MaxFlow & MinCut

A project presented by **Group Titanic**

Group Members

Aisulu Bakhtybayeva

Ramazan Irmagaliev

Aruzhan Medetova

Amirzhan Zhuvandykov

Aisaule Issakulova

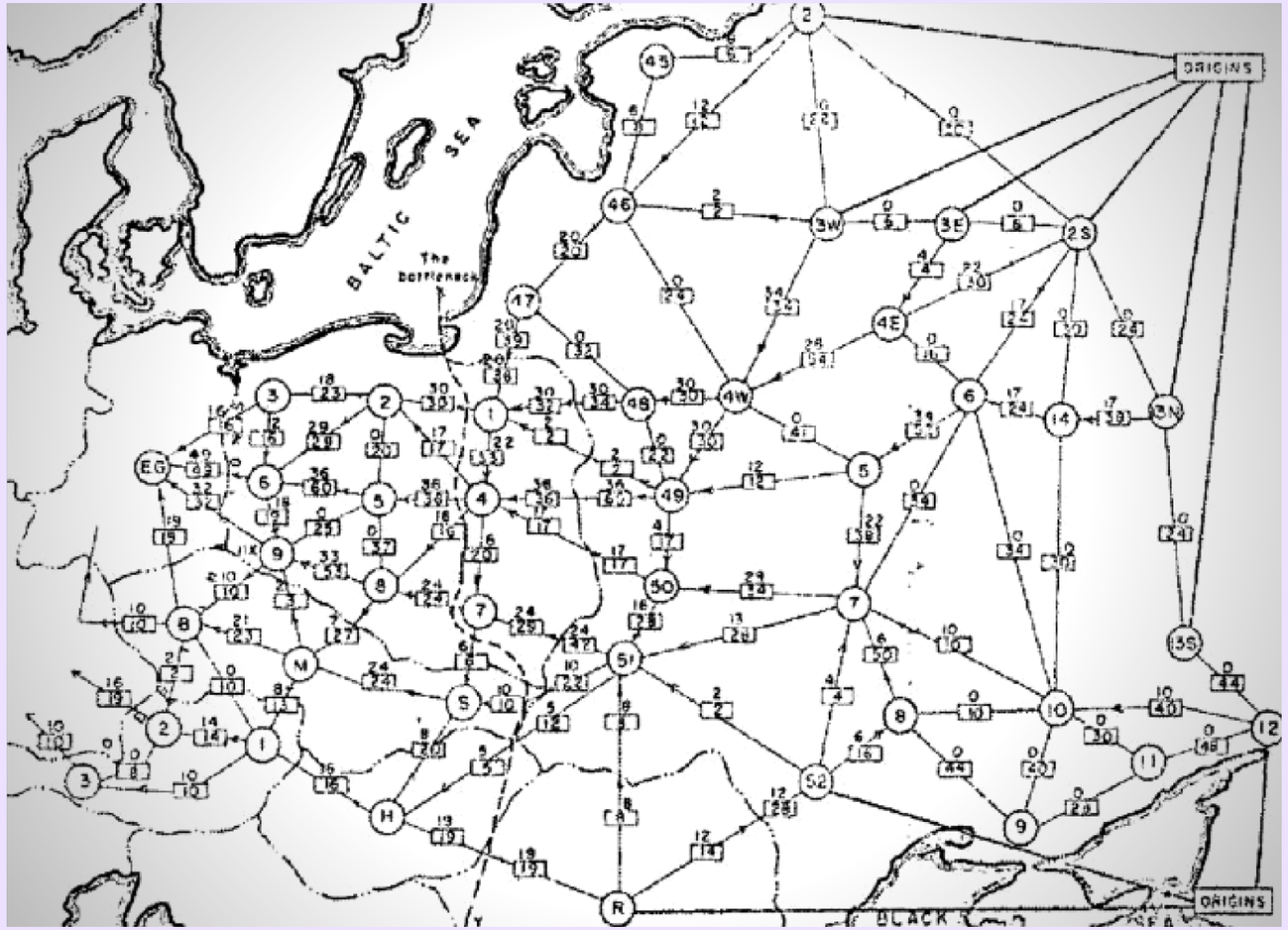
Introduction

Maximum Flow & Minimum Cut

- Two very rich algorithmic problems.
- Cornerstone problems in combinatorial optimization.
- Beautiful mathematical duality

- A directed, weighted graph is called a (flow) network.
- Each edge has a weight and direction.
- We assume there exists a source and a sink.

Soviet Rail Network, 1955



Source: On the history of the transportation and maximum flow problems.
Alexander Schrijver in Math Programming, 91: 3, 2002.

Nontrivial applications / reductions

- Network connectivity
- Bipartite matching
- Data mining
- Open-pit mining
- Airline scheduling
- Image processing
- Project selection
- Baseball elimination
- Network reliability
- Security of statistical data
- Distributed computing
- Egalitarian stable matching
- Distributed computing
- Many many more

Cuts

1. A cut is a partition of the vertices into disjoint subsets S and T. In a flow network, the source is located in S, and the sink is located in T.
2. The cut-set of a cut is the set of edges that begin in S and end in T.
3. The capacity of a cut is sum of the weights of the edges beginning in S and ending in T.

