CSE421: Design and Analysis of Algorithms

Homework 6

Anup Rao Due:

Each problem is worth 10 points:

1. An edge of a flow network is called *critical* if decreasing the capacity of this edge results in a decrease in the maximum flow. Give a polynomial time algorithm that finds a critical edge in a network.

   Solution. The pseudocode is given below.

```
Input: Directed graph D = (V, A)
Result: A critical edge e
Use the capacity scaling algorithm to find the max flow f on D
Use BFS to find the connected component B of s on the residue graph D_f.
for edge e = (u, v) ∈ A do
    if u ∈ B, v ∈ V/B then
        return e
    end
end
```

   Runtime The capacity scaling algorithm runs in polynomial time of $\log C, n$. The construction of the residue graph costs time $O(n^2)$ by creating each edge. The BFS costs time $O(n^2)$. The loop will be executed at most $O(n^2)$ times. In total the run time shall be polynomial.

   Correctness First we want to show that if $B, V/B$ is a min-cut, then any edge $e = (u, v)$ from B to V/B will be critical. This is because the max flow on e must reach its capacity, otherwise e will also appear on the residue graph, and v will be in B then. So if we reduce the capacity of e, then the value of the min-cut will decrease, hence the value of the max flow will also decrease. Then the above algorithm just find such an edge. So the algorithm is correct.

2. Suppose we are given a flow network, where instead of capacities on edges, each internal vertex has a capacity on the total flow that is allowed to pass through it. So for each vertex v, there is a non-negative integer c_v, and the flow must satisfy $f^{in}(v) \leq c_v$. Each edge can carry an arbitrary amount of flow. Give a polynomial time algorithm to find the maximum flow in such a network. (Hint: try to convert the problem into a flow network of the type we are used to.)

   Solution: The polynomial time algorithm will generate a new standard flow network G' as follows.

   (a) G' has a start vertex s and a sink vertex t.

(b) For every vertex u of G with capacity c, G' has two vertices u_0, u_1, and an edge from u_0 to u_1 of capacity c.

(c) For every edge (s, u) in G, we add an edge (s, u_0) to G with infinite capacity.

(d) For every edge (u, v) in G between intermediate vertices, we add an edge (u_1, v_0) to G' of infinite capacity.

(e) For every edge (v, t) in G, add an edge (v_1, t) to G' of infinite capacity.

The algorithm then finds the max-flow in G' using the polynomial time algorithm discussed in class.

To see that the algorithm is correct, we prove that there is a valid flow of value v in G if and only if there is a flow of value v in G'. Suppose there is a flow of value v in G. Obtain a flow in G' by setting the flow value for every edge (u_1, v_0) to be the same as the flow on (u, v) in G, and the flow value for every edge (s, u_0) to be the same as the flow on (s, u) in G, and the flow value for every edge (v_1, t) to be the same as the value on (v, t). Finally set the flow value on every edge (u_0, u_1) to be the same as the total flow into the vertex u. This flow in G' respects the capacities and has the same value as the flow in G.

Suppose there is a flow of value v in G'. Then we obtain a flow in G by setting the flow value of (s, u) to be the same as the flow of (s, u_0), the flow of (u, v) to be the same as the flow on (u_1, v_0) and the flow on (v, t) to be the same as the flow on (v_1, t). Once again, this gives a valid flow that respects the capacity constraints in G'. Thus, the value of the maximum flow in G is the same as the maximum flow in G', and the algorithm is correct.

3. Give a polynomial time algorithm to find a vertex cover of smallest size in a bipartite graph. Hints:

(a) Construct a flow network from the input bipartite graph just as in the maximum matching algorithm.

(b) Show that every min-cut in this flow network gives a vertex cover whose size is the same as the capacity of the cut.

(c) Show that every minimum sized vertex cover in the bipartite graph gives a cut whose capacity is the same as the size of the vertex cover.

(d) Conclude by giving an algorithm to find the smallest vertex cover.

Solution. As in the hint, construct a flow network from the input bipartite graph. Let L and R be the sets of vertices on the left and right, respectively. Compute the min-cut (A, B) of this network. Output $(A \cap R) \cup (B \cap L)$.

