CS 372/469 – Spring 2022

Alternate Lab 5

Due: 05/03/2022 11:59 pm

For each of the following questions, write a successful running code in any programming language that you prefer. Your code should run without any errors for any *valid* input. **The total grade for this alternate lab is 50% of the Lab 5. Submit this only if you would like to replace your existing Lab 5 grade.**

All problems are borrowed from <https://www.techiedelight.com/graphs-interview-questions/>

**Question 1 (40 points):**

Given a directed graph (nodes and edges), find out if a vertex j is reachable from another vertex i for all vertex pairs (i, j) in the given graph. Here reachable mean that there is a path from vertex i to j. The reachability matrix is called transitive closure of a graph.

Source: <https://practice.geeksforgeeks.org/problems/transitive-closure-of-a-graph0930/1/>

<https://www.techiedelight.com/transitive-closure-graph/>

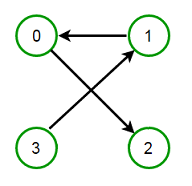
Format of the input:

***Input:*** *0 -> 1, 0 -> 2, 1 -> 2, 2 -> 0, 2 -> 3, 3 -> 3*

Your code should be able to traverse the above input format (e.g. *2 -> 0, 0 -> 2, 1 -> 2, 0 -> 1, 3 -> 3, 1 -> 3*) **from a given text file** and create edges in your data structure.

Example from: <https://www.techiedelight.com/transitive-closure-graph/>

For the following graph:



The Transitive closure/Reachability matrix is: a cell has value 1 if there is a path from the 2 nodes it represents.

1   0   1   0  
1   1   1   0  
0   0   1   0  
1   1   1   1

**Question 2 (30 points):**

Write the naïve version for solving the 0-1 Knapsack problem. Your goal is to enumerate all possible 2^n combinations given the list of items. From these combinations, choose and return the best combination.

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Source: <https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/>

Method 1 in the above source shows the recursive version to solve this naively. You should not use recursion for question 2. Question 3 asks you to use recursion.

Input and Output Example (from the above source):

Input will include Items as (value, weight) pairs and Knapsack capacity W *{{60, 10}, {100, 20}, {120, 30}}, W = 50*

*Output:*

*Maximum possible value = 240*

*Items to take :Item 1, 2*

Your code should be able to traverse the above input **from a given text file** and create edges in your data structure.

**Question 3 (30 points):**

Write the naïve version for solving the 0-1 Knapsack problem. Your goal is to enumerate all possible 2^n combinations given the list of items ***using recursion***. From these combinations, choose and return the best combination.

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

Source: <https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/>

Method 1 in the above source shows the recursive version to solve this naively.

Input and Output Example (from the above source):

Input will include Items as (value, weight) pairs and Knapsack capacity W *{{60, 10}, {100, 20}, {120, 30}}, W = 50*

*Output:*

*Maximum possible value = 240*

*Items to take :Item 1, 2*

Your code should be able to traverse the above input **from a given text file** and create edges in your data structure.

**Submission Instructions**: Put all your solutions in a properly commented file named *alternatelab5\_lastname\_firstname.EXTENSION*, where EXTENSION = the appropriate extension for the programming language that you chose.

Email your solutions to the TA ([lamia@nmsu.edu](mailto:lamia@nmsu.edu)) and cc me on it ([nagarkar@nmsu.edu](mailto:nagarkar@nmsu.edu))