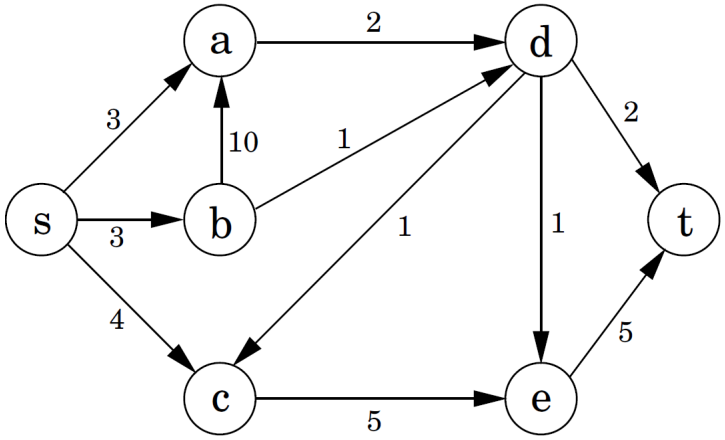
CMPSC 465 Spring 2024 | Data Structures & Algorithms | Mehrdad Mahdavi and David Koslicki

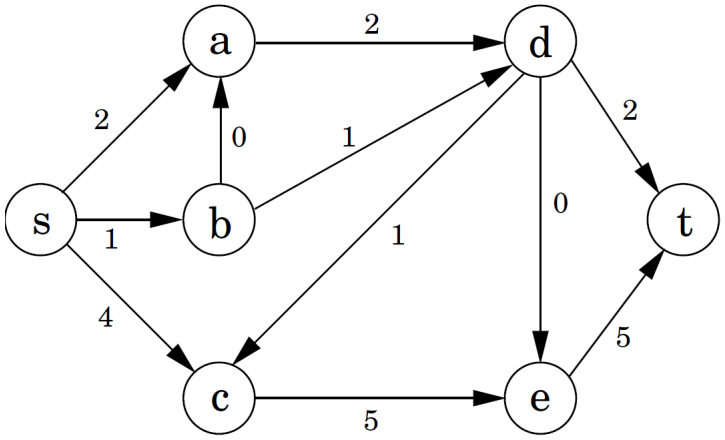
Worksheet 8

Wednesday, March 13, 2024

1. **Max-Flow.** Find the maximum flow from vertex s to vertex t in the following graph using the Ford-Fulkerson Algorithm. Show the residual graph at each step of the process.



Solution: The numbers below indicate the value of the flows on the edges of the graph. The flow itself is not necessarily unique, but the max flow value of 7 is.

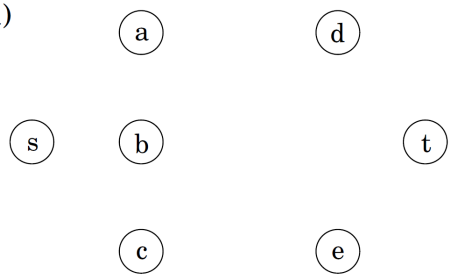


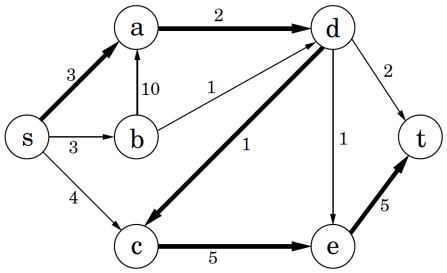
Below is one approach to get to the maximum flow. The bold edges in the residual graph represent augmenting paths from *s*-*t*

Current flow

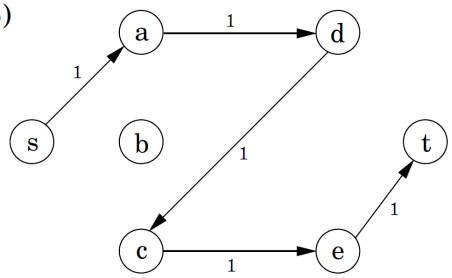
Residual graph

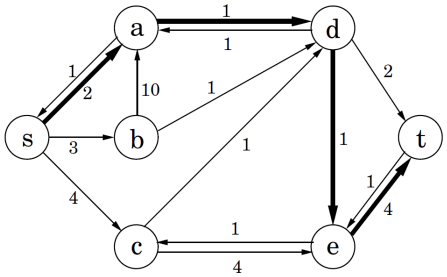
(a)