Claim 1. *The output is a valid vertex cover.*

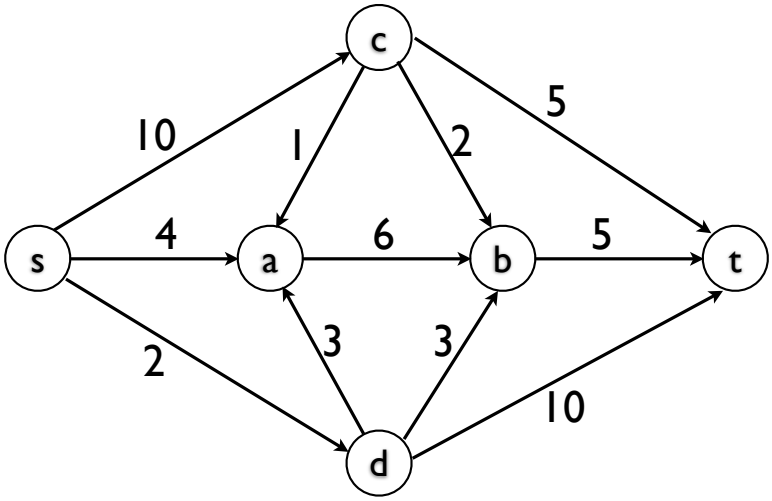
Proof Since the graph is bipartite, every edge goes from L to R. Thus the only edges that can get missed by the algorithm go from $A \cap L$ to $B \cap R$. However, there can be no such edge, since any such edge would make the capacity of the cut A, B infinite. ■

Claim 2. *The output is the optimal vertex cover.*

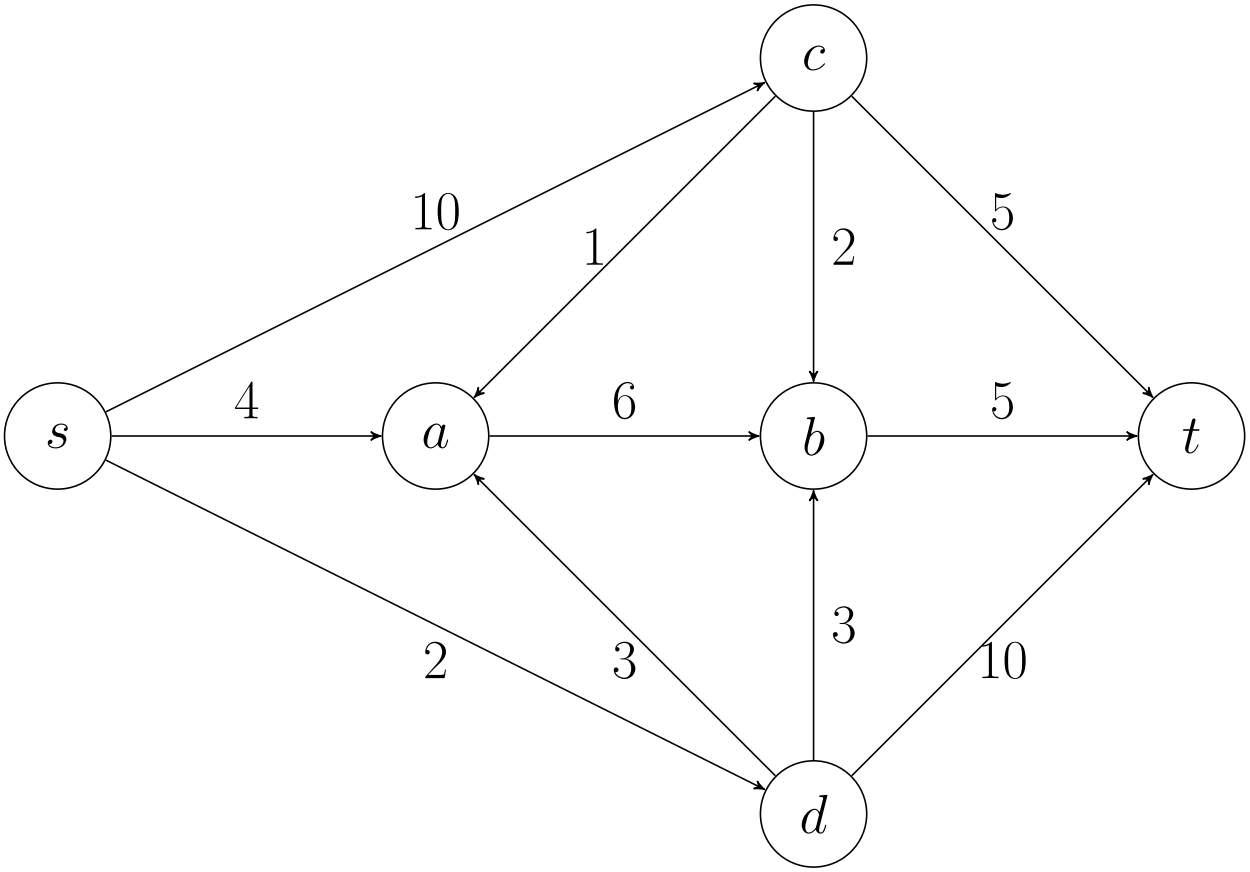
Proof The size of the vertex cover output by the algorithm is equal to $|A \cap R| + |B \cap L|$, which is exactly equal to the capacity of the cut (A, B) in the flow network.

