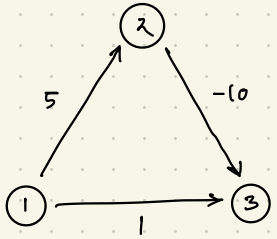
Andrés Alam Sánchez Torres 492253

1. Dijkstra's algorithm is known for solving the shortest path problem, but it requires that the edge weights be non-negative. Give an explanation or an example to show that the Dijkstra's algorithm does not always find the shortest path in case of negative edge weights. List at least one algorithm to find shortest path in case of negative weights.



Consider the example of finding the shortest path from 1 to 3. The first edge that will be searched is (1,3), and will mark that as the shortest path, when the right answer is the path $1 \to 2 \to 3$ with cost -5, i.e. edges with higher costs will be deprioritized, even if they are part of the shortest path.

- The greedy approach of Dijkstra's assumes the first path found from expanding the cheapest path will yield the optimal solution, but negative weights remove this guarantee.

2. Let G be a graph and let l be a non-negative length function on the edges. Define a function f on the vertex pairs by letting $f(u,v) =$ the length of a shortest $u - v$ path in G with respect to l.
Show that f (which is the "distance function") satisfies the triangle inequality, that is, for all triples x, y, z of vertices it holds that $f(x,y) \leq f(x,z) + f(z,y)$.

The function f can be defined as follows

$$f(x,y) = \min_{\substack{w_i \in V(G) \\ (w_i, w_{i+1}) \in E(G)}} \sum l(w_i, w_{i+1}) \qquad i = 1, \dots, n$$

Where w_1 and w_n correspond to x and y, respectively. So, f is the minimum lengths sum of every sequence of adjacent edges that connect x and y. Then,

$$\begin{aligned} f(x,z) + f(z,y) &= \min \sum l(u_i, u_{i+1}) + \min \sum l(v_j, v_{j+1}) \\ &= \min \sum l(u_i, u_{i+1}) + \sum l(v_j, v_{j+1}) \end{aligned}$$

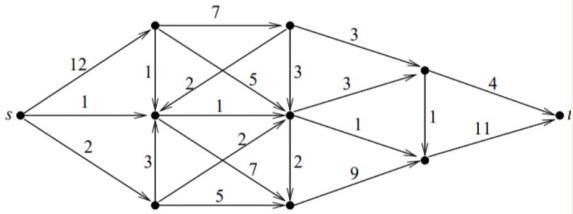
$i = 1, \dots, n \quad j = 1, \dots, m$

where u_1 and v_m correspond to x and y, respectively, and $u_n = v_1 = z$. So the sum $\sum l(u_i, u_{i+1}) + \sum l(v_j, v_{j+1})$ is among the sums that $f(x,y)$ minimizes. Therefore, $f(x,y) \leq f(x,z) + f(z,y)$.

3. Give a simple algorithm to decide whether it is possible to construct a minimum spanning tree of a connected graph containing an arbitrarily chosen edge.

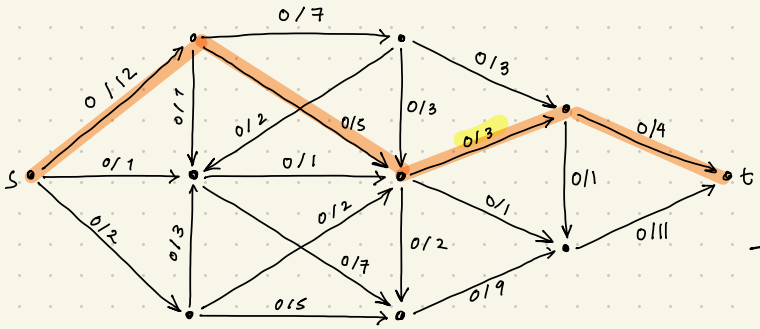
Modifying Kruskal algorithm. Before starting adding edges to the tree, we add the chosen edge. If the algorithm doesn't add all vertices to the tree, then it's not possible.

4. Kruskal's algorithm and Prim's algorithm are known to find a minimum spanning tree of a connected weighted graph. By modifying both algorithms, is it possible to find a spanning forest of minimum weight in a disconnected weighted graph?
5. Find an $s - t$ flow of maximum value and an $s - t$ cut of minimum capacity in the following graph:



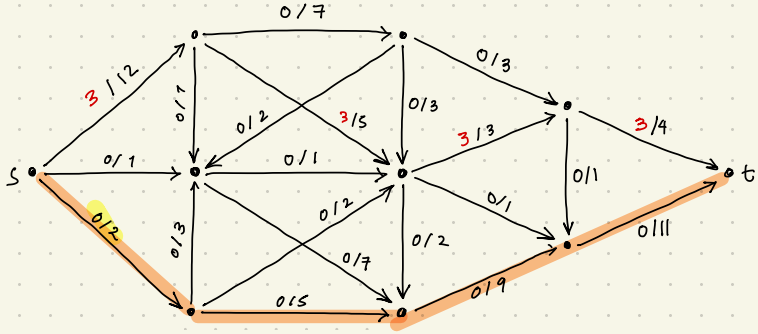
4. • Kruskal's will produce a spanning forest without modification. Although an implementation will require to get the number of connected components (looking at the produced disjoint sets) and split the selected edges accordingly.
• Prim's can be applied multiple times starting from every vertex that has not been picked for an MST yet. Similar to the algorithm to find connected components.

