Machine Learning

Exact Solution

Dias, Schiewe, Pattanaik *Instructor: Pattanaik*

§ Week VIII §

Problem 1: Whatcha Packin'

Solve the following binary knapsack problem using enumeration.

item	weight	value
1	1	2
2	2	4
3	3	5
4	4	5
5	2	3
6	1	3

with knapsack capacity of 8.

Solution:

At first glance, the goal would be to try all possible subsets: $2^6 = 64$ subsets. However, based on the weight limit, a few of those subsets can be skipped:

Using sub-problems and recursion, provide a possible algorithm to solve this problem. From the limit of 8, we can figure out the combinations of weights that are equal to 8 are:

- Items 1, 6, 2, 4;
- Items 1, 6, 5, 4;
- Items 1, 3, 4;
- Items 6, 4, 6;
- Items 2, 3, 5, 6;
- Items 1, 2, 3, 5;
- Items 2, 4, 5.

With a total of 7 subsets, the total amount of sets to be tested is only 11% of the total. Based on their value, the best one is:

- Items 1, 6, 2, 4: value 14;
- Items 1, 6, 5, 4: value 13;
- Items 1, 3, 4: value 12;
- Items 6, 4, 6: value 13;
- Items 2, 3, 5, 6: value 14;
- Items 1, 2, 3, 5: value 15;
- Items 2, 4, 5: value 12.

The best set to pack consists of items 1, 2, 3, and 5 with a total value of 15 and weight equal to 8.

Problem 2: Tracking Order

Considering the following graph, provide a solution for the TSP using enumeration:

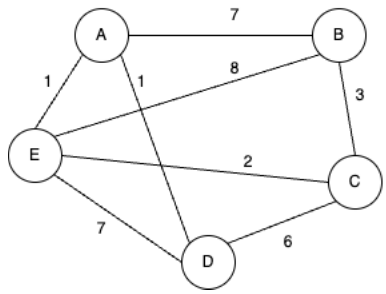

Figure 1: Undirected weighted graph

Is it necessary to check all subsets?

Solution:

First, let us pick a starting node. For example, node A. With that, we create the following tree of options:

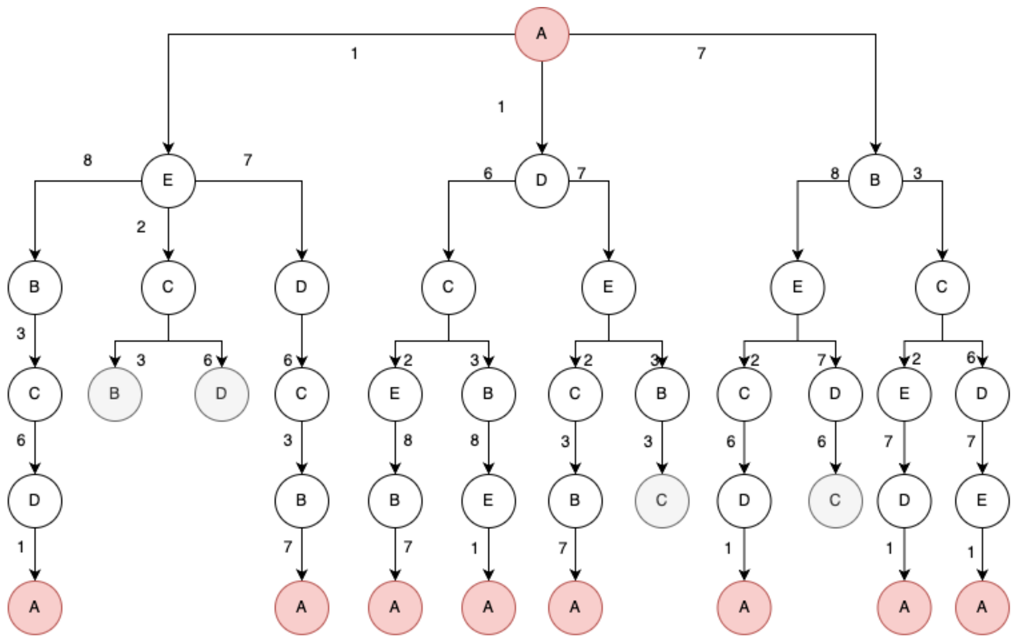

Figure 2: All possible routes in this graph

Note that this number is drastically smaller than all subsets available in the graph (maximum 12, with actual 9 applicable out of 2^5).

