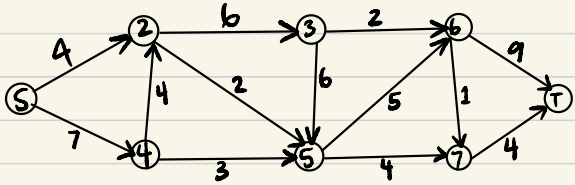
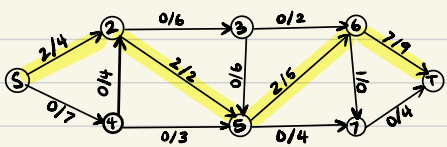
1)

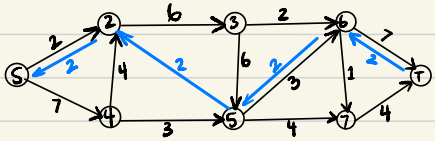


i) shortest augmenting path

↳ this is the path that takes the least amount of edges from S to T



Flow Network



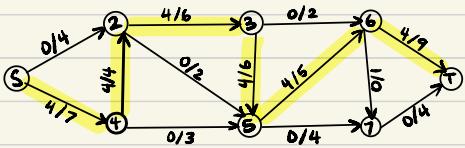
Residual Network

Path: S → 2 → 5 → 6 → T

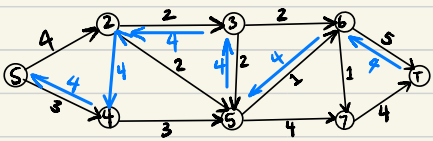
the bottleneck is 2 because the path from 2 → 5 is the lowest capacity value throughout the path. Thus anything > 2 will surpass capacity.

ii) highest capacity path

↳ this is the path that has the highest bottleneck



Flow Network



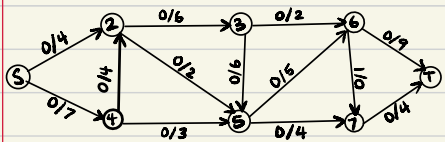
Residual Network

Path: S → 4 → 2 → 3 → 5 → 6 → T

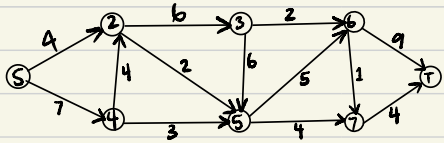
the bottleneck is 4 because the path from 4 → 2 is the lowest capacity value throughout the path. Thus anything > 4 will surpass capacity.

(iii) use ford fulkerson algorithm to determine max flow

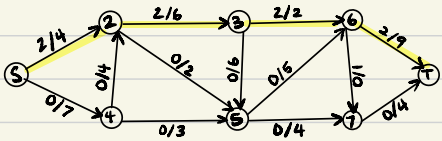
Flow Network

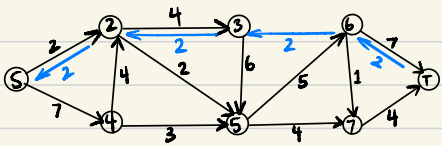


Residual Network

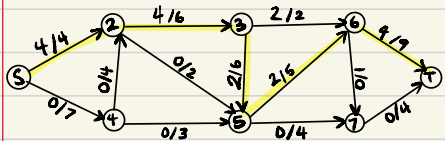


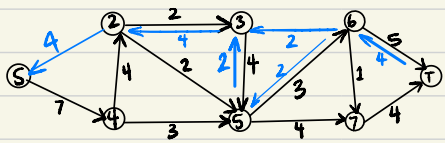
1) Selecting path S→2→3→6→T. Bottleneck capacity = 2



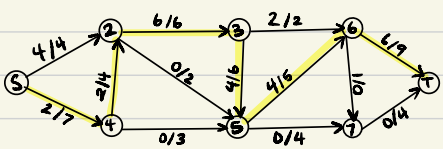


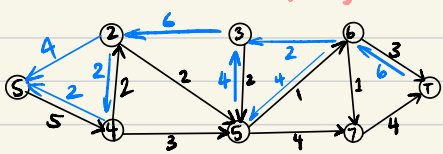
2) Selecting path S→2→3→5→6→T. Bottleneck capacity = 2



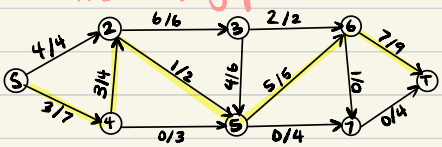


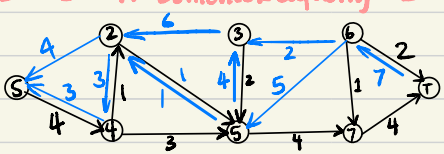
3) Selecting path S→4→2→3→5→6→T. Bottleneck capacity = 2



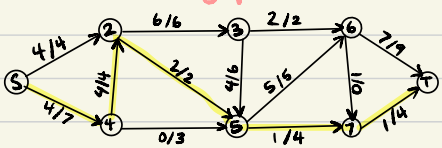


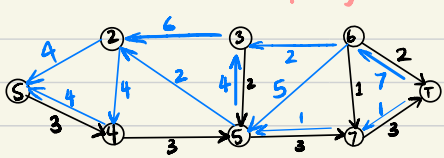
4) Selecting path S→4→2→5→6→T. Bottleneck capacity = 1



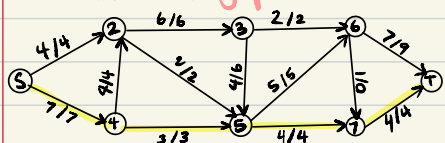


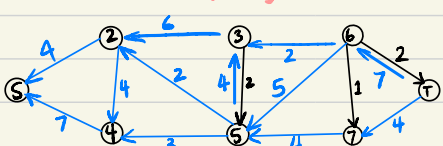
5) Selecting path S→4→2→5→7→T. Bottleneck capacity = 1





6) Selecting path S→4→5→7→T. Bottleneck capacity = 3





max flow = 4+7 = 11

3) The process of selecting the augmenting path is by selecting the shortest available path using breadth-first search (BFS). This means that that path that takes the shortest number of edges, regardless of what the capacities (or weights) on each edge are. Finding the augmenting path through this method will take O(V+E) time. As the algorithm goes through multiple iterations, the length of the shortest path will never decrease.

With every augmenting path or iteration, at least 1 edge hits it capacity because the lowest value in the path will be the bottleneck (aka 1 less path is available in the residual graph). This leads to other edges being needed for the next augmenting path. The chosen shortest path gets longer and longer. Since the greedy criteria for the algorithm would be to take the shortest path, the algorithm will definitely terminate as edges hit their capacities and s and t eventually have no available path that connects them.

By choosing the path with the least edges, this will often allow for having less restricted bottlenecks (as shown in example below where DFS and BFS are compared). DFS would take a longer time than BFS in the example. With taking the path with the shortest number of edges in selecting the augmenting path, the ford-fulkerson algorithm will end up taking a time complexity of O(V*E^2).