CMPSC 465 Fall 2025	Data Structures & Algorithms Ke Chen and Yana Safonova	Worksheet 7 Solution

Monday, Oct 20, 2025

1. **Max Flow Min Cut** Consider the following network (Figure 1, the numbers are edge capacities).

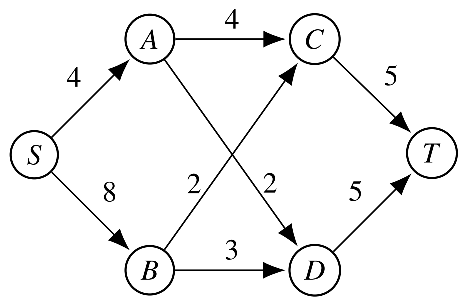

Figure 1: Flow Graph

(a) Find the maximum flow f and a minimum cut using Ford-Fulkerson algorithm.

(b) Draw the residual graph G_f (along with its edge capacities). In this residual network, mark the vertices reachable from S and the vertices from which T is reachable.

Solution

(a) The augmenting paths can be found in the following order: $S-A-C-T$(Bottleneck: 4), $S-B-D-T$(Bottleneck: 3), and $S-B-C-A-D-T$(Bottleneck: 2). The maximum flow is given by the function: $f(S,A)=4, f(S,B)=5, f(A,C)=2, f(A,D)=2, f(B,C)=2, f(B,D)=3, f(C,T)=4, f(D,T)=5$. It produces the mincut $(\{S,B\},\{A,C,D,T\})$ of capacity 9, certifying the optimality of the flow.

(b) The residual graph is shown in the Figure 2. Vertices S and B are reachable from S. T can be reached from all other vertices.

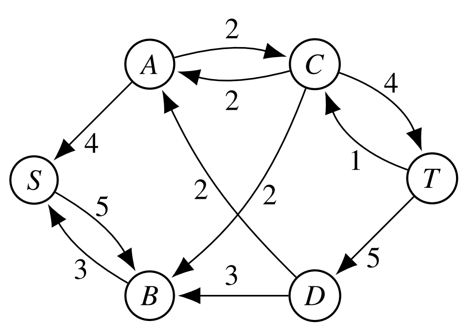

Figure 2: Flow Graph

2. **Max-Flow Extended** Suppose, there are many sources and many sinks, and we wish to maximize the total flow from all sources to all sinks. This problem can be solved efficiently. Show this by reducing it to the original `max-flow` problem.

   Solution

   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 same value in the original network . Hence, the maximum flow must have the same value in both networks. The running time also remains the same as now we only have two extra nodes, and at most $O(|V|)$ extra edges.

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

   Solution The max-flow algorithm has found the maximum flow when there is no $s-t$ path in the residual graph. Therefore, we just search for an $s-t$ path in the residual graph of the given flow to see if the given flow is maximal.

   procedure CHECKFLOW(G, f)
   Check that $\forall v \in V, v \neq s,t$, $\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}$
   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 that f is a valid flow takes $O(|V|+|E|)$ time. Constructing G^f takes $O(|V|+|E|)$ time. Running BFS on G^f takes$O(|V|+|E|)$ time since G^f has $O(|V|)$ vertices and $\leq 2|E|$ edges. Therefore, the algorithm is $O(|V|+|E|)$, which is linear.

4. **Assigning Projects.** A company has n employees and m projects. Each employee is qualified to work on a subset of the projects, and each project requires exactly one employee. However, each employee can be assigned to at most one project. We need to determine whether it is possible to assign employees to projects so that every project is staffed, and if so, find such an assignment. Reduce this problem to max-flow problem and analyze why it is correct.

   Solution: We reduce the problem to a maximum flow instance. Construct a directed flow network $G = (V,E)$ as follows. Create a source node s and a sink node t. From s, draw directed edges to each employee vertex e_i ($1 \leq i \leq n$) with capacity 1. For every project p_j ($1 \leq j \leq m$), draw an edge from p_j to t with capacity 1. For every qualification pair (employee e_i, project p_j) where the employee can work on the project, add a directed edge (e_i, p_j) with capacity 1.

   We then compute the maximum flow from s to t using any standard algorithm, such as Ford-Fulkerson algorithm. If the value of the maximum flow equals m, then there exists an assignment of employees to projects such that each project is filled. The flow of 1 unit along (e_i, p_j) indicates that employee e_i is assigned to project p_j.

Correctness. Each $s \to e_i$ edge with capacity 1 ensures that no employee is assigned to more than one project, while each $p_j \to t$ edge with capacity 1 ensures that no project receives more than one employee. Because flow conservation holds at each intermediate vertex, any feasible integral flow corresponds to a valid assignment of employees to projects, and vice versa. Since all capacities are integers, the Max-Flow Min-Cut Theorem guarantees that there exists an integral maximum flow. Therefore, the maximum flow solution exactly characterizes a valid assignment of maximum possible size.