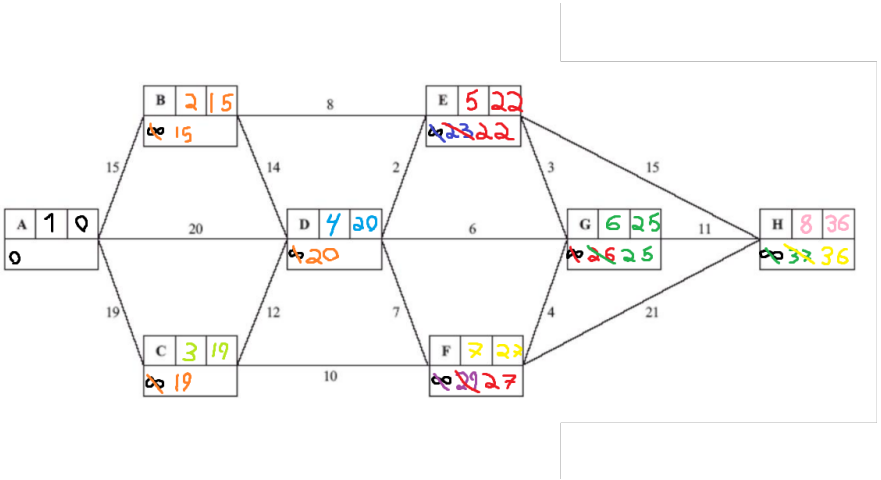
3500 hw 4

Casey Provitera

February 2024

1 (30 points) Question 1

Peter wants to minimize his driving time from his residence at point A to the university located at point H. The provided diagram displays the towns along with the respective driving times, measured in minutes, between them.



(a) Use Dijkstra's algorithm to determine the optimal routes from point A to point H that minimize the total travel time.
Dijkstra's algorithm is a greedy algorithm that uses a priority queue to chooses the edge with the lowest weights and find the lowest cost path from a starting node to an ending node
Dijkstra's algorithm would find the path $A- > D- > E- > G- > H$ since this path uses the lowest combined weighted edges.

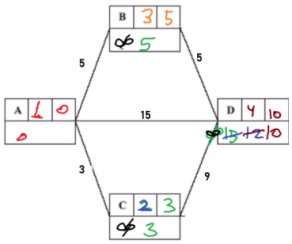
(b) What is the shortest path that should be taken from point A to point H?
The shortest path that would be taken from $A- > H$ is $A- > D- >$

$E->G->H$ since it uses the smallest weighted edge with a total of 36. (I know this is repetitive from part a but I think my answer to part a is necessary in how it is structured.)
Unless the shortest path is referring to the least amount of edges, in that case the shortest path could be any of the following 3 edge paths:
$A->D->G->H$ Weight = 37
$A->C->F->H$ Weight = 50
$A->B->E->H$ Weight = 38

Use the following key table to fill in the values in the graph.

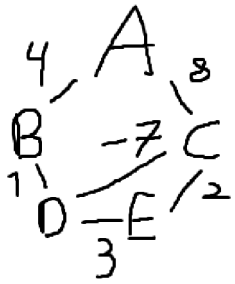
Vertex	Order of extraction from the queue	Final distance
Changing of the distance		

For example



2 (20 points) Question 2

During our discussions in class, we learned that Dijkstra's algorithm reliably computes the shortest paths in a graph where all edge weights are positive.
Why does Dijkstra's algorithm encounter issues or produce inaccurate results when negative edge weights are introduced into the graph? Discuss the reason behind this limitation and provide an example to illustrate the scenario.
Contrary to what would sound like the common sense of Dijkstra's algorithm it does not work with negative weights. This is due to the fact the is it a greedy algorithm and once a node is visited with its final distance calculated it can not be adjusted again.
Imagine we have the following undirected weighted graph:



In this graph the distances Dijkstra's algorithm would find are as follows:
d[A] = 0, d[B] = 4, d[C] = -2, *d[D] = 5*, d[E] = 2
I put stars around the distance to D since with the weight of -7 the real shortest distance to D is 1 by traveling the path $A- > C- > D$. The reason the correct distance to D is never found is because the path $A- > C- > B$ is never searched since $A- > C$ has a higher weight than the edge $A- > B$ which would make Dijkstra's algorithm search $A- > B$ first and not find the optimal path to D.

