CS300 Homework #8

Total 100 points Due: 2020-06-11 23:59:00 KST

- Write in English or Korean.
- Make sure your writing is readable before upload it (especially for scanned image).

1. Knapsack Problems (40 pts)

1.1. Three Knapsack Problems (20 pts)

You are required to solve three problems: 0-1 knapsack, Fractional knapsack, 0-1 knapsack with repetition. For a knapsack problem, you have a knapsack with limited capacity and several items with its own value and weight. Your goal is to maximize the value of the items in the knapsack where the total weight of items in the knapsack should be less than or equal to the capacity of the knapsack.
Suppose you have 4 items. Their values and weights are given in the following table.

	item 1	item 2	item 3	item 4
Value	2	4	3	9
Weight	1	3	2	4

Your goal is to fill in all the blanks of table below for each problem.

Usable	Maximum Weight								
Items	0	1	2	3	4	5	6	7	8
[ ]	**A**								
[1]		**B**							
[1,2]									
[1,2,3]									
[1,2,3,4]									

In the table, there are maximum weight, usable items, and many blanks. Maximum weight means capacity of the knapsack. Usable Items is a list that contains all indices of usable items. Each

blank should be filled with maximum value under the constraints of each problem. Maximum value means the sum of the values of the items in the knapsack with consideration of the amount of each item.
In detail, the equation below shows the definition of Maximum value:

$$\begin{aligned}&\text{MaximumValue(Usable Items, Maximum Weight)}\\&\qquad = \max\Big(\sum_{i\,\in \text{Usable Items}} (\text{value of } item_i) * (\text{the number of } item_i \text{ in the knapsack})\Big)\end{aligned}$$

$$\text{where} \sum_{i\,\in \text{Usable Items}} (\text{weight of } item_i) * (\text{the number of } item_i \text{ in the knapsack}) \leq \text{Maximum Weight}$$

For instance, for the cell A where Usable Items = [ ] and Maximum Weight = 0, the maximum value must be 0 because we cannot use any items and the knapsack cannot contain anything. For the cell B where Usable Items = [1] and Maximum Weight = 1, we can use item 1 whose value = 2 and weight = 1, and the knapsack can contain items where their total weight is less than or equal to 1. Therefore, Maximum Value for the cell B is 2 with one unit of item 1.

1. Fill the following table as fractional knapsack problem.

   Fill in the table below. In fractional knapsack problem, you can take fraction of an item.

   [**Example**] For given Maximum Weight and Usable Items, if you think that you should put item 2,3,4 with its fraction 1.0, 0.5, 0.5 in the knapsack to get maximum value, then Maximum value is the sum of the values of item 2,3,4 regarding its fraction. Therefore, in this case,

$$\text{MaximumValue} = \sum_{i=1}^{4} (value_i * fraction_i) = 2 * 0.0 + 4 * 1.0 + 3 * 0.5 + 9 * 0.5 = 10.0$$

   where $value_i$ is the value of item i, and $fraction_i$ is the fraction of the item i in the knapsack ($0 \leq fraction_i \leq 1$).

Usable	Maximum Weight								
Items	0	1	2	3	4	5	6	7	8
[ ]									
[1]									
[1,2]									
[1,2,3]									
[1,2,3,4]									

   Write down how you derive each cell briefly. You don't have to prove the correctness.

2. Fill the following table as 0-1 knapsack problem with **repetition**.

   Fill in the table below. In 0-1 knapsack problem with repetition, you are allowed to use the same item many times. However, you are not allowed to take fraction of an item. Therefore, the number of each item should be natural number including zero.

   [**Example**] For given Maximum Weight and Usable Items, if you think that you should put one unit of item 2 and two units of item 3 in the knapsack to get maximum value, then Maximum value is the sum of the value of item 2,3 regarding the number of each item.

   Therefore, in this case,

$$\text{MaximumValue} = \sum_{i=1}^{4} (value_i * number_i) = 2 * 0 + 4 * 1 + 3 * 2 + 9 * 0 = 10$$

   where $value_i$ is the value of item i, and $number_i$ is the number of the item i in the knapsack.

Usable	Maximum Weight								
Items	0	1	2	3	4	5	6	7	8
[ ]									
[1]									
[1,2]									
[1,2,3]									
[1,2,3,4]									

   Write down how you derive each cell briefly. You don't have to prove the correctness.

3. Fill the following table as 0-1 knapsack problem with **no repetition**.

   Fill in the table below. In 0-1 knapsack problem with no repetition, you are not allowed to use one item many times. You are not able to take fraction of an item neither. Therefore, the number of each item should be 0 or 1. In this case, maximum value is the sum of the values of all items in the knapsack.

   [**Example**] For given Maximum Weight and Usable Items, if you think that you should put item 2, 3, 4 in the knapsack to get maximum value, then Maximum Value is the sum of the values of items in the knapsack.