Flows

1. The flow over a network is a function f: E -> R, assigning values to each of the edges in the network which are nonnegative and less than the capacity of that edge.
2. For each intermediate vertex, the outflow and inflow must be equal.
3. The value of this flow is the total amount leaving the source (and thus entering the sink)

The Problem

Minimum Cut Problem

- Network: abstraction for material FLOWING through the edges.
- Directed graph.
- Capacities on edges.
- Source node s, sink node t.

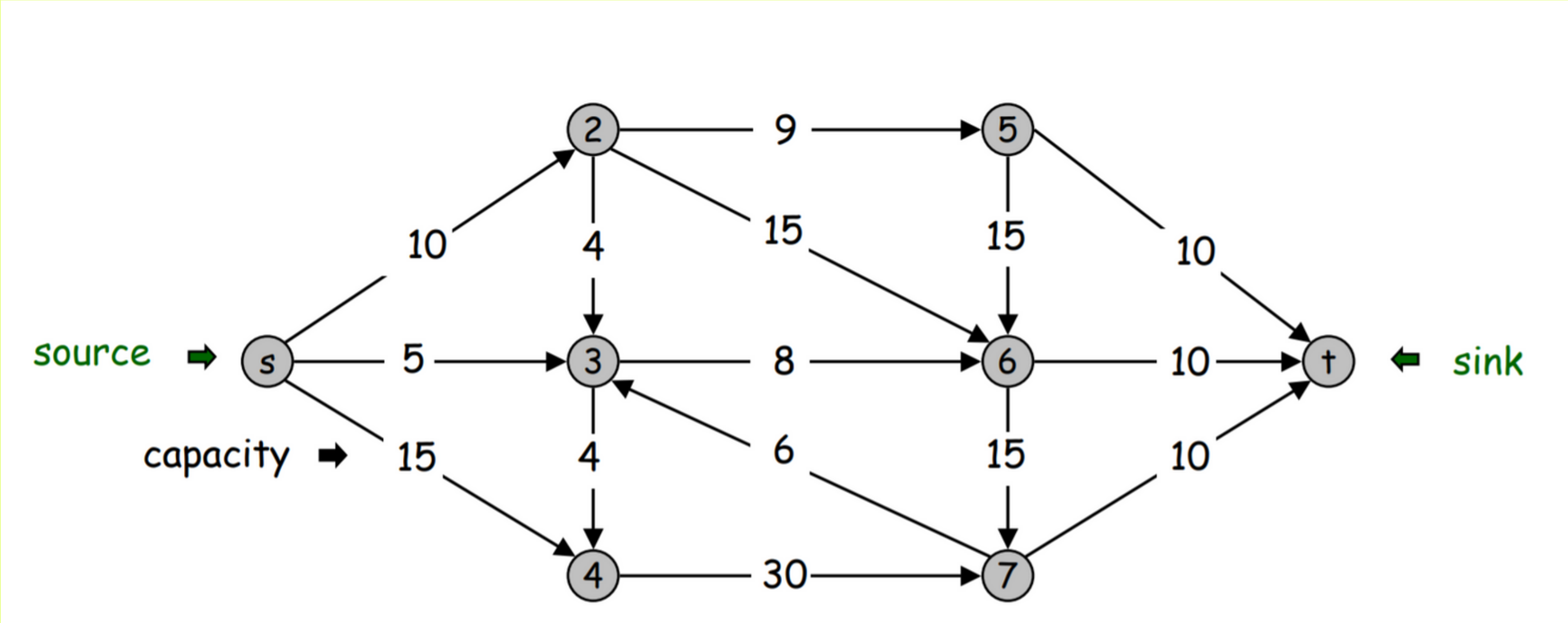
Maximum Flow Problem

Network: abstraction for material FLOWING through the edges.

- Directed graph.
- Capacities on edges.
- Source node s, sink node t.