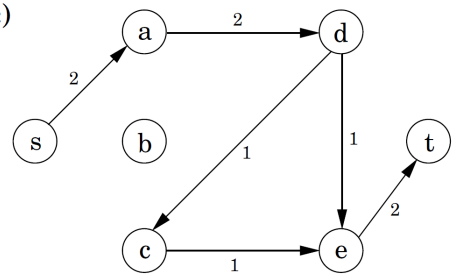


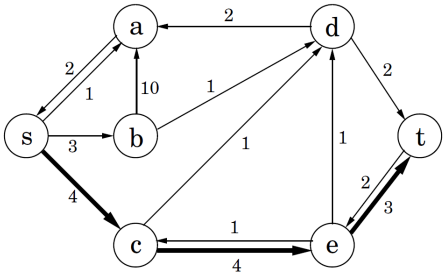
(b)



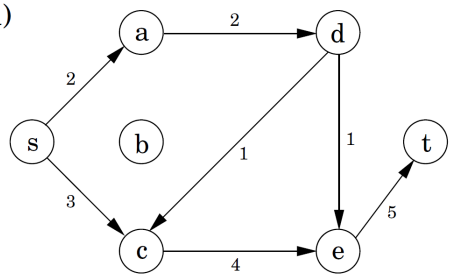


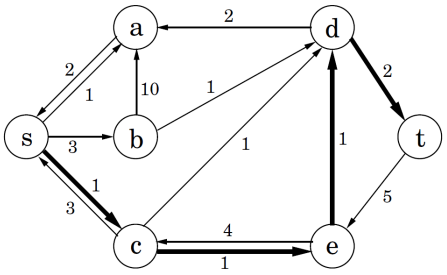
(c)

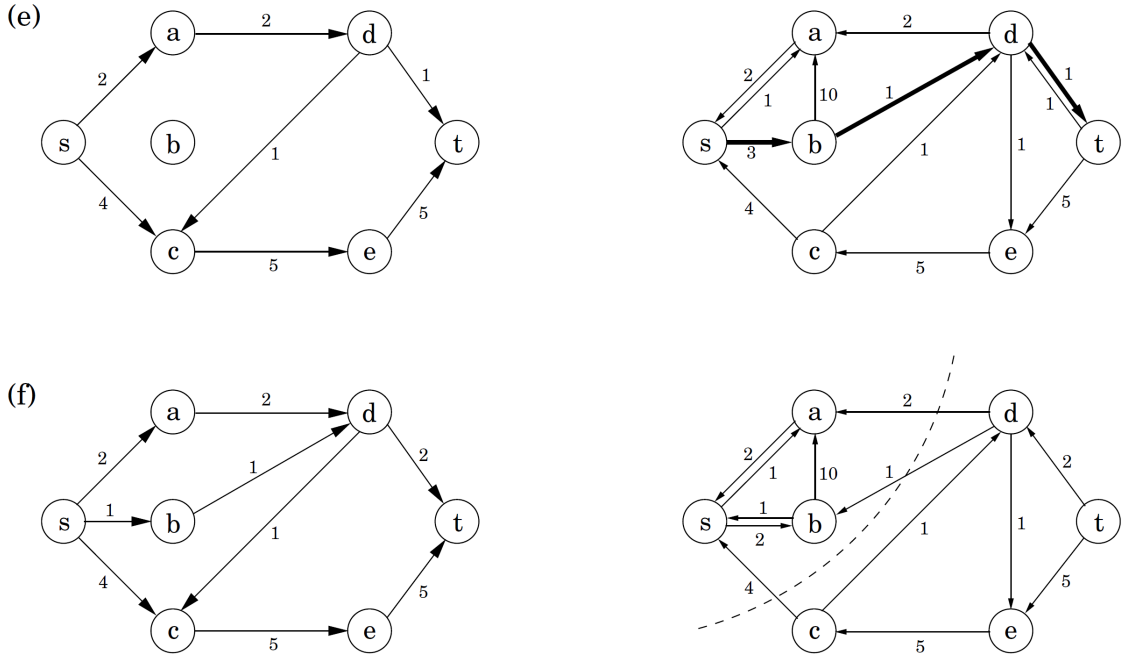




(d)







