Saleh Alkhalifa
11/28/2022

Homework #10

1) Pipe network capacities in Genovia (20 pts total)

After finding out you are secretly part of the royal family of Genovia, you inherit a 16th-century Genovian castle with an elaborate plumbing system that has accumulated pipes, junctions, and clogs over four centuries. Instead of a diagram, you are given a list of pipes and their capacities leading from the water source to your bathroom

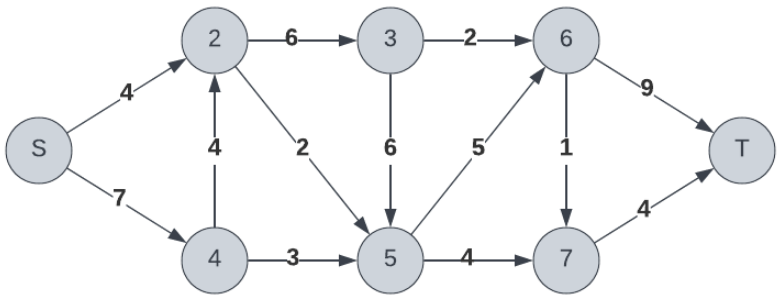
Pipes	S,2	S,4	2,3	2,5	4,2	4,5	3,5	3,6	5,6	5,7	6,7	6,T	7, T
Capacity	4	7	6	2	4	3	6	2	5	4	1	9	4

Draw the flow graph of your new castle and list:

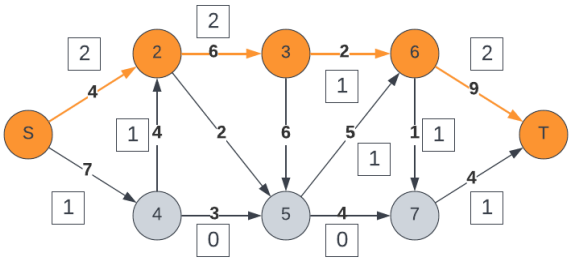
a. The shortest augmenting path (and it's bottleneck)
b. The highest capacity augmenting path (and it's bottleneck)

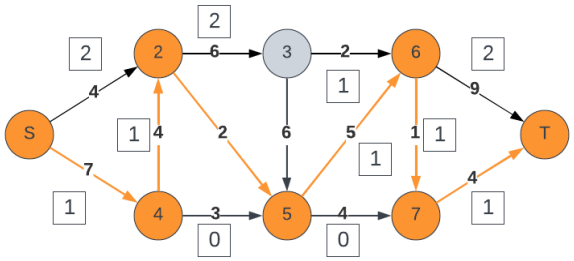
- We can define an augmenting path as a path that is created by repeatedly finding and forming a path of positive capacity from a given source to a sink, and repeatedly adding to the flow.
- A shortest augmenting path algorithm generally performs at most O(mn) augmentations.
- Shortest augmenting paths ensures that the number of augmenting paths is minimized to reduce overall running time.
- We can use the Ford-Fulkerson algorithm to determine the maximum flow for a given system, and we can do this by augmenting the flow repeatedly using residual graphs.
- Using the data provided we can create the following representation.

Original Path



- We can then identify the two augmenting paths represented below. We start off with the flows at 0, and update with each step. We can see the two paths below where the first has a **bottleneck of 2**, and the second with **bottleneck of 1**.