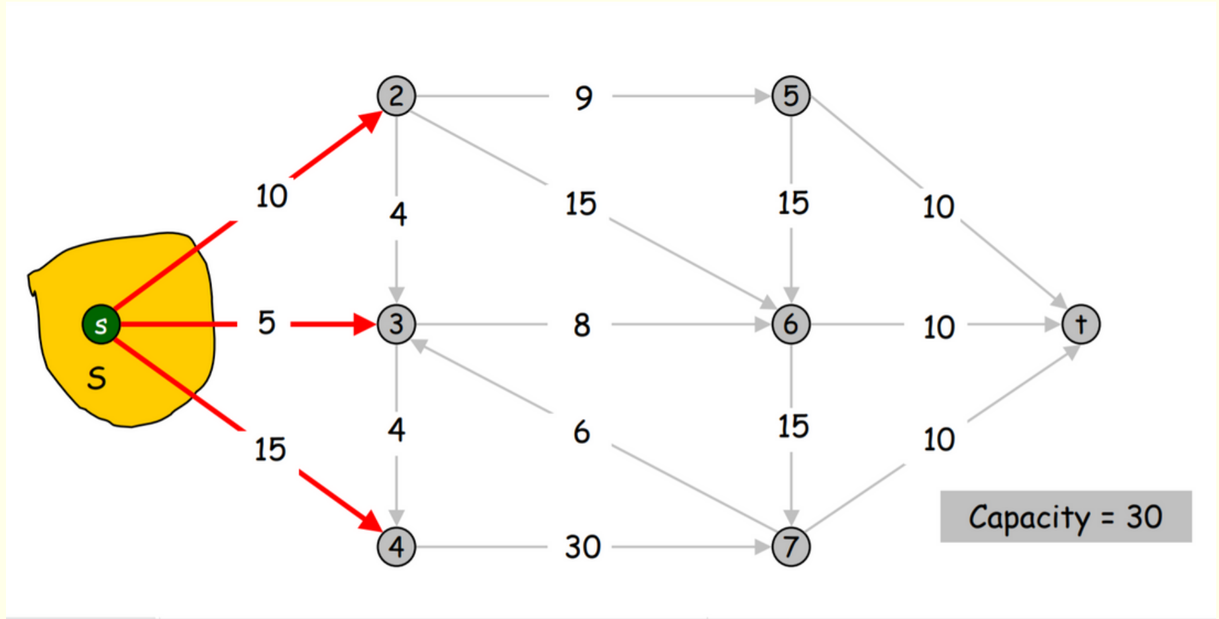
Min cut problem

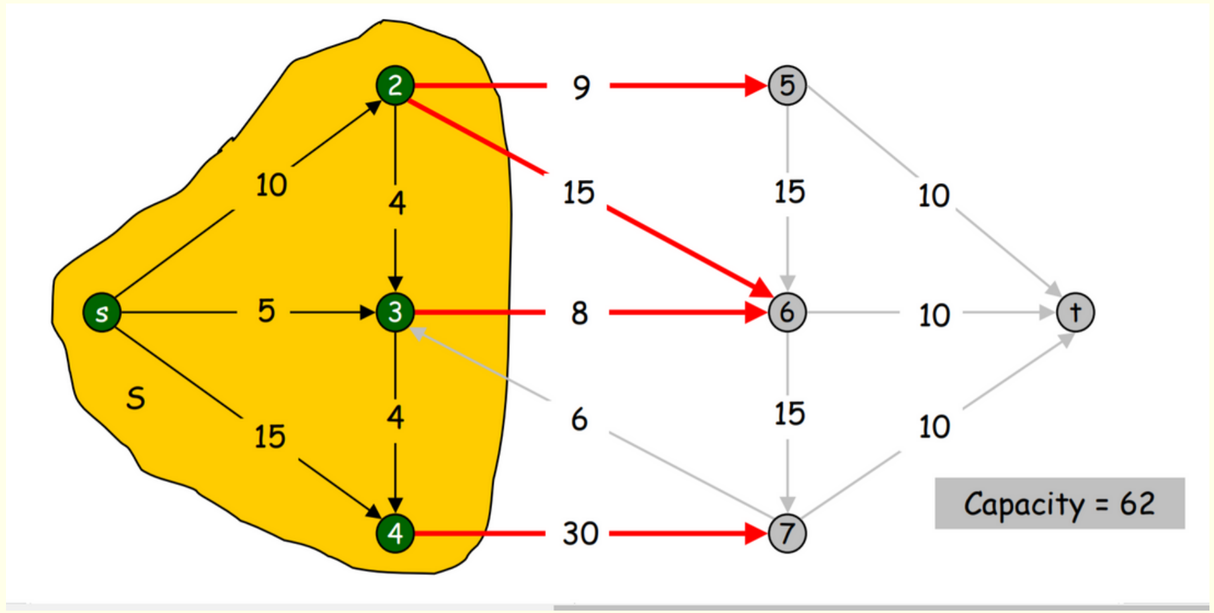
Delete "best" set of edges to disconnect t from s.



