

Max Flow--Min Cut

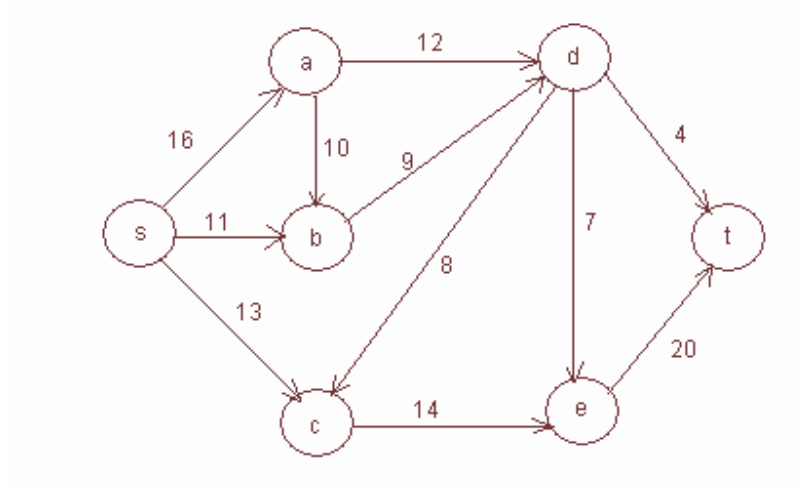


figure 1. The flow network

Let the above graph represent an oil pipeline network where the intermediate vertices represent pumping stations and the edges represent the capacity of the pipe in millions of barrels/day. Vertex s is the source and vertex t is the target refinery. We want to determine the maximum amount of oil that can be delivered from the source to the refinery.

The graph represents a flow network. If $u, v \in V$ are any two vertices in the Graph, $c(u, v)$ is the capacity of the edge from u to v , and $f(u, v)$ is the amount of flow between u and v , then flow networks must obey the following constraints:

$$\begin{aligned}
 &f(u, v) \leq c(u, v) \\
 &f(u, v) = -f(v, u) \\
 &\text{for all } u, v \in V - \{s, t\}
 \end{aligned}$$

$$\sum_{v \in V} f(u, v) = 0$$

Capacity constraint
 Skew symmetry
 Flow into u = flow out

We can represent this graph as an adjacency list.

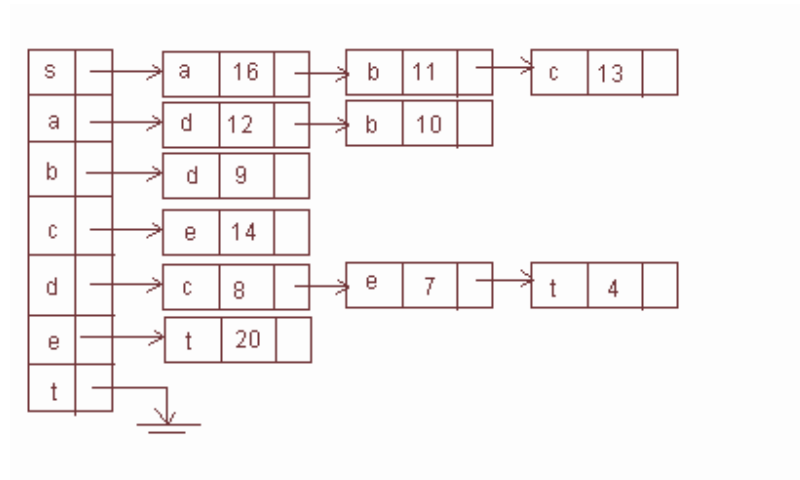


figure 2. The Adjacency list representation of the graph in fig. 1

From this graph we can construct a second graph, G_f , representing the residual capacity of the network after a flow f has been established. Initially the graph G_f is just G , but for each edge (u,v) of capacity $c(u,v)$ carrying flow $f(u,v)$ in G , there are edges (u,v) with capacity $c(u,v) - f(u,v)$, and (v,u) with capacity $f(u,v)$ in G_f . (Let $c_f(u,v) = c(u,v) - f(u,v)$, the residual capacity of the edge (u,v) in G_f . Similarly, $c_f(v,u) = f(u,v)$, the amount of flow that can be "pushed back" on edge (u,v) in G_f . We again depict the residual graph G_f with an adjacency list.