2. **Verifying a Max-Flow.** Suppose someone presents you with a solution to a max-flow problem on some network. Give a linear time algorithm to determine whether the solution does indeed give a maximum flow.

   Solution:
   First, we verify that the given flow is a valid flow on the network. It is valid if it satisfies the capacity constraint and the flow conservation constraint. Assuming it is a valid flow, note that for every max-flow, there is no $s-t$ path in the residual graph. Therefore, to determine if the given flow is a max-flow, we search for an $s-t$ path in the residual graph.

---

procedure CHECKFLOW(G, f)
Check that $\forall v \in V, v \notin \{s,t\}$: $\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}$
Check that $\forall e \in E, 0 \leq f(e) \leq C(e)$
Compute G^f, the residual flow network of f.
Run BFS(G^f, s)
If BFS finds an s-t path, return false, otherwise return true.
end procedure

Checking the flow conservation condition can be done in $O(|V|+|E|)$ time by iterating over every edge and storing one incoming and one outgoing sum for each node. At edge (u,v), increment node u's outgoing variable and v's incoming variable. After the sums have been computed, iterate over every node and check that the sums are equal. Checking the capacity condition takes $O(|E|)$ time, simply iterate over every edge. Constructing G^f takes $O(|V|+|E|)$ time. Running BFS on G^f takes $O(|V|+|E|)$ time since G^f has $|V|$ vertices and $\leq 2|E|$ edges. Therefore, the algorithm is $O(|V|+|E|)$, which is linear.

3. **Flow Variations.** Show how to reduce the following variations of the max-flow problem to the standard max-flow problem discussed in class.

   (a) There are many sources and many sinks, and we wish to maximize the total flow from all sources to all sinks.

   (b) In addition to the existing edge capacities, each *vertex* has a capacity on the maximum flow that can enter it.

   Solution

   a) Introduce a dummy source node and a dummy sink node. Connect the dummy source node to all source nodes with edges of infinite capacity. Similarly, connect all sink nodes to the dummy sink with edges of infinite capacity. We claim the problem then reduces in maximizing the flow from the dummy source to the dummy sink in this new network. To show the correctness of this approach notice that any valid flow of value f in the original network can be converted into a flow of value f in the new network by routing flow along infinite capacity edges. Similarly, any flow in the new network can be made into a valid flow of the same value in the original network. Hence, the maximum flow must have the same value in both networks.

   b) Construct a new network G' by splitting each vertex v of G into two vertices: v_{in} and v_{out}. Make all edges going into v in G go into v_{in} and all edges leaving v in G leave v_{out} in G'. Finally, insert an edge from v_{in} to v_{out} with capacity equal to the capacity of v in G. Now it is sufficient to solve the maximum flow problem with edge capacities in G'. This works as every valid flow through v in G that satisfies v's capacity constraint can be made into a valid flow through v_{in} and v_{out} in G', and vice versa.

4. **Flow Formulation.** Professor Adam has two children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any sidewalk that the other child has used that day. Assume every street has a sidewalk on either side and the children always walk with the street to their right. Fortunately, the children have no problem with their paths crossing at an intersection, and both the professor's house and the school are at intersections. Given a map of the town, formulate the problem of determining whether both children can go to the same school as a maximum-flow problem.

   Solution:
   We construct the network graph as follows. Create a node for every intersection in town and one directed edge for every sidewalk (two for every street). Let the source and sink nodes be the intersections outside the professor's house and the school, respectively. Because a maximum of one child can use each sidewalk per day, we can set the capacity of each edge to 1. If the maximum flow in this graph is at least two, both children can walk to the same school. If the maximum flow is less than two, the professor will have to send them to separate schools.