Now, we calculate the shortest route:

The shortest route is, at the end: $A - E - B - C - D - A$.

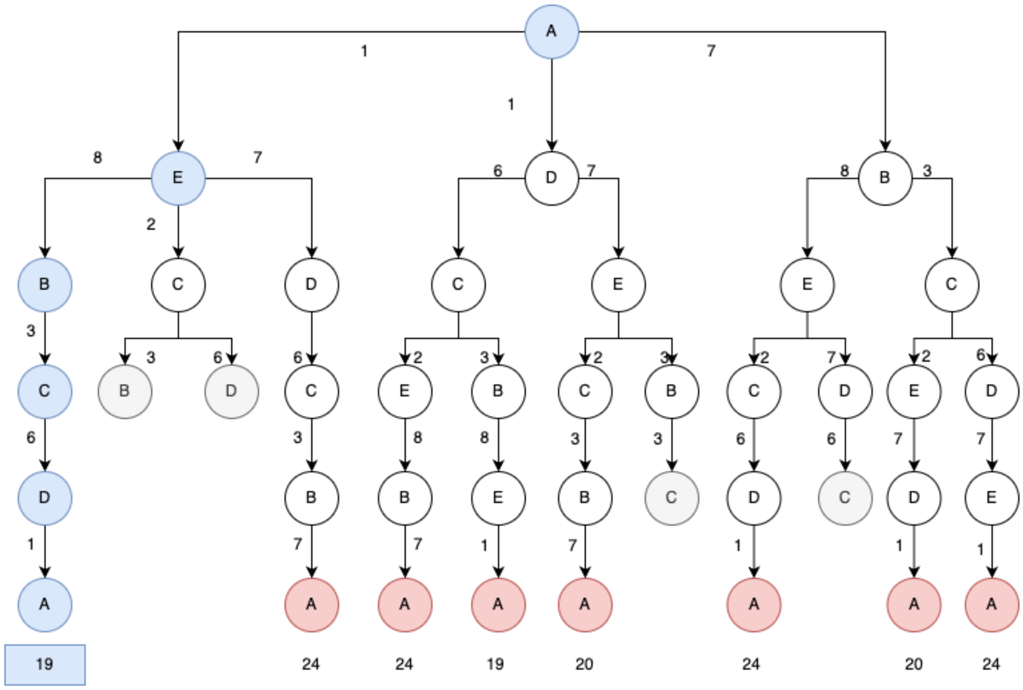

Figure 3: All possible routes in this graph, with cost calculated

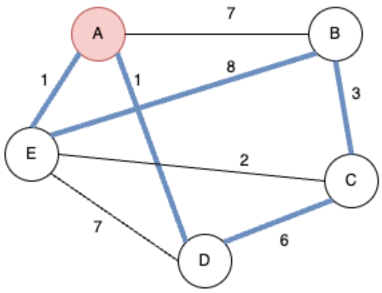

Figure 4: All possible routes in this graph

Problem 3: One by One

Solve the shortest path between node s and node 4 using the dynamic programming principle.