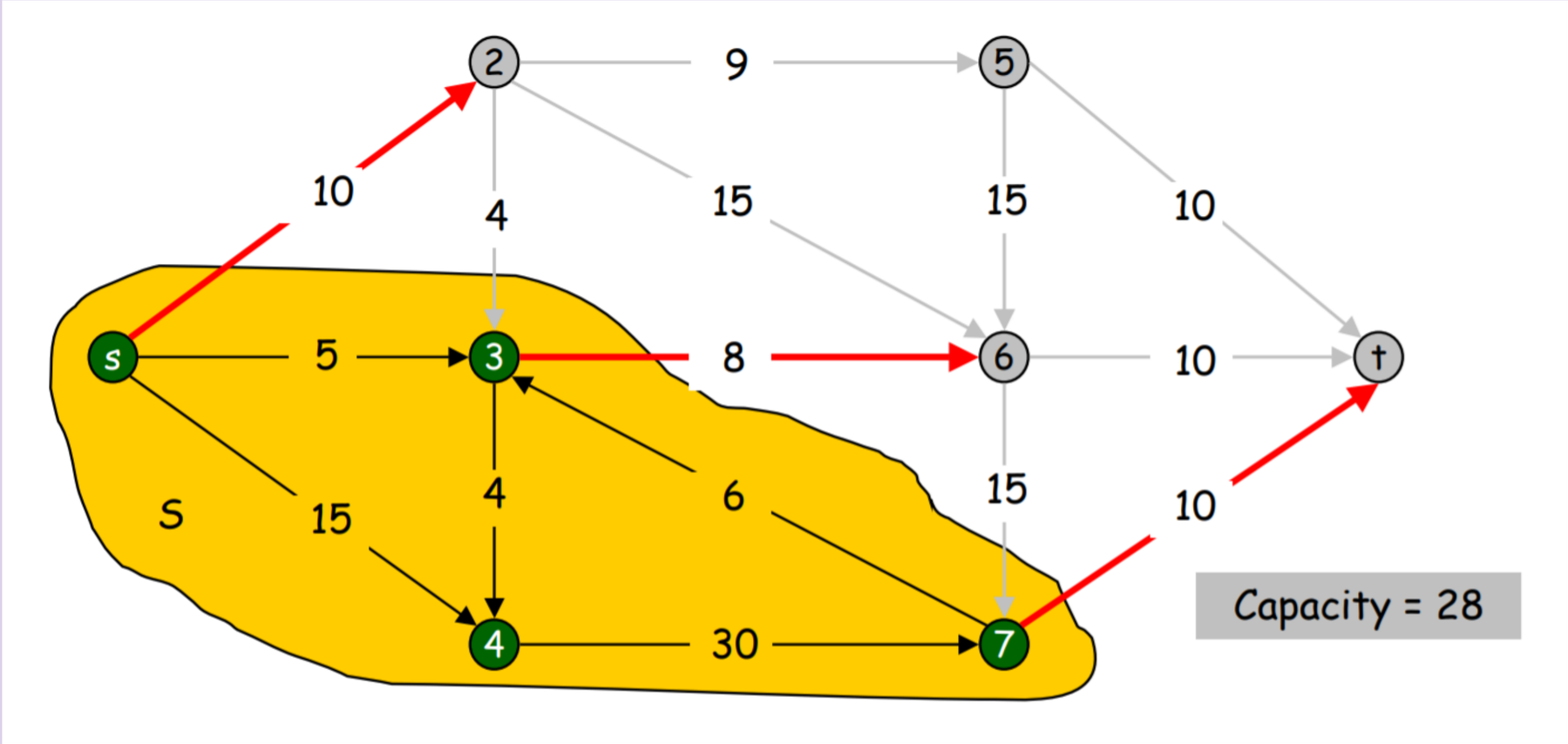
A cut is a node partition (S, T) such that s is in S and t is in T.

- capacity(S, T) = sum of weights of edges leaving S.