   Therefore, in this case,

$$\text{MaximumValue} = \sum_{i=1}^{4} (value_i * number_i) = 2*0+4*1+3*1+9*1 = 16$$

   where $value_i$ is the value of item i, and $number_i$ is the number of the item i in the knapsack.

Usable	Maximum Weight								
Items	0	1	2	3	4	5	6	7	8
[ ]									
[1]									
[1,2]									
[1,2,3]									
[1,2,3,4]									

   Write down how you derive each cell briefly. You don't have to prove the correctness.

1.2. Greedy Approach (20 pts)

1. (16 pts) For given capacity of knapsack W and n items $\{i_1, i_2, \ldots, i_n\}$ with its own value $\{v_1, v_2, \ldots, v_n\}$ and weight $\{w_1, w_2, \ldots, w_n\}$, find a greedy algorithm that solves fractional knapsack problem, and prove its correctness.

2. (4 pts) If you naively use the greedy algorithm in problem 1.2.1 to solve 0-1 knapsack problem with no repetition, then the greedy algorithm does not ensure an optimal solution anymore. Give an example that a solution from the greedy algorithm is not an optimal solution for 0-1 knapsack problem.

2. Shortest Path - Floyd Warshall (60 pts)

Floyd-Warshall algorithm is an algorithm for finding shortest paths between any pairs of two nodes in a weighted graph with positive or negative edge weights. For a graph with nodes $\{1, 2, \ldots, n\}$, if we define $l(i, j)$ as the length(i.e., weight) between adjacent nodes i and j and $dist(i, j, k)$ as the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediate nodes, then the Floyd-Warshall algorithm can be implemented as below:

Algorithm 1 Floyd-Warshall Algorithm

```
procedure FloydWarshall($n, E$)
    for $i = 1$ to $n$ do
        for $j = 1$ to $n$ do
            $dist(i, j, 0) = \infty$
    for all $(i, j) \in E$ do
        $dist(i, j, 0) = l(i, j)$
    for $i = 1$ to $n$ do
        $dist(i, i, 0) = 0$
    for $k = 1$ to $n$ do
        for $i = 1$ to $n$ do
            for $j = 1$ to $n$ do
                $dist(i, j, k) = \min\{dist(i, k, k-1) + dist(k, j, k-1), dist(i, j, k-1)\}$
    return $dist$
```

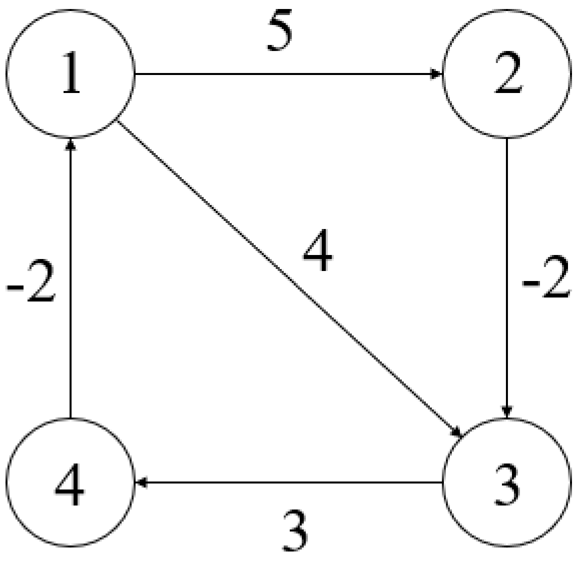

Figure 1: Directed graph

1. (12 pts) You are given an directed graph as Fig.1. Fill in the table *dist* which is the result after applying the Floyd-Warshall algorithm on the given graph.

k=0		**j**			
		1	**2**	**3**	**4**
i	**1**	0	5	4	∞
	2	∞	0	-2	∞
	3	∞	∞	0	3
	4	-2	∞	∞	0

k=1		**j**			
		1	**2**	**3**	**4**
i	**1**				
	2				
	3				
	4				

k=2		**j**			
		1	**2**	**3**	**4**
i	**1**				
	2				
	3				
	4				

k=3		**j**			
		1	**2**	**3**	**4**
i	**1**				
	2				
	3				
	4				

k=4		**j**			
		1	**2**	**3**	**4**
i	**1**				
	2				
	3				
	4				

2. (18 pts) You may not find the correct shortest path lengths if the graph has a negative cycle (i.e., a cycle whose edges sum to a negative value). Suppose you have a filled table *dist* as a result of FloydWarshall(). Using the table, can you detect whether there is a negative cycle or not? If so, give a method, or if not, explain why it is impossible.

3. (30 pts) The table *dist* has only lengths of each pair's shortest path. Given a pair of nodes i and j, give an algorithm to reconstruct the shortest path from i to j, and prove its correctness. Assume that you already have a filled table $dist(i, j, k)$ for all i, j, k.