4810-1183 Approximation and Online Algorithms with Applications
Lecture Note 6: Inapproximability

Introduction

When we work on approximation algorithm, we want to find an algorithm of which the approximation ratio is close to 1 (0.99999 for maximization problem and 1.00001 for minimization problem). However, for the vertex cover problem, we have a 2-approximation algorithm. Other researchers might have questions if you can have a better approximation ratio for the problem. They might try to beat you by finding a 1.9999-approximation algorithm for the problem. However, it might be impossible to have that. Those researchers may waste time, but cannot find anything.

In Lecture 2, we discuss "NP-hardness", which how to prove that a problem is not solvable. We will use the concept to formally show that 1.9999-approximation algorithm for vertex cover is not possible in this lecture note.

Reformulating Problems

Recall that the formulation of the vertex cover problem is as follows:

Input: Set V, Set $E \subseteq \{\{u, v\}: u, v \in V\}$

Output: $S \subseteq V$

Constraint: For all $\{u, v\} \in E$, $u \in S$ or $v \in S$

Objective Function: Minimize $|S|$

We want to find a valid set S that is smaller than any valid set S'. In other word, if S is an optimal solution of the optimization model, $|S| \leq |S'|$ for all S' that satisfies the constraint. We can replace the objective function with the following constraint.

Input: Set V, Set $E \subseteq \{\{u, v\}: u, v \in V\}$

Output: $S \subseteq V$

Constraint: For all $\{u, v\} \in E$, $u \in S$ or $v \in S$.
For all S' such that, for all $\{u, v\} \in E$, $u \in S$ or $v \in S$, we have $|S| \leq |S'|$

Objective Function: None

All solutions of the new problem are optimal solutions of vertex cover. When we have an 1.9999-approximation algorithm for vertex cover, it means that, from the algorithm, we have S such that $|S| \leq 1.9999 \cdot OPT$. Because OPT is the smallest size of all the valid S', we know that the above inequality is equivalent to $|S| \leq 1.9999 \cdot |S'|$ for all valid S'.

As a conclusion of the discussion in the previous paragraph, if we have an 1.9999-approximation algorithm for vertex cover, we can solve the following optimization model. On the other hand, if we can solve the optimization model, we will have an 1.9999-approximation algorithm. We will call the model as *1.9999-approximation vertex cover problem*.

Input: Set V, Set $E \subseteq \{\{u, v\}: u, v \in V\}$

Output: $S \subseteq V$

Constraint: For all $\{u, v\} \in E$, $u \in S$ or $v \in S$.
For all S' such that, for all $\{u, v\} \in E$, $u \in S$ or $v \in S$,
we have $|S| \leq 1.9999 \cdot |S'|$

Objective Function: None

Having the 1.9999-approximation algorithm is *as hard as* solving the 1.9999-approximation vertex cover problem. If the solving the problem is not possible, then having the 1.9999-approximation algorithm is also not possible. To show that solving the problem is not possible, we can prove that the problem is NP-hard.

Inapproximability of the Vertex Cover Problem

Unfortunately, no one can prove that the 1.9999-approximation vertex cover problem is NP-hard. On the other hand, no one can find a 1.9999-approximation algorithm for the vertex cover problem. Because of that, the smallest approximation ratio is still an open problem.

Recall that we believe that NP-hard problems are not solvable based on our belief that SAT is not solvable. There is some people believe that, not only SAT is not solvable, but some problems easier than SAT are also not solvable. Two of the most well-known examples are called as unique games conjecture, which believe that a problem called unique game is not solvable, and exponential-time hypothesis, which believe that we cannot solve SAT in $O(1.9999^n)$. We know that a 1.9999-approximation algorithm for the vertex cover is not possible under the unique games conjecture.

Warm up

When a 1.9999-approximation algorithm for vertex cover is not possible, we know that 1.5-approximation algorithm is also not possible. However, let try to show that systematically.

We know that having a 1.9999-approximation algorithm is equivalent to solving the 1.9999-approximation vertex cover problem. Using the same argument, we know that having a 1.5-approximation algorithm is equivalent to the following model, called as 1.5-approximation vertex cover problem.