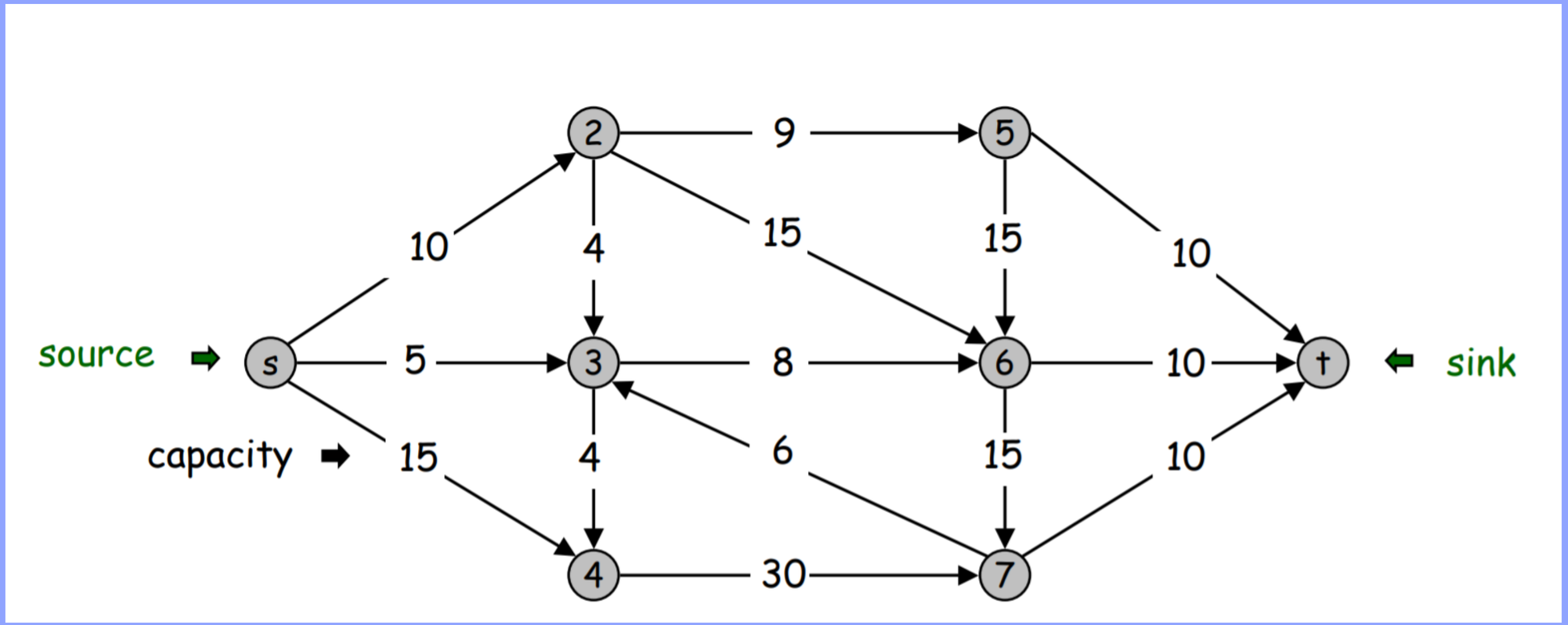


Min cut problem. Find an s-t cut of minimum capacity



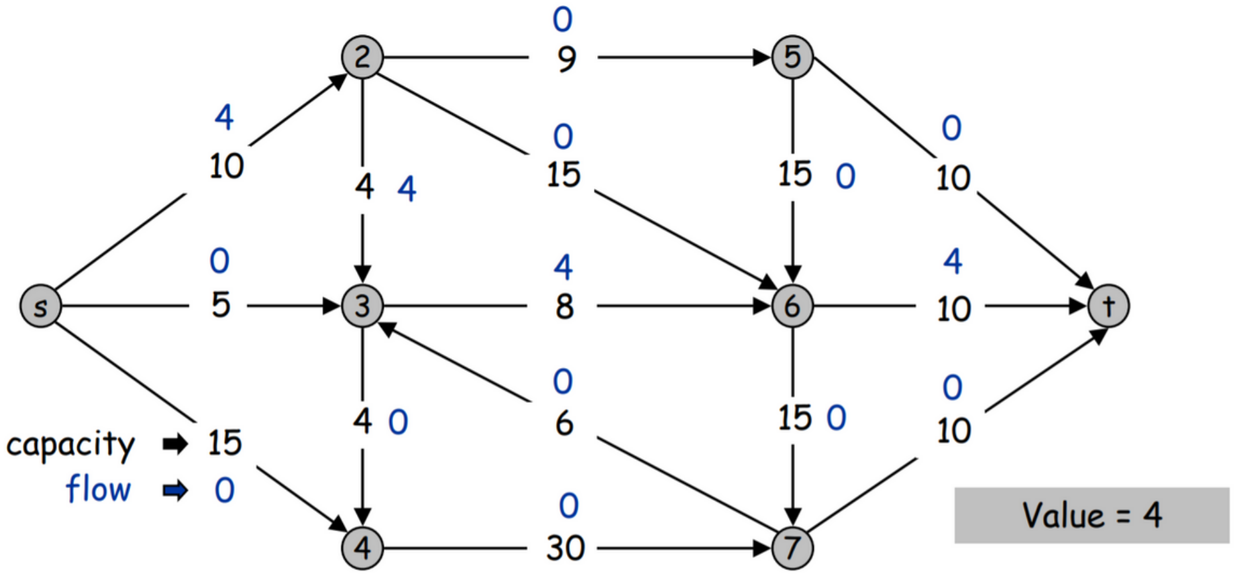
Max flow problem