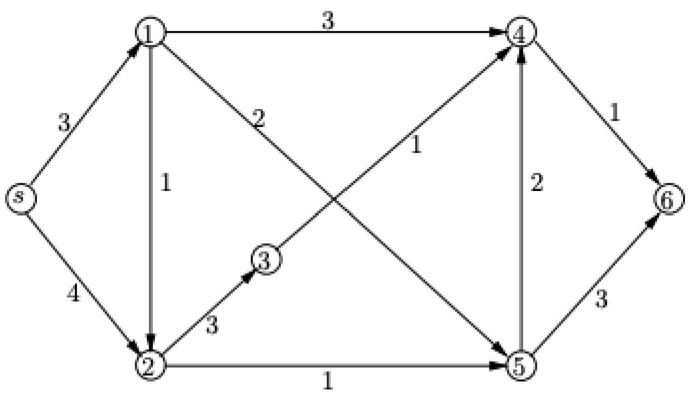

Figure 5: Example of a flow network

Repeat the process between node s and node 6.

Solution:

We first calculate the path through recursion and sub-problems to solve and apply Dijkstra's algorithm (or any algorithm for the shortest path).

In this case, the path between node s and node 4 is calculated by finding the optimal path between node s and the predecessor of node 4. The shortest path for the node predecessor of node 4 is done by the shortest path between node s and the predecessor of the predecessor of node 4. Repeat this process until it reaches a trivial/base case and rebuild the solution from that.

For this graph, the shortest path from node s to node 4 has to be either through node 1 or node 3 or node 5, so we must solve all those subproblems. To solve the shortest path to node 1, we can solve directly from node s. Therefore, we reach a base case. From node 5, we can go through node 2 or node 1. From node 2, there are two options: either through node 1 or node s, which is a trivial case and for node 2, the only option is also from node s, which is the same trivial case. From note 3, the option goes through node 2, which returns to the case described beforehand. This can be expressed in a simple tree:

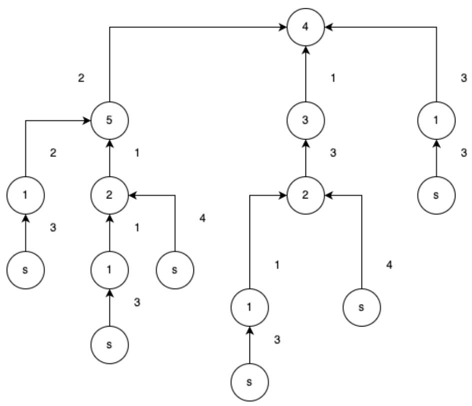

Figure 6: All routes towards node 4.

With those branches in mind, we calculate the cost of each path:

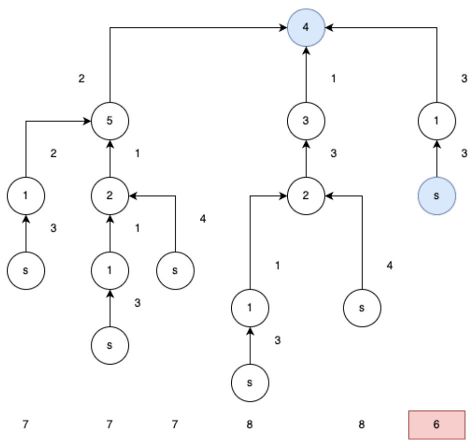

Figure 7: All routes towards node 4 with cost calculation.

The best path is $s - 1 - 4$ with cost of 6.

Now, for node 6, the same procedure is done, resulting in:

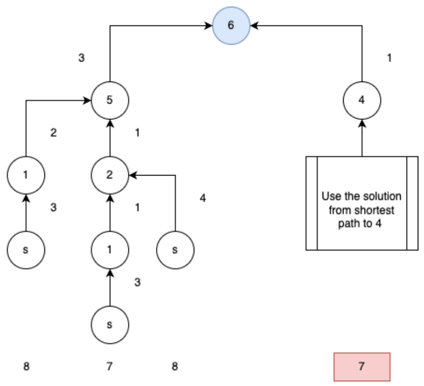

Figure 8: All routes towards node 6 with cost calculation.

with path $s - 1 - 4 - 6$ with cost equal to 7.