3 (50 points) Question 3

You've joined a game where you're given a box with a maximum weight limit, denoted as W Inside the room, there's an assortment of items, each identified by its weight W_i and value v_i where i = 1,2,3 ... n. The main objective is to carefully select items to place inside your box, aiming to maximize the total value of the chosen items while ensuring the total weight doesn't exceed the box's capacity. If the weight limit is surpassed, the box will break, leaving you with nothing.
An interesting feature of this game is the option to take fractional parts of items. For instance, if your box has a maximum weight capacity of 5 lbs and you've already loaded 3 lbs of items, you could opt to take half of an item weighing 4 lbs with a value of $14, resulting in 2 lbs and a corresponding value of $7.
You were considering three different greedy strategies to maximize the value:

1. The first approach involves selecting the item with the highest value.
2. The second strategy focuses on selecting the item with the lowest weight.
3. The third approach revolves around calculating the ratio of value to weight.
   (a) Do you believe the first strategy will yield an optimal solution? Justify your answer and illustrate with an example.
   I do not believe the first approach will yield an optimal solution for all inputs. For example imagine we start the game with two objects,

one with a weight equal to 5 and a value equal to 7, and another with a weight equal to 1 and a value equal to 2. And we can only hold a total weight of 5. If we choose the object with the highest points we will immediately fill our basket and have a total value of 7, but if we had chosen to hold 5 of the other object with the value of 2 we will be able to hold 5 of those objects for a total value of 10. Showing the approach of choosing the object with the most points is not always optimal.

(b) Do you believe the second approach will yield an optimal solution? Justify your answer and illustrate with an example.
I do not believe this approach will always yield an optimal solution either. Let us take a different example where we have 2 objects, one with a weight of 2 and a value of 3 and one with a weight of 6 and a value of 10, and where our basket can hold a total weight of 6. If we use the approach where we choose the item with the lowest weight we will end up holding 3 of the items with the weight of 2 for a total value of 9. But this is not the optimal solution for this problem since if we had chosen to hold one object with the weight of 6 we would have ended with a total value of 10. Showing this second approach of choosing the object with the lowest weight is also not always optimal.

(c) Do you believe the third strategy will yield an optimal solution? Justify your answer and illustrate with an example.
Unlike the other two approaches I do believe this one will always results in an optimal solution. To show this let us use the two prior examples that were used to disprove approaches 1 and 2.
Example 1:
2 objects - one with the weight of 1 and value of 2 and one with the weight of 5 and the value of 7.
If we calculate the ratio of value to weight for both objects we get the following results, for the lower weight object we get a 2:1 ratio which reduces to 2, and for the higher weight object we get a 7:5 ratio which reduces to 1.4. Clearly the 2:1 ration is much better so we will pick this object and end with the optimal solution with a value of 10.
Example 2:
2 objects - one with the weight of 2 and a value of 3 and one with the weight of 6 and a value of 10.
If we calculate the ration of value to weight for both objects we get the following results, for the lower weight object we get a 3:2 ratio which reduces to 1.5, and for the higher weight object we get a ration of 10:6 which reduces to 1.67. Here we can again see that the 10:6 ratio is better than the 3:2 ratio and if we pick this object we end up with the optimal solution with a value of 10.

(d) Write an algorithm based on the chosen greedy approach.

i. findMaxValue(array of items each having a weight and a value)

ii. ratio = {}
iii. max = 0
iv. for item in items
v. ratio[item] = item[value]/item[weight]
vi. if ratio[item] > max
vii. max = ratio[item]
viii. exclusively use the item whose ratio = max to fill basket

(e) Analyze the time complexity of your algorithm.
The time complexity of my algorithm is n since we need to calculate the ratio for each item that is in the current game.
Every step in the algorithm is O(1) but wwe loop through the list which has a length of n, making the final time complexity O(n).

(f) Prove why the proposed algorithm consistently provides the optimal solution.
Since we are able to take fractions of items and get the corresponding fractions of the value there will only ever be 1 correct option to pick out of the items and that would be the one that gives you the most value from the ratio of value to weight.
Let's try to prove the above using contradiction, Lets say there is one item with a value to weight ratio of 2 but there is also a value to weight ratio of 1.5. If we take the item with the 1.5 ratio we would be missing out on the .5 additional value we could get from the 2. But wait maybe the 2 ratio item weighs a lot more so it is better to take the 1.5? NO that would not matter, since we converted all the weights and values of each item into a ratio it is now like every item weighs a unit but has different values. Even if the item with the 2 ratio weighs triple the item with the 1.5 ratio and we can only hold one of the 2 ratio item it will be more beneficial to choose the heavier item also known as the one with the higher ratio.