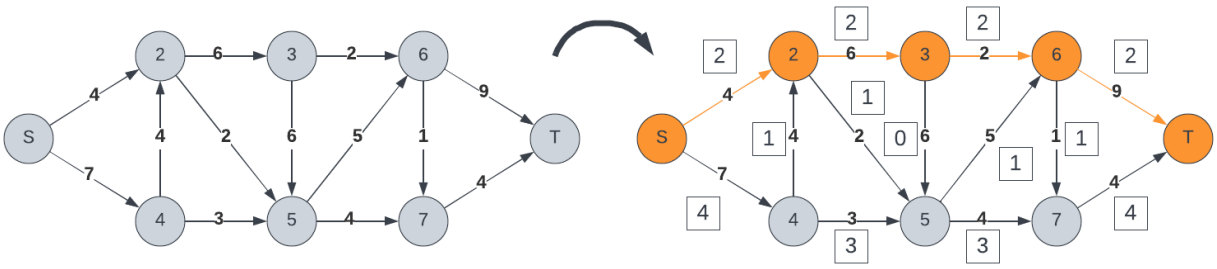




Homework #10

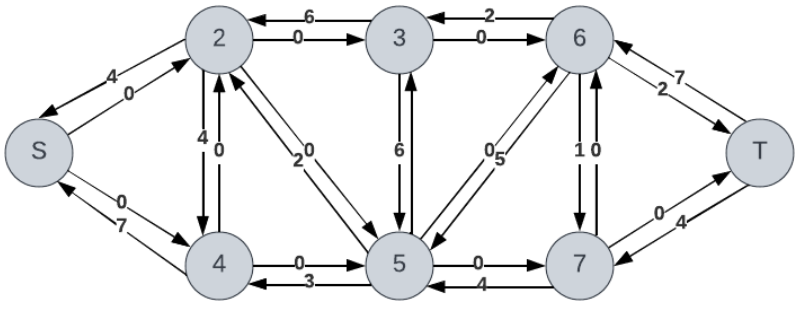
#	Path	Bottleneck
S	S -> 2 -> 3 -> 6 -> T	2
H	S -> 4 -> 2 -> 5 -> 6 -> 7 -> T	1

- As we update the graph to account for the paths, we arrive at the augmenting path:
  - **Path: S -> 4 -> 5 -> 7 -> t**
  - **Bottleneck Capacity = 3**



You'd like to know if it is safe to install a modern shower, or if this will eventually overflow the historic bathtub. Use the Ford Fulkerson algorithm to determine the max flow of this flow network/graph. Please draw your final residual graph and write the calculated max flow.

- Using this methodology (**Ford-Fulkerson**), we are able to see the maximum flow determined.
- Algorithm:
  - Start a while loop while an augmenting path exists
  - We first find the augmenting path via the depth-first searching (DFS) algorithm
  - Start a for loop for each of the edges u -> v in the given path
    - We first decrease capacity of u->v
    - Increase capacity of v -> u
  - Break out of the loop and update the maximum flow
- Following that, we arrive at the following diagram showing the results:



Maximum flow = minimum cut from s to t = 11

Homework #10

2) Augmenting Paths (10 points)

Not all augmenting paths are equal, and starting with different paths leads to different residual graphs, although all selections produce the same max-flow result. Determine a process for selecting your augmenting paths. Justify your answer. **Hint:** most implementations of Ford-Fulkerson take a greedy approach.

- When it comes to determining the augmenting paths in a given network, there are many different algorithms and methods one can use.
- Some of these methods include: (1) Shortest-Path and (2) Highest Capacity
- These methods are seen in algorithms such as Ford-Fulkerson, Edmonds-Karp, or Dinic.
- For the purposes of this question, I will focus on Edmonds-Karp.
- Edmonds-Karp is an implementation of Ford-Fulkerson, with the purpose or objective of computing the maximum flow in a given network.
- The **Edmonds-Karp** algorithm can determine maximum flow in $O(|V| * |E|^2)$ time

- The Ford-Fulkerson Algorithm comprises the following steps to determine maximum flow
    - Start a while loop while an augmenting path exists
        - We first find the augmenting path via the depth-first searching (DFS) algorithm
        - Start a for loop for each of the edges u -> v in the given path
            - We first decrease capacity of u->v
            - Increase capacity of v -> u
        - Break out of the loop and update the maximum flow

- The **Edmonds-Karp Algorithm** is almost identical; however, it refines the algorithm by always selecting the augmenting path with the **smallest number of edges**. We can see that in the following pseudocode:
    - Start a while loop while an augmenting path from S -> T exists:
        - Set variable P as the path in the graph with the minimum number of edges using BFS (Breadth First Search)
        - Augment f using the path P
        - Upgrade the graph G

- This algorithm allows us to reach a polynomial running time based on m edges and n nodes.

- It's important to note the two lemmas:
    - Throughout the algorithm, the length of the shortest path determined will never decrease
    - After m shortest path augmentations, the length of the shortest path strictly increases

- Theorem. The runtime for the shortest augmenting path here runs in $O(m^2n)$
    - We can say that O(m+n) is the time to find the shortest augmenting path via Breadth-first search (BFS)
    - Way can say that there is O(m) augmentations for the paths comprising k arcs
    - If an augmenting path exists:
        - 1 =< k < n
        - O(mn) augmentations

Resources:

[1] https://northeastern.instructure.com/courses/117409/pages/module-10

[2] https://en.wikipedia.org/wiki/Ford%E2%80%93Fulkerson_algorithm

[3] Introduction to Algorithms, Cormen, Third Edition. (CLRS)

[4] https://visualgo.net/en/maxflow