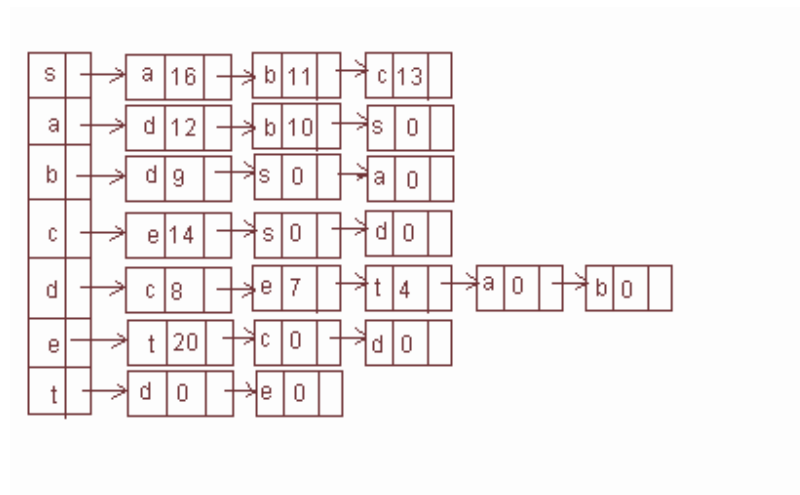


figure 3 Adjacency list representation of G_f

The Ford-Fulkerson algorithm for finding the maximum flow between the source and target is stated in first-level pseudocode as:

```

FordFulkerson(G, s, t) {
  form the residual Graph  $G_f$ 
  //initialize the flow to zero along each edge of G
  for (each edge  $(u,v) \in E[G]$ ) {
     $f(u,v) = 0$ ;
     $f(v,u) = 0$ ;
  }
  while (there exists a path p from s to t in  $G_f$ ) {
     $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is in } p\}$ 
    for (each edge  $(u,v)$  in p)
       $f(u,v) = f(u,v) + c_f(p)$ ;
       $f(v,u) = -f(u,v)$ ;
       $c_f(u,v) = c_f(u,v) - c_f(p)$ ;
       $c_f(v,u) = c_f(v,u) + c_f(p)$ ;
    }
  }
}

```

This first-level description of the Ford-Fulkerson algorithm is short on implementation details. In **while**(there exists a path from s to t in G_f) find the shortest path from s to t using a breadth-first search. Then traverse this path to find the edge with the least remaining capacity, $c_f(p)$. When this value is found, update the flow on each edge in the path and update the capacity of the edges in the residual graph. A breadth-first traversal of G_f in figure 3 first locates the path:

$s \rightarrow a \rightarrow d \rightarrow t$

with $c_f(p) = 4$. After the update, the non-zero edges in the residual graph are shown in figure 4 below.

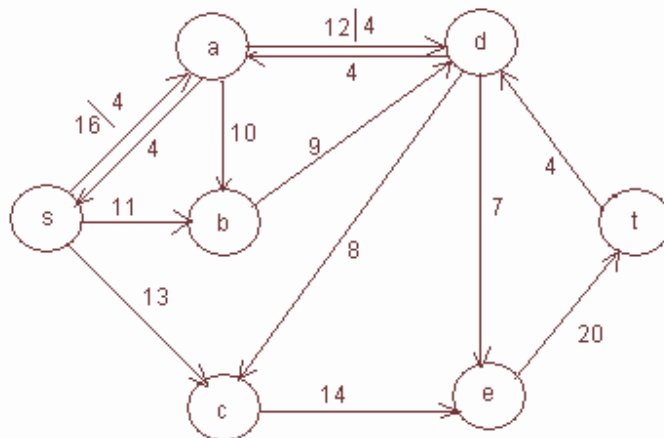


figure 4. G_f after the first iteration

The next iteration of the while loop finds the path: $s \rightarrow c \rightarrow e \rightarrow t$ with $c_f(p) = 13$ and the updated G_f shown in figure 5.

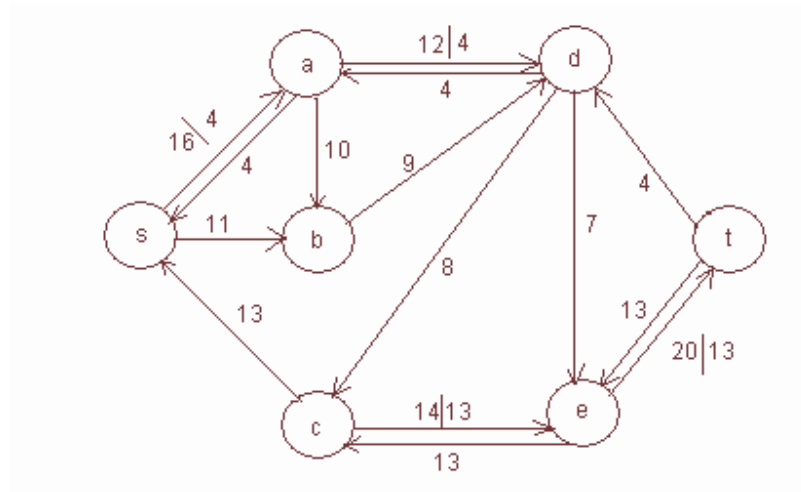


figure 5. G_f after the second iteration

The third iteration of the while loop finds the path: $s \rightarrow b \rightarrow d \rightarrow e \rightarrow t$ with $c_f(p) = 7$. The updated G_f is shown in figure 6.

