question 3.6: Discuss why there is no strategy of which the competitive ratio is less than 7 for the above map. Use the fact that there is no strategy of which the competitive ratio is less than 7 on the first map.