Now let V be any other vertex cover. Then consider the cut $(A', B') = (\{s\} \cup (V \cap R) \cup (V^c \cap L), \{t\} \cup (V^c \cap R) \cup (V \cap L))$. This is clearly a valid cut, and it's capacity is exactly $|V \cap L| + |V \cap R| = |V|$. Thus if there was a smaller vertex cover, we would obtain a smaller cut than (A, B), which is not possible. ■

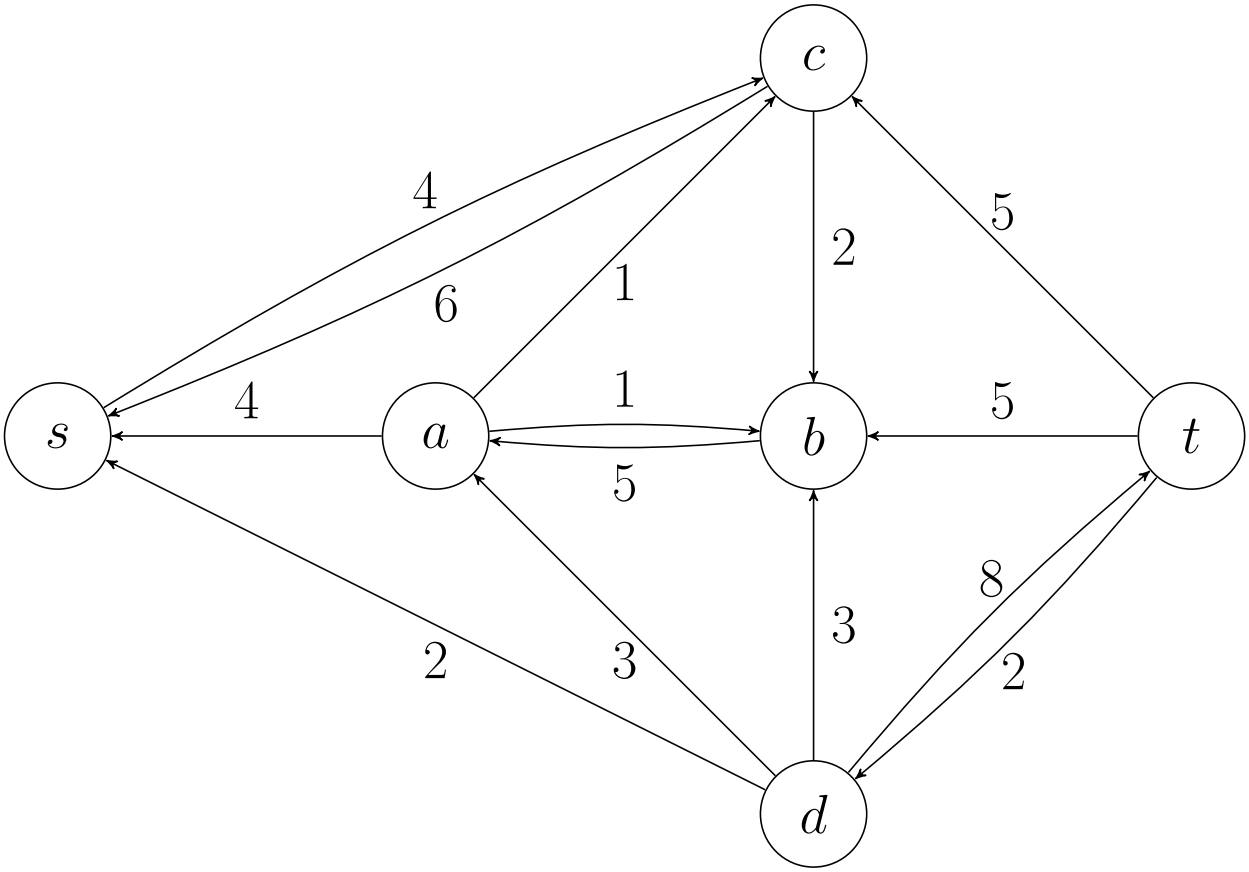
4. Draw out a maximum $s - t$ flow for the graph below, and the corresponding residual graph G_f. What is the minimum cut that corresponds to this max flow?