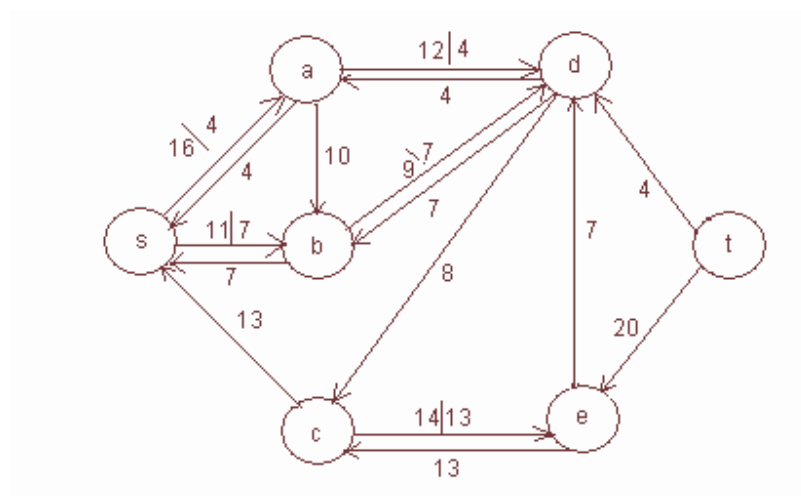


figure 6. G_f after the third iteration

There is no other path from s to t in the residual graph, and the algorithm terminates. The maximum flow from s to t is the sum of the three flows = 24.

Min-Cut

A cut is formed by partitioning the vertices into two subsets -- X some subset of V containing the source, and X_c , the complement of X containing the sink. In the graph depicted above one cut is formed by $X = \{s, a, b, c\}$ and $X_c = \{d, e, t\}$. The cut $C(X, X_c) = \{(a, d), (b, d), (d, c), (c, e)\}$, the set of edges with one vertex in X and the other in X_c . A minimum cut is the cut with the smallest capacity.

Theorem The value of the maximum flow in a network is equal to the capacity of the minimum cut.

Proof:

Let f be a feasible flow of value v and $C(X, X_c)$ be a cut of capacity c in a network. Consider the flow across this cut defined as the difference between the sum of the flows on the edges from X to X_c and from X_c to X .

$$v = \sum_{i \in X, j \in X_c} f(i, j) - \sum_{j \in X_c, i \in X} f(j, i) \quad (1)$$

The second sum is non-negative, hence

$$v \leq \sum_{i \in X, j \in X_c} f(i, j) \leq \sum_{i \in X, j \in X_c} c(i, j) \quad v \leq \text{the capacity of the edges from } X \text{ to } X_c$$

$$v \leq c$$

Thus, the value of any feasible flow cannot exceed the capacity of any cut in the network. Let v^* be the value of a final flow, f^* , obtained by the Fulkerson-Ford augmenting path method. If we now find a cut whose capacity is equal to v^* , we will have to conclude that (i) the value v^* is maximal among all feasible flows; (ii) the cut's capacity is minimal among all cuts in the network; (iii) the maximum-flow value is equal to the minimum-cut capacity.

To find such a minimum-cut, consider the set of vertices X^* that can be reached from the source by following an undirected path composed of forward edges with positive unused capacities (with respect to the final flow f^*) and backward edges with positive flows on them. This set does not contain the sink, for if it did, we would have an augmenting path for the flow f^* , which would contradict the assumption that the flow f^* is final. Consider the cut $C(X^*, X_c^*)$. By the definition of set X^* , each edge (i, j) from X^* to X_c^* has zero unused capacity, and each edge (j, i) from X_c^* to X^* has zero flow on it (otherwise, j would be in X^*). Applying (1) to the final flow f^* and the set X^* we obtain

$$v^* = \sum_{i \in X^*, j \in X_c^*} f^*(i, j) - \sum_{j \in X_c^*, i \in X^*} f^*(j, i) = \sum_{i \in X^*, j \in X_c^*} c(i, j) - 0 = c(X^*, X_c^*)$$

which proves the theorem.