Input: Set V, Set $E \subseteq \{\{u, v\} : u, v \in V\}$

Output: $S \subseteq V$

Constraint: For all $\{u, v\} \in E$, $u \in S$ or $v \in S$.
For all S' such that, for all $\{u, v\} \in E$, $u \in S$ or $v \in S$,
we have $|S| \leq 1.5 \cdot |S'|$

Objective Function: None

We will prove that the 1.5-approximation vertex cover is not solvable, based on the fact that 1.9999-approximation vertex cover is not solvable. In other words, we will show that the 1.5-approximation vertex cover is harder than the 1.9999-approximation vertex cover.

To do that, we will assume that there is an algorithm for the 1.5-approximation vertex cover provided in a library. Then, we will write a program for the 1.9999-approximation vertex cover. That is:

```
Set 1.5ApproxVertexCover(Set V, Set E)

Set 1.9999ApproxVertexCover(Set V, Set E){
      return 1.5ApproxVertexCover(V, E)
}
```

Let us discuss why the above program for `1.9999ApproxVertexCover` is valid. From the function `1.5ApproxVertexCover`, we have the output S such that $|S| \leq 1.5|S'|$ for all valid S'. That means S is a set such that $|S| \leq 1.5|S'| \leq 1.999|S'|$ for all valid S', and S is also a solution for `1.9999ApproxVertexCover`.

We can solve `1.9999ApproxVertexCover` using `1.5ApproxVertexCover`. `1.5ApproxVertexCover` is harder than `1.9999ApproxVertexCover`. When the 1.9999-approximation vertex cover problem is not solvable, the 1.5-approximation vertex cover problem is also not solvable.

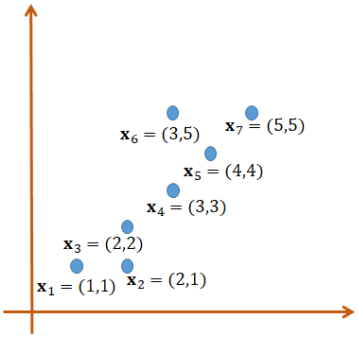
k-Center Problem

Called as "clustering", dividing a set of information to groups are one of basic problems in machine learning. However, the word "clustering" is not formally defined. There are many works trying to formalize clustering into optimization models. Two of the most well-known formulation are "k-means" and "k-center". We will focus on the k-center, which is formulated below, in this lecture note.

Input:	Number of data points n For all i, j, distance between i and j denoted by $d(i,j)$ Assumption: for all i, j, j', $d(i,j) \leq d(i,j') + d(j',j)$ Number of groups to find k
Output:	Positions of centers for each group $S \subseteq \{1, \dots, n\}$
Constraint:	$\lvert S \rvert = k$
Objective Function:	Distance to closest center of $\mathbf{x}_i$ is $d_i = \min_{j \in S} d(i,j)$ Minimize $\max_i d_i$

To understand the formulation, let us consider the case that we have n houses in our city. Their locations are $\mathbf{x}_1, \dots, \mathbf{x}_n$. We want to set up k ward offices in the city, and we want to minimize the commuting time from houses to ward office. We believe that each house will commute to the closest office, and their commuting distance would be d_i. The most suffering house is going to be a house with the largest d_i, i.e. a house with commuting time equals $\max_i d_i$. We want to having the most suffering house has the least suffer as possible, so we minimize $\max_i d_i$.

Houses will be put to the same group if they use the same ward office. Consider a city with 7 houses. The locations of the houses is shown below. Suppose that we set the ward offices at position $S = \{3,5\}$. All houses can commute to ward offices within distance $\sqrt{2}$. We know that $\{3,5\}$ is the optimal solution for this input. Houses that will go to the ward office located at $\mathbf{x}_3$ is 1,2,3, while houses that will go to the ward office located at $\mathbf{x}_5$ is 4, 5, 6, 7. By that, we can divide the houses (data) into 2 groups $\{1,2,3\}$ and $\{4,5,6,7\}$.



There is a 2-approximation algorithm for the k-center formulation [1]. The algorithm is based on the greedy scheme that we discussed in lecture note 3. In this lecture note, we will show that a 1.9999-approximation algorithm is not possible. That is equivalent to showing that the following optimization model is not solvable.