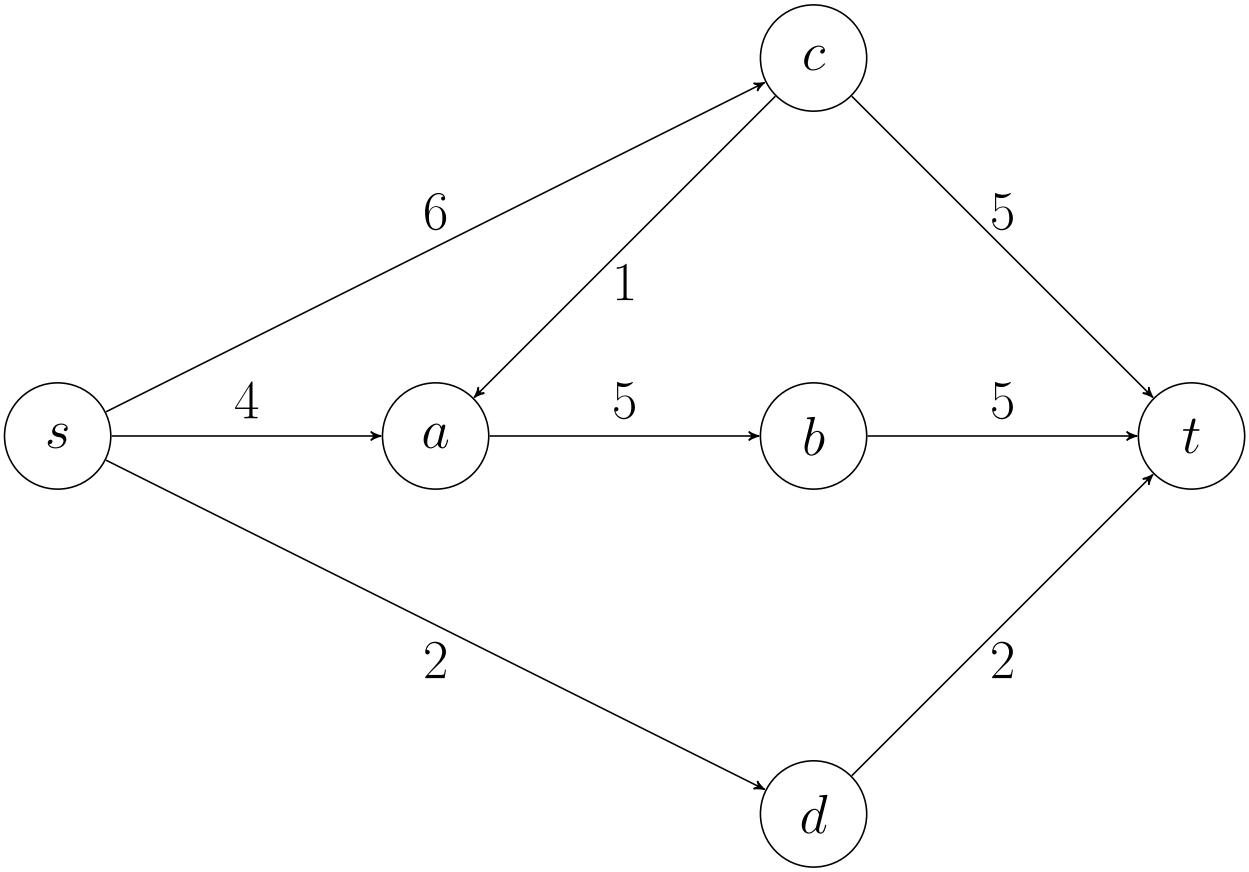
Solution. Original Graph:



Residual Graph:



Max $s - t$ flow:



Min-Cut: $\{s, a, b, c\}, \{d, t\}$. The capacity of the Min-Cut is 12, same as the max-flow.