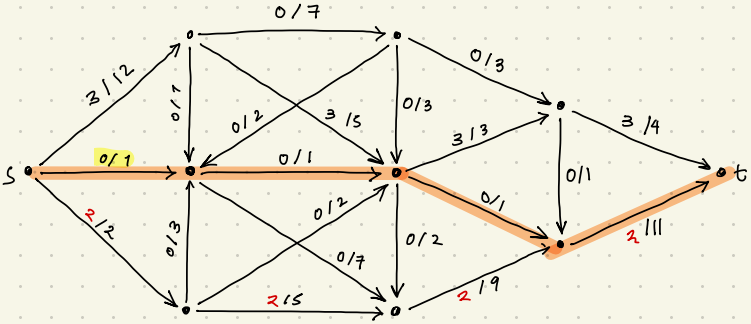
5. We apply the Ford-Fulkerson algorithm, pushing flows through s-t in the residual graph until there are no more paths. Note for simplicity the backward edges are implicit.

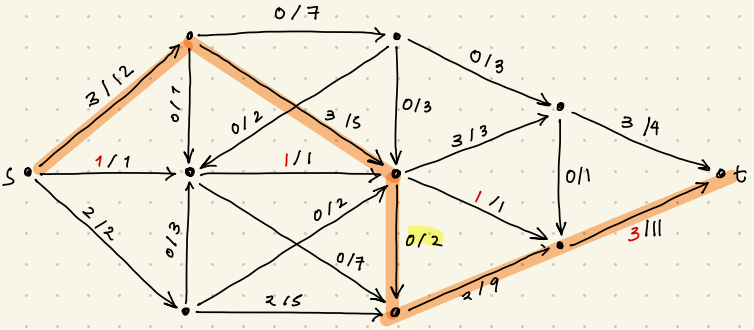


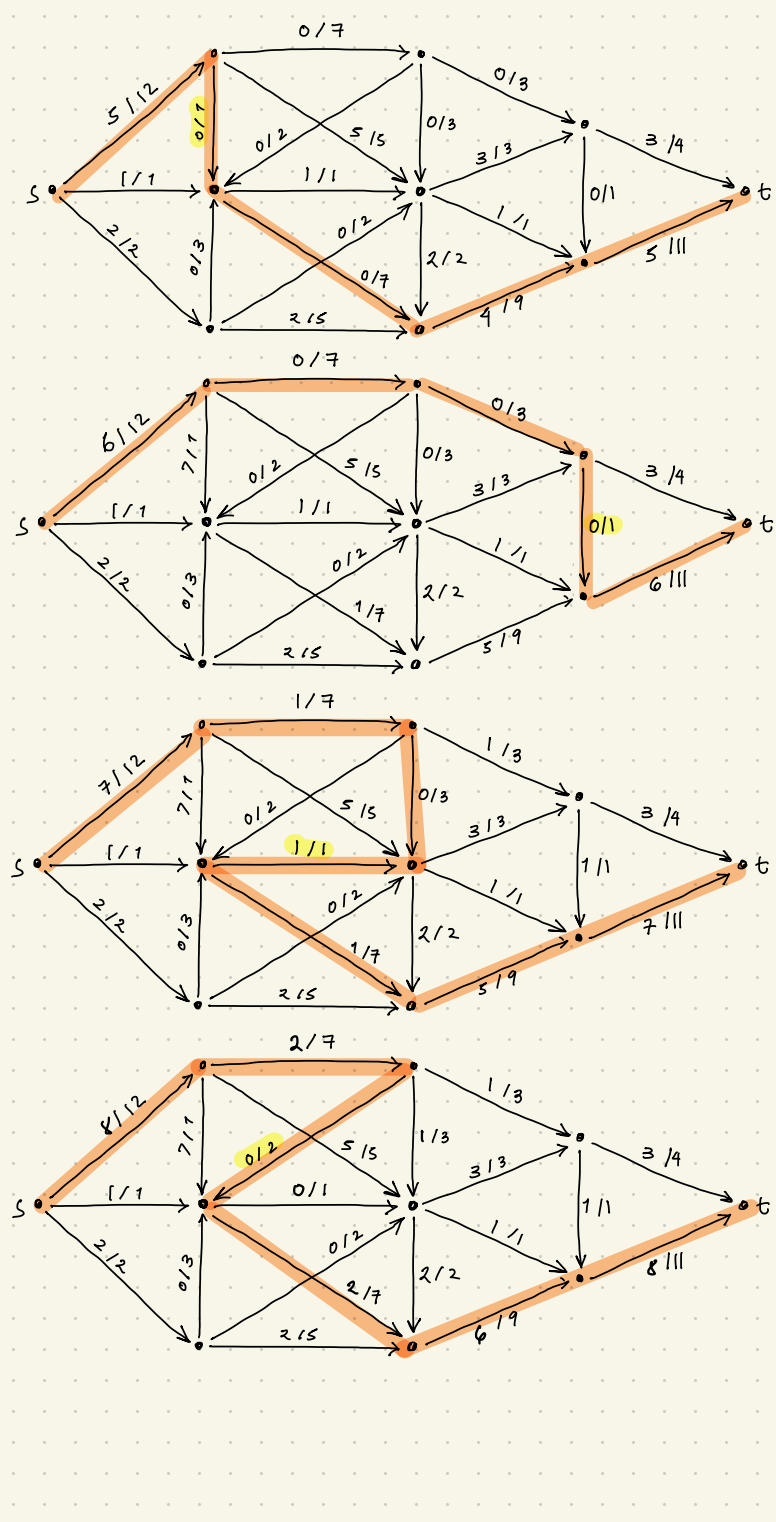
On each path is highlighted the edge w/ minimum capacity