Assign flow to edges so as to



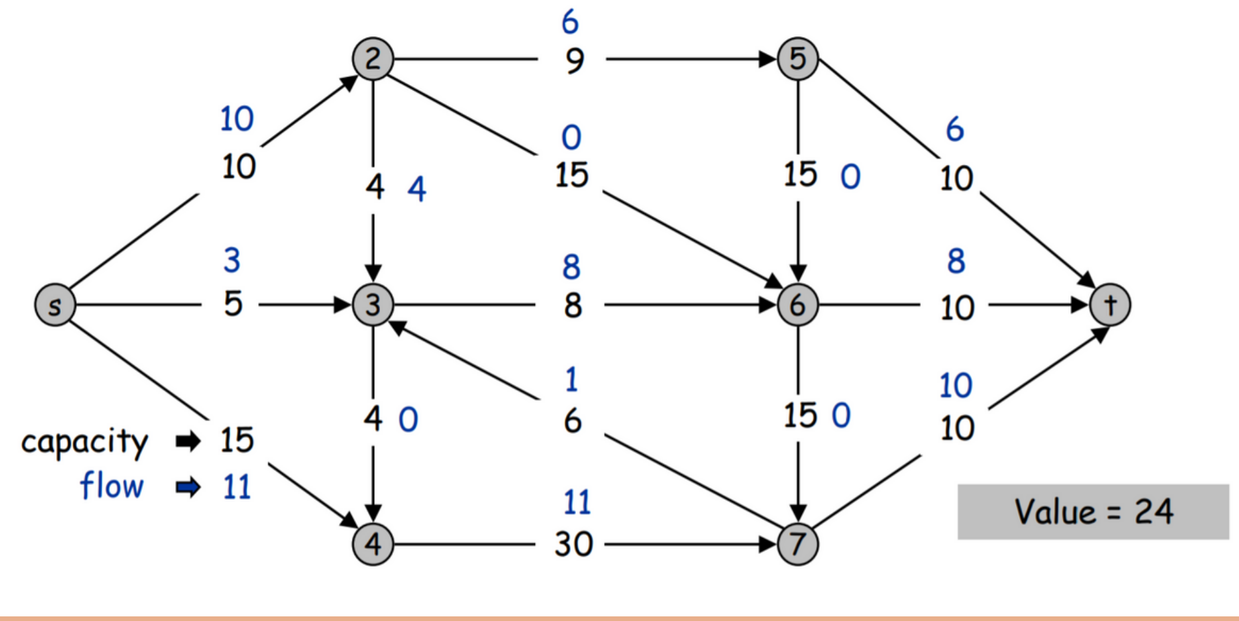
Equalize inflow and outflow at every intermediate vertex.

- Maximize flow sent from s to t.