Input:	Number of data points n For all $i, j \in \{1, \dots, n\}$, distance between i and j denoted by d Assumption: For all i, j, j', $d(i,j) \leq d(i,j') + d(j',j)$ Number of groups to find k
Output:	Positions of centers for each group $S \subseteq \{1, \dots, n\}$
Constraint:	$\lvert S \rvert = k$ For any set S, suppose that $D_S \coloneqq \max_{i \in \{1,\dots,n\}} \min_{j \in S'} \left\lvert\left\lvert \mathbf{x}_i - \mathbf{x}_j \right\rvert\right\rvert$. For any S' such that $\lvert S' \rvert = k$, $D_S \leq 1.9999 \cdot D_{S'}$
Objective Function:	None

We will call the problem as 1.9999-approximation k-center problem. From now, we will prove that the problem is NP-hard. The proof will based on the fact that the dominating set, defined in the following section, is NP-hard.

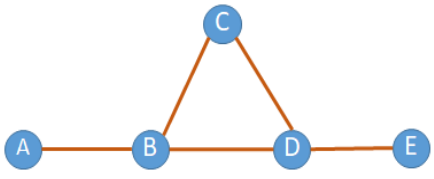
Dominating Set Problem

We will turn back to a problem on social networks. Previously, in the vertex cover problem, we choose a set of persons to cover all communications (friendships). In dominating set problem, we want to cover all persons. In other words, every persons must be in the set or next to someone in the set.

Consider a social network with a set of persons V and a set of friendships $E \subseteq \{\{u, v\} : u, v \in V\}$. A set $S \subseteq V$ is a dominating set of the social network, if all $v \in V$ satisfy one of the following conditions:

1. $v \in S$
2. There is u such that $u \in S$ and $\{u, v\} \in E$.

In the below social network, $\{A, E\}$ is not a dominating set, as C is not in the set and $\{A, E\}$ do not have a friendship with C. On the other hand, $\{B, D\}$ is a dominating set, as $\{B, D\}$ are friends of everyone in the social network.



The formulation of the dominating set problem is as follows. The dominating set problem is one of the most NP-hard problem.

Input:	Set V, Set $E \subseteq \{\{u, v\} : u, v \in V\}$ Positive integer k
Output:	Yes or No
Constraint:	Yes, if there is a dominating set S with $\lvert S \rvert \leq k$. No, otherwise

Objective Function: None

Inapproximability of the k-center Problem

Now, we are ready to prove that the 1.9999-approximation k-center problem is NP-hard. We will prove that the problem is harder than the dominating set problem, which is NP-hard. Thus, we will assume that a program for the 1.9999-approximation k-center problem is provided in a library, and we will write a program for the dominating set problem based on that.

A program for the dominating set problem is as follows:

```
Set 1.9999ApproxKCenter(int n, double[][] d, int k);

boolean dominatingSet(Set V, Set E, k){
      Suppose that V = {1, …, n}
      Let d be a 2-dimenstional array with size n × n.
      d[i][j] = { 0  when i = j
                { 1  when {i, j} ∈ E
                { 2  otherwise.

      S' = 1.9999ApproxKCenter(n, d, k)
      If D_S' ≤ 1:   return Yes
      Else:          return No

}
```

When there exists a dominating set S with $|S| = k$, all persons are in S or have friends in S. By the definition of $d[i][j]$ in the program, for all persons $i \in V$, there always exists person $j \in S$ such that $d[i][j] = 0$ or $d[i][j] = 1$. As we use d as a distance in the k-center problem, if we set up ward offices at all houses in S, we know that the distance from i to the closest ward office ($j \in S$) would not be more than 1. We have $D_S = 1$.

We know that the output of `1.9999ApproxKCenter`, denoted by S', has $D_{S'} \leq 1.9999 \cdot D_S = 1.9999$. As, for this input, $D_{S'}$ can be only 1 and 2, when $D_{S'} \leq 1.9999$, we know that $D_{S'} = 1$. The output of `dominatingSet` is Yes as it should be.