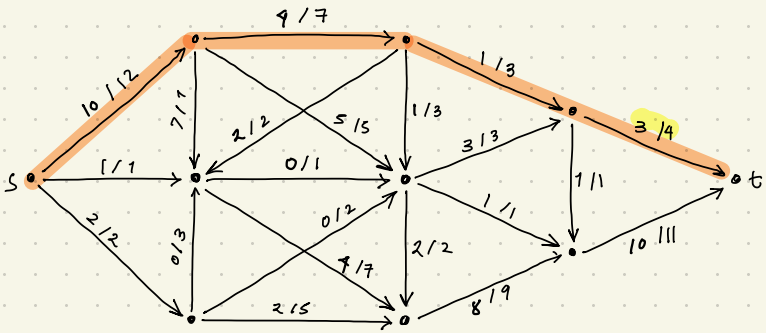
→ flow pushed to the path.

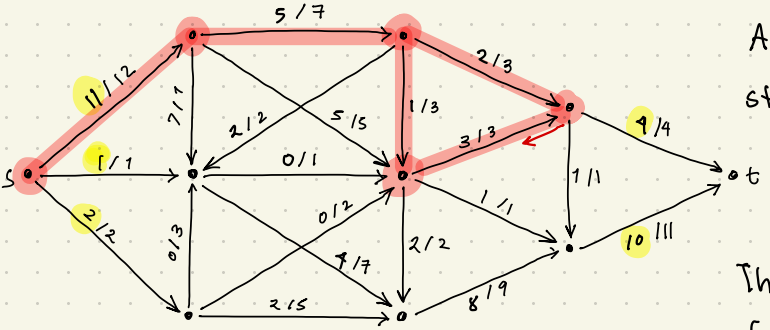






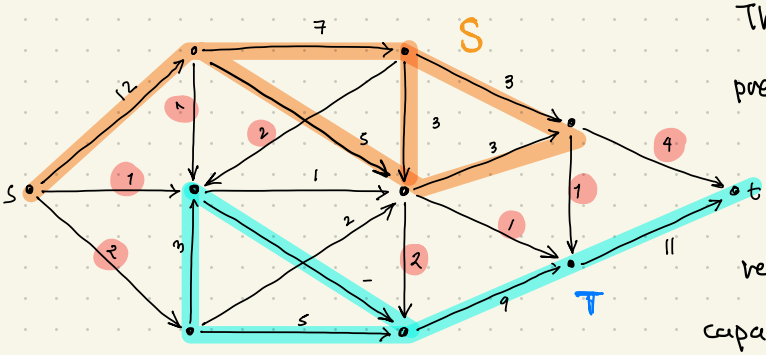

0 / 7
5 / 12
0 / 7
0 / 2
5 / 5
0 / 3
0 / 3
3 / 3
3 / 4
s
1 / 1
t
0 / 1
2 / 2
0 / 3
0 / 2
1 / 1
0 / 7
2 / 2
5 / 11
2 / 5
4 / 9
0 / 7
6 / 12
7 / 1
0 / 3
0 / 2
5 / 5
0 / 3
3 / 3
3 / 4
s
1 / 1
t
0 / 1
2 / 2
0 / 3
0 / 2
1 / 1
1 / 7
2 / 2
6 / 11
2 / 5
5 / 9
1 / 7
7 / 12
7 / 1
1 / 3
0 / 2
5 / 5
0 / 3
3 / 3
3 / 4
s
1 / 1
1 / 1
t
2 / 2
0 / 3
0 / 2
1 / 1
1 / 7
2 / 2
7 / 11
2 / 5
5 / 9
2 / 7
8 / 12
7 / 1
1 / 3
0 / 2
5 / 5
1 / 3
3 / 3
3 / 4
s
0 / 1
t
1 / 1
2 / 2
0 / 3
0 / 2
2 / 7
2 / 2
8 / 11
2 / 5
6 / 9





After all the previous steps, we notice we can't push any more flow past the marked nodes. Thus we found the max flow.

max flow = 14



The nodes marked on the previous step also make the set S of the minimum cut, as the flow is restricted by the outgoing capacity of S.

This also allow us to confirm 14 is the max flow, since the sum of the edges from S to T is also 14.