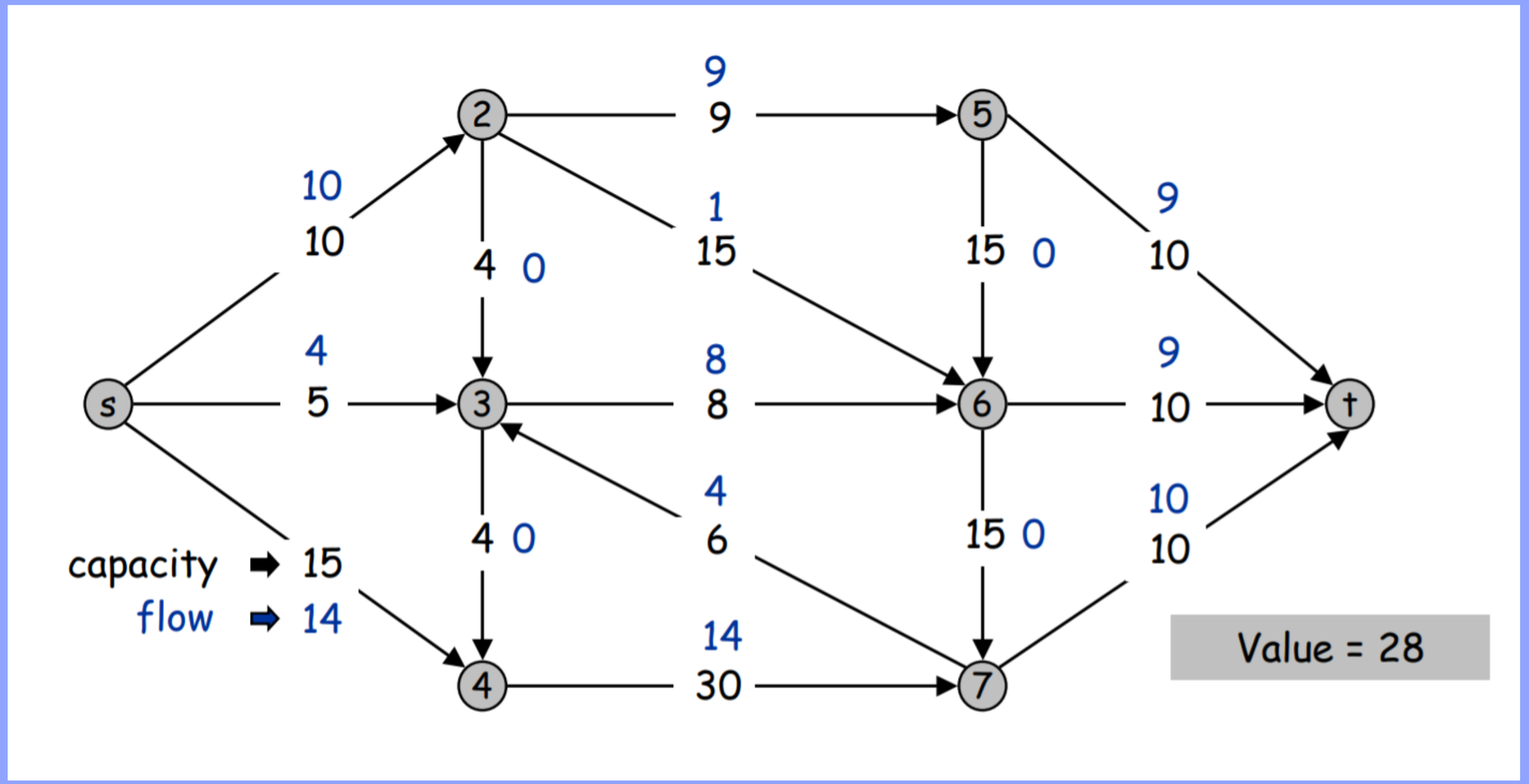


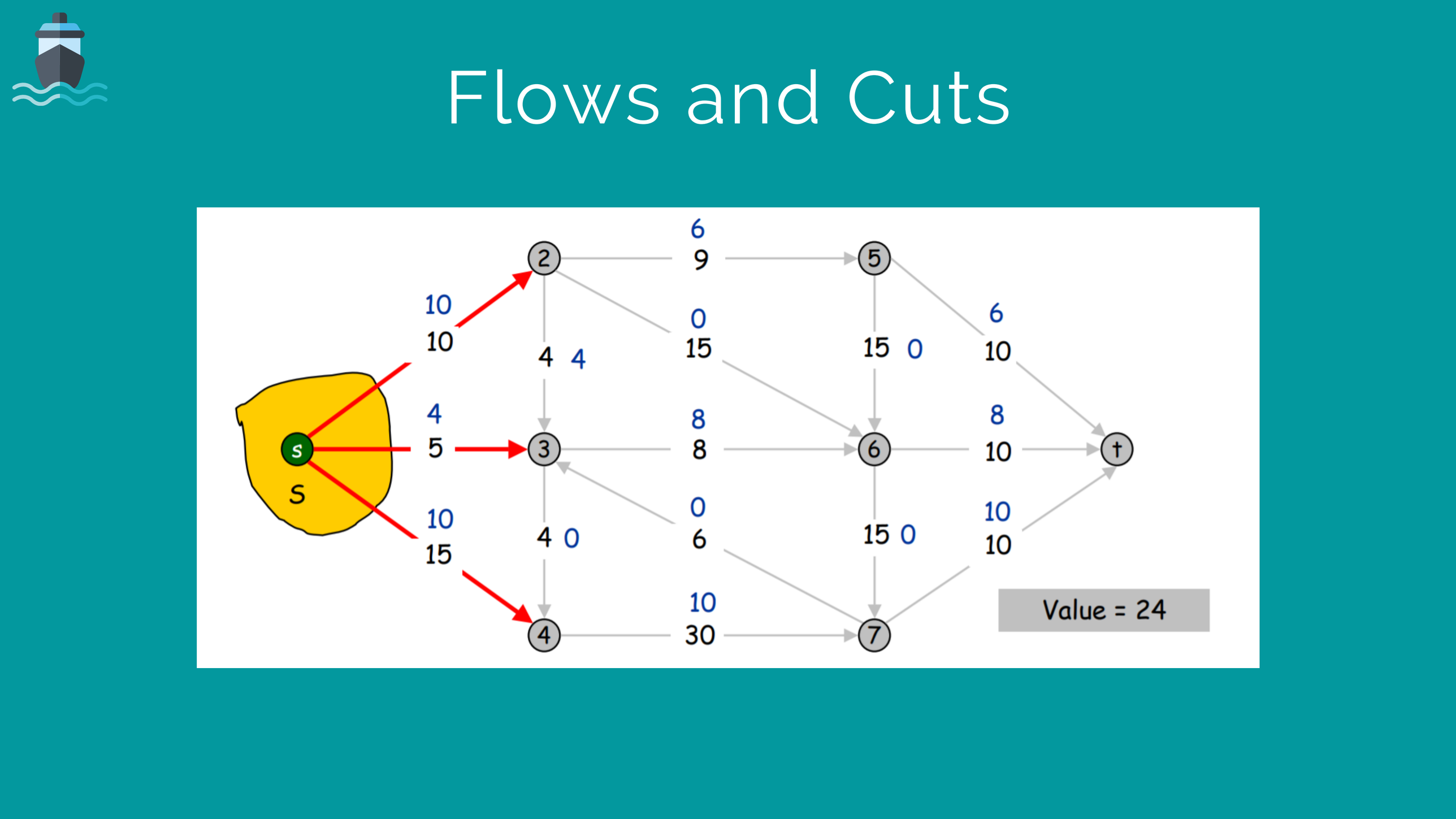
A flow f is an assignment of weights to edges so that:

- Capacity: 0 f(e) u(e).
- Flow conservation: flow leaving v = flow entering v

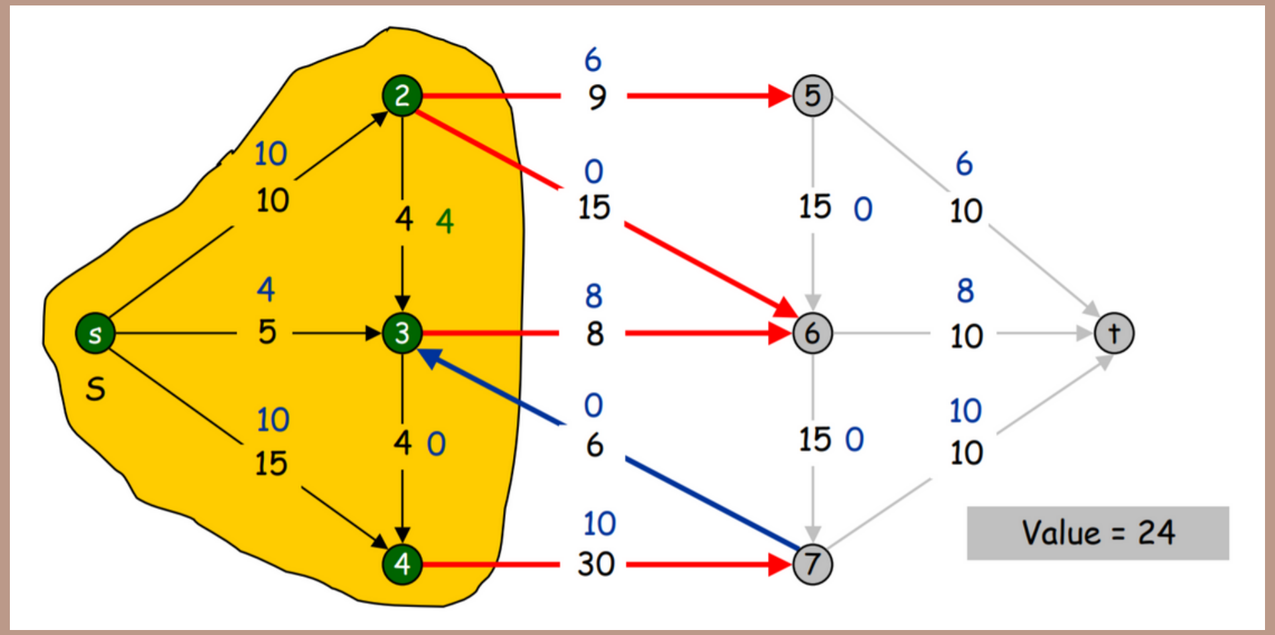


Max flow problem: find flow that maximizes net flow into sink



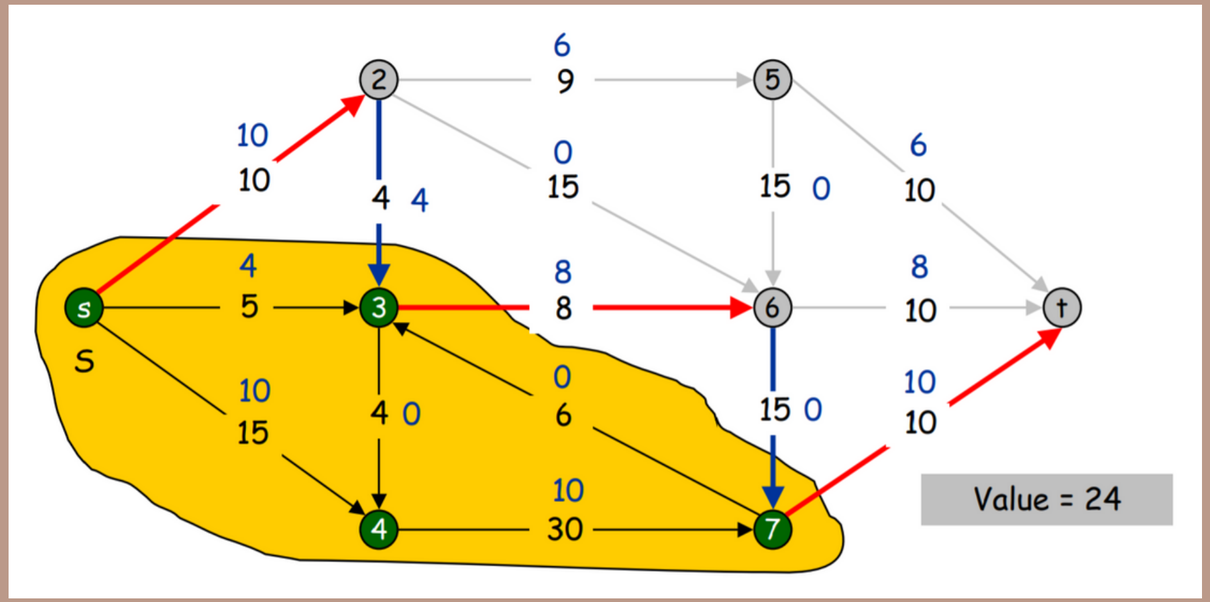


Flows and Cuts

