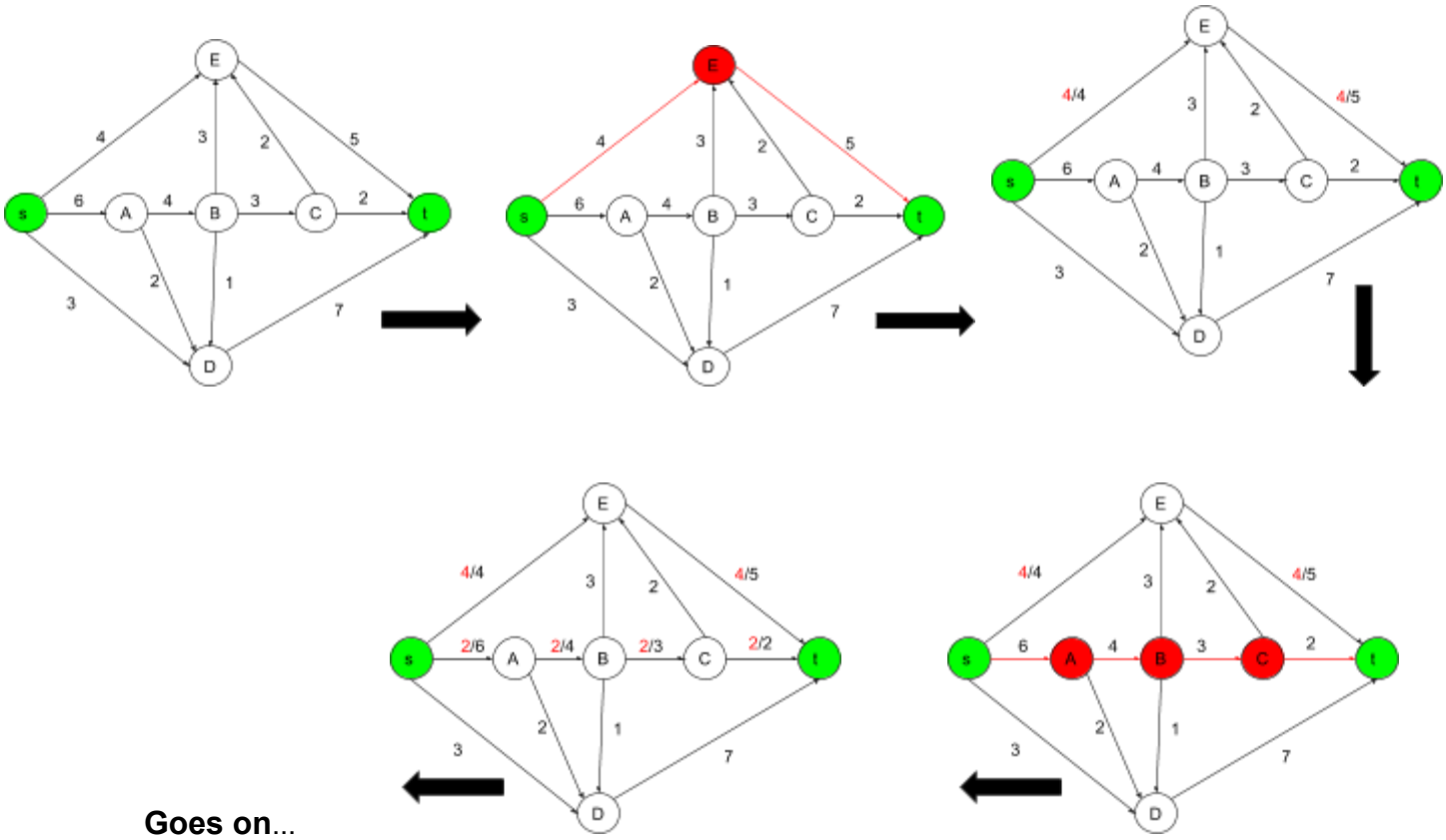
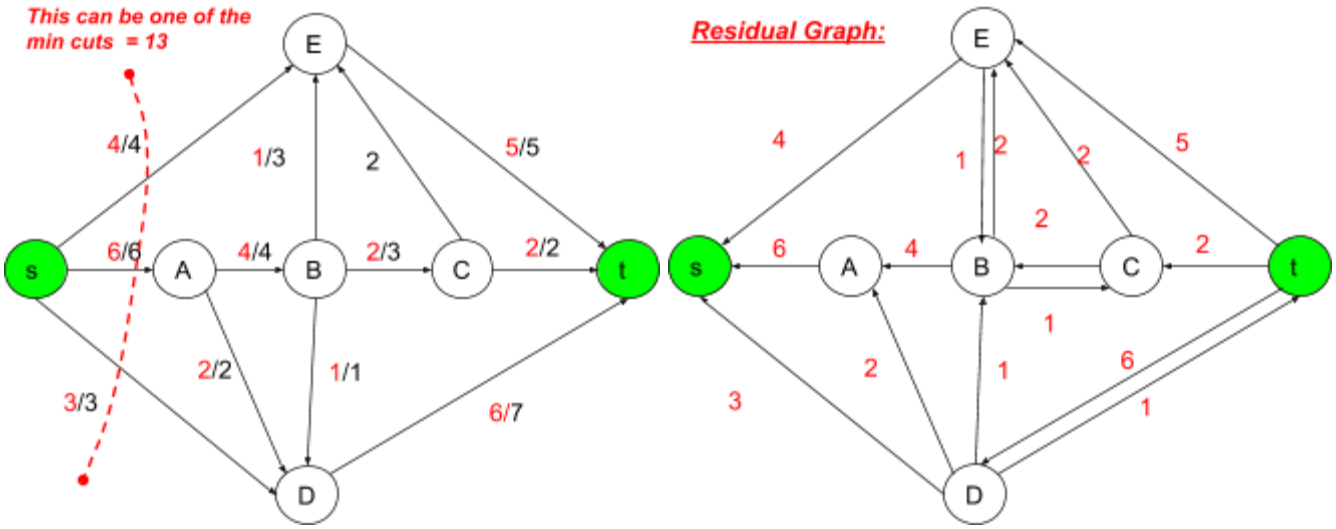
Question1:



Goes on...

If we keep doing the above tracing and updating, we would eventually get a resultant network graph as below:



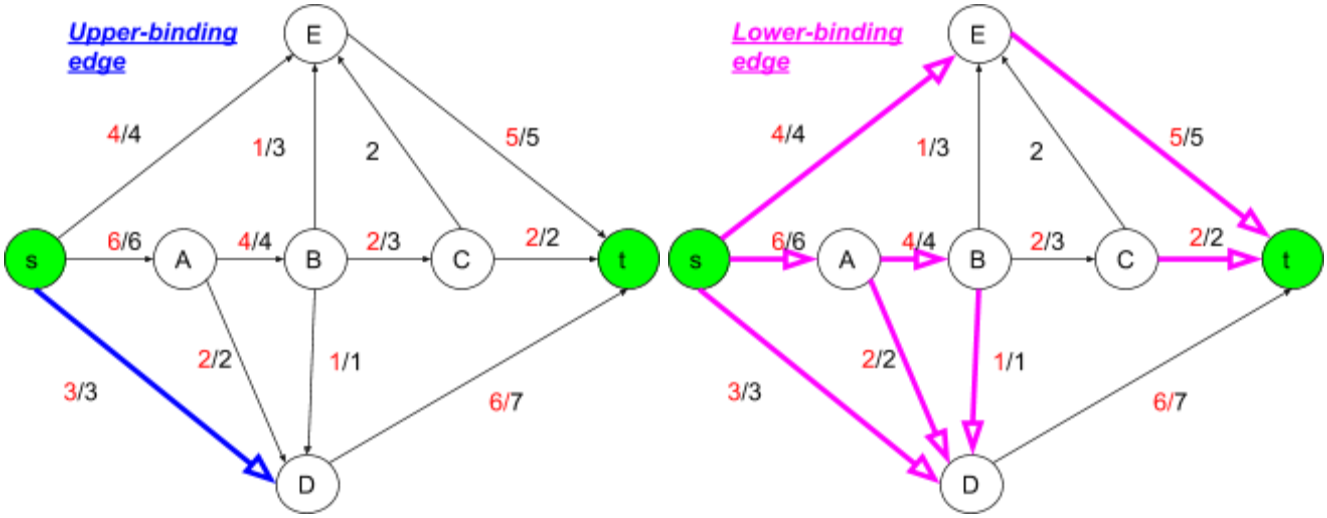
According to the graph:

Max s-t flow $f*$: 4+6+3 = **13**

Minimum s-t cut = **13**

The dotted line in the above graph can be one of the minimum cuts.

Question2:



Question3: findUpper():

Let capacity(e) represents the capacity of a particular edge in the graph

Algo analysis and explanation:

1. Assume we have an edge e with $f^*(e)$ = $Capacity(e)$, for example, the edge $s \rightarrow D$. If we increase the capacity(e) for that particular edge, what happens is that there will be am edge going from edge D to edge s with a capacity of 1.
2. According to Ford-Fulkerson Algorithm, when we get the max s-t flow f^*, there is no path from s to t in the graph G_{f*}. Say the edge $s \rightarrow D$ has a capacity of 3, and originally it is the max flow. When we increase the capacity for that edge by 1, the new edge generated $D \rightarrow s$ has a capacity of 1. This new edge could very well be a new part of the path $s \rightarrow t$. If there exists a path through $s \rightarrow t$, we can imply that the flow from $s \rightarrow t$ increases by 1 and $f^*(e)$ is no more max, in another word, e is the upper-binding edge. If after we increased the $Capacity(e)$ by 1, there is still no path from $s \rightarrow t$, then we can say that f^* remains the max.

Algorithm for find the upper-binding edge

---

1. Make a max flow graph G_{f*}
2. Use BFS to find all vertices adjacent to s (starting vertex), store them in a set A
3. Reverse the direction of edges in G_{f*} and store them in $G_{reverse}$
4. Use BFS again to find all edges in $G_{reverse}$ adjacent to t (destination vertex), store them in a set B
5. Find any edge $a \rightarrow b$ in the graph G such that $a \in A$ and $b \in B$, such an edge is the upper-binding edge.
6. COMPLETE

Running Time analysis

If we choose to implement an adjacency list data structure, for each vertex v, we traverse to the nodes that are adjacent to it. The total time spent scanning the list is $O(E)$, and enqueuing/dequeuing each takes $O(1)$, and the total time for queue operation is $O(V)$. In total, the total time for BFS is $O(V+E)$ or in the notation given, $O(m+n)$. The two BFS operation each takes $O(m+n)$, and the step5 takes $O(m),$ therefore, the total running time for my algorithm is $2\,O(m+n)+O(m) \rightarrow O(m+n) \rightarrow$ linear!

Proof of correctness

1) I have shown in the explanation of the algorithm that originally we get the max s-t flow f^*, there is no path from s to t in the graph G_{f*}. An edge e is an upper-binding edge if and only if after increasing its capacity there exists a path from s to t. In my algorithm, BFS algo find set A and B which contains vertices reachable from s and all vertices reachable from t in the reversed graph. And then it finds all the edges that connect these two vertex sets. This is exactly the definition of upper-binding edge.
2) Furthermore, if choose the upper-binding edge, assisted by the Ford-Fulkerson, the edges will be added to residual graph and the max flow will be changed, increasing the capacity of other not chosen ones will not affect the max flow since they don't satisfy the requirement/definition imposed by the algorithm.
3) COMPLETE.