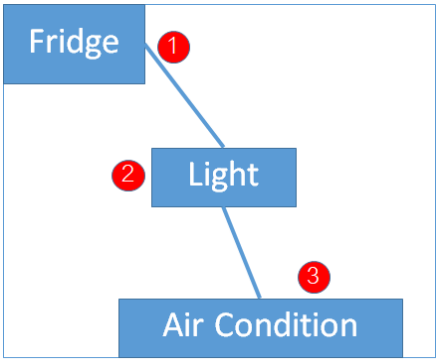
When there does not exist a dominating set S with $|S| = k$, by the same argument, we will know that all S has $D_S = 2$. The output of `1.9999ApproxKCenter`, denoted by S' is then has $D_{S'} = 2$. The output of `dominatingSet` is Yes as it should be.

RG-TOSS Problem

In this section, we will consider the robustness guaranteed task-optimized group search (RG-TOSS) problem. The problem is proposed in [2].

The authors consider "social networks of the Internet of Things (IoTs)". Mathematically, it is very similar to ordinary social networks we have discussed so far. The only difference is the set V is not a set of persons, but a set of "anything" (e.g. lights, air conditions, fridges). We have $\{u, v\} \in E$ when u and v can communicate using any type of networks (4G, Wi-fi).

Let us consider the social network in the following page. In a smart house, we have sensors at the following 3 furniture, fridge, light, and air condition. To precisely set the power of the air condition, we want to measure the temperature at three red points. The sensor attached at the fridge can guess the temperature at point 1 very well, and may be able to measure the temperature at point 2. Suppose that the probability that the sensor can have the temperature from point 1 is 0.9 and from point 2 is 0.5.



Suppose that we would like to have the temperature information only at point 1. We cannot collect the information at all of the sensors as the data collection is costly. We can collect from the sensor at fridge, then we can collect the information with probability 0.9. However, it is possible that the information is incorrect. The sensor wants to make sure that what they have got is correct with their peers, which are sensors that can directly communicate with them. If we select more sensors around them, the information will be rechecked more, and we will have a robust information.

We will require each selected sensor to be friends of at least k selected sensors. For example, in the above sensor network, we cannot select only the sensor at the fridge but we have to also include the sensor at the light.

By the discussion, our optimization model, is as follows:

Input: Set of sensors V, Set of friendships $E \subseteq \{\{u, v\}: u, v \in V\}$
Set of measurement point T
For all $t \in T$ and $v \in V$, the probability that we have information t by the sensor $v \in V$, denoted by $p(t, v)$
Robustness parameter $k \in \mathbb{Z}_{>0}$, budget $s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: For all $v \in S$, $|\{\{u, v\} \in E: u \in S\}| \geq k$, $|S| = s$

Objective Function: Maximize $\sum_t P(t)$ when $P(t) \coloneqq \sum_{v \in S} p(t, v)$ is a probability of having the information at the measurement point t

The optimization model is clearly NP-hard, even to find an output that satisfies the constraint. Let us now consider the above model when we do not have the objective function. Without the objective function, we do not need to have T and $p(t, v)$ as our inputs anymore. The optimization model is as follows:

Input: Set of sensors V, Set of friendships $E \subseteq \{\{u, v\}: u, v \in V\}$
Robustness parameter $k \in \mathbb{Z}_{>0}$, budget $s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: For all $v \in S$, $|\{\{u, v\} \in E: u \in S\}| \geq k$, $|S| = s$

Objective Function: None

We will call the optimization model as RG-TOSS', and will prove that the problem is NP-hard. Previously, we introduce you the clique problem. Recall that the optimization model of the problem is as follows:

Input: Set V, Set $E \subseteq \{\{u,v\}: u,v \in V\}$
$s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: For all $u,v \in S, \{u,v\} \in E, |S| = s$

Objective Function: None

The input, output, and objective function of the clique problem. The only different is the constraint. Let us consider the case when $k = s - 1$. We have $|\{\{u,v\} \in E, u \in S\}| \geq k - 1$. As $|S| = k$, when we have the constraint, we know that all $u \in S - \{v\}$ are friends of v. We know that, for all $u,v \in S$, $\{u,v\} \in E$, which is a constraint of the clique problem. To conclude, we know that RG-TOSS' is equivalent to the clique problem when $k = s - 1$. We can write a program for the clique problem based on a library for RG-TOSS' as follows:

```
Set RG-TOSS'(Set V, Set E, int k, int s)

Set clique(Set V, Set E, int s){
      RG-TOSS'(V, E, s-1, s)
}
```

By the above program, we know that RG-TOSS' is NP-hard. Even finding a solution is NP-hard, it is straightforward to show that find a solution that optimize an objective function is also NP-hard.

When it comes to approximation algorithms, things will become more complicated. Previously, algorithms can find a solution just it is not an optimal, so we can prove that $SOL \leq OPT$. We cannot even find a solution now. The definition of approximation algorithm cannot apply here, because SOL is not very well defined. The authors cope with the problem with the following reformulation.

Input: Set V, Set $E \subseteq \{\{u,v\}: u,v \in V\}$
Set T, for all $t \in T$ and $v \in V$, probability $p(t,v)$
$k,s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: $|S| = s$

Objective Function: Maximize L
$L = 0$ when $|\{\{u,v\} \in E: u \in S\}| < k$.
Otherwise, $L = \sum_t P(t)$ when $P(t) \coloneqq \sum_{v \in S} p(t,v)$

Personally, I am skeptical on the reformulation, but let us try to discuss the inapproximability of the above problem, which we will call as RG-TOSS''.

The authors of [2] show that, for RG-TOSS'', it is not possible to have α-approximation algorithm for any $\alpha > 0$. To prove that, we have to reformulate the problem. Let L_S be the value of L we will have for a set S. We will reformulate RG-TOSS'' to the optimization model.

Input: Set V, Set $E \subseteq \{\{u,v\}: u,v \in V\}$
Set T, for all $t \in T$ and $v \in V$, probability $p(t,v)$
$k,s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: $|S| = s$
$L_S \leq L_{S'}$ for all $S' \subseteq V$ such that $|S'| = s$.

Objective Function: None

Having α-approximation algorithm is as hard as solving the following optimization model.

Input: Set V, Set $E \subseteq \{\{u, v\}: u, v \in V\}$
Set T, for all $t \in T$ and $v \in V$, probability $p(t, v)$
$k, s \in \mathbb{Z}_{>0}$

Output: $S \subseteq V$

Constraint: $|S| = s$
$L_S \leq \alpha \cdot L_{S'}$ for all $S' \subseteq V$ such that $|S'| = s$.

Objective Function: None

By the definition of L_S, when $s = k - 1$, $L_S = 0$ for all non-clique S. When S is a solution of the clique problem, we have $L_S > 0$. By the following program, we will prove that above optimization model, called as α-approximation RG-TOSS'' is NP-hard based on the fact that the clique problem is NP-hard.

```
Set alpha-approx-RG-TOSS''(Set V, Set E, int k, int s)

Set clique(Set V, Set E, int s){
      return alpha-approx-RG-TOSS''(V, E, s-1, s)
}
```

Suppose that there exists a clique S' with $|S'| = s$. We know that there is some S' such that $|S'| = s$ and $L_{S'} > 0$. Let us assume that the output of `alpha-approx-RG-TOSS''` is S. By the constraint of the α-approximation RG-TOSS'', we have $L_S \geq \alpha \cdot L_{S'}$. As $\alpha > 0$ and $L_{S'} > 0$, $L_S > 0$. S is a clique and the function `clique` runs correctly.

We then know that it is not possible to have any approximation algorithm for the RG-TOSS'' problem.

Exercises

Consider the following optimization model:

Input: $\mathcal{A}$

Output: $\mathcal{B}$

Constraint: $\mathcal{C}$

Objective Function: Maximize $\mathcal{D}$

We assume that, for any output that satisfies the constraint $\mathcal{C}$, the objective value $\mathcal{D}$ is non-negative. Let us call the above optimization model as **Problem 1**.

Question 1: Define an optimization model that is as hard as having a 0.5-approximation algorithm for Problem 1.

From now, we will assume that it is not possible to have a 0.5-approximation algorithm for Problem 1. Let us recall an optimization model with the same input, output, and constraint, but with the following objective function.

Objective Function: Minimize $1/\mathcal{D}$

We will call the optimization model as **Problem 2**.

Question 2: Define an optimization model that is as hard as having a 2-approximation algorithm for Problem 2.

Question 3: Assume that there is a problem for your optimization model in Question 2. Write a program to solve an optimization model in Question 1. Discuss why your program works.

Question 4: Based on your answer in Question 3, discuss why it is not possible to have a 2-approximation algorithm for Problem 2.

From the following question, let us assume that, for any output that satisfies the constraint $\mathcal{C}$, the objective value $\mathcal{D}$ is between 0 and 1. Consider **Problem 3** with the same input, output, and objective function, but with the following objective function.

Objective Function: Minimize $1 - D$

Question 5: Assume that we have a 2-approximation algorithm for problem 3. If we use that algorithm for Problem 1, what are the guarantees on the outputs' objective values when the optimal value (of Problem 1) are 0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2.

Question 6: Based on your answer in Question 5, discuss why knowing that "an α-approximation algorithm is not possible for problem 1" does not help us showing that "an α'-approximation algorithm is not possible for problem 3".

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 7: State inputs of this problem by a mathematical formulation.

Question 8: State outputs of this problem by a mathematical formulation.

Question 9: State constraints of this problem by a mathematical formulation.

Question 10: State objective functions of this problem by a mathematical formulation.

Question 11: State constraints of *1.999-approx k-center with disabilities*.

Question 12: 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.

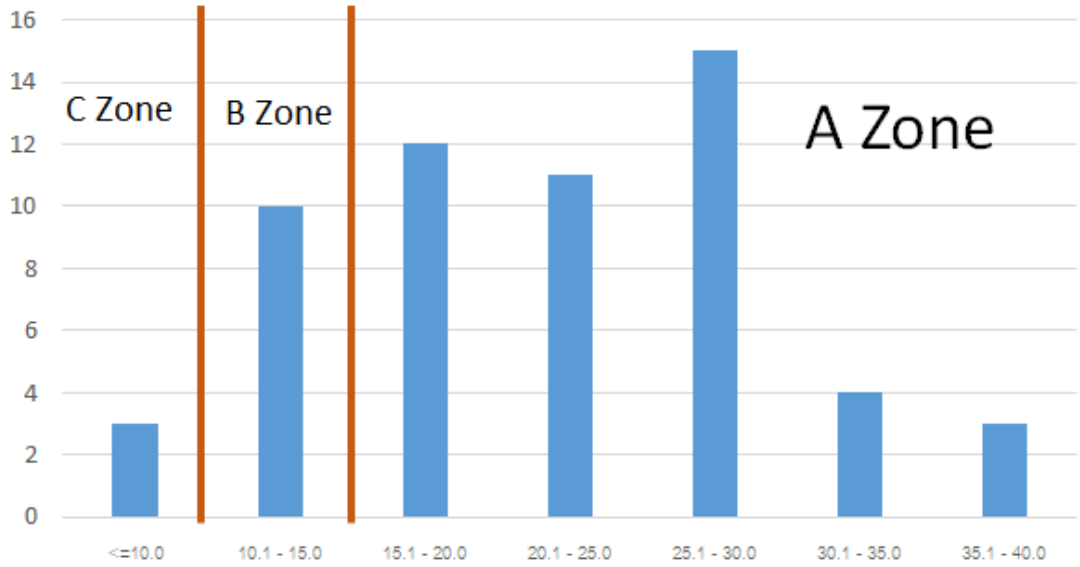
```
[AnswerOf1_2] 1.999ApproxKCenterwithDisabilities([AnswerOf1_1]);

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

Question 13: Based on your answer of Question 6, discuss why it is not possible to find a 1.999-approximation algorithm for the *k-center with disabilities* problem.

Reference

[1] D. P. Williamson and D. Shmoys, "*The design of approximation algorithms*", Cambridge University Press, 2011. Chapter 2.2

[2] C. Y. Shen, H. H. Shuai, K. F. Hsu, and M. S. Chen, "Task-Optimized Group Search for Social Internet of Things", Proceedings of the 20th International Conference on Extending Database Technology (EDBT'17), pages 108-119, 2017.



Important Points

1. The zones above are to give you an idea how competitive this class will be. They will have nothing to do with your final grade. At one of my classes, everyone who were in B and C Zone at the midterm examination have got A from this class.
2. To obtain credits from this class, you must attend the final examination on July 16th.
3. After the final examination, those who do not get enough scores for the credits will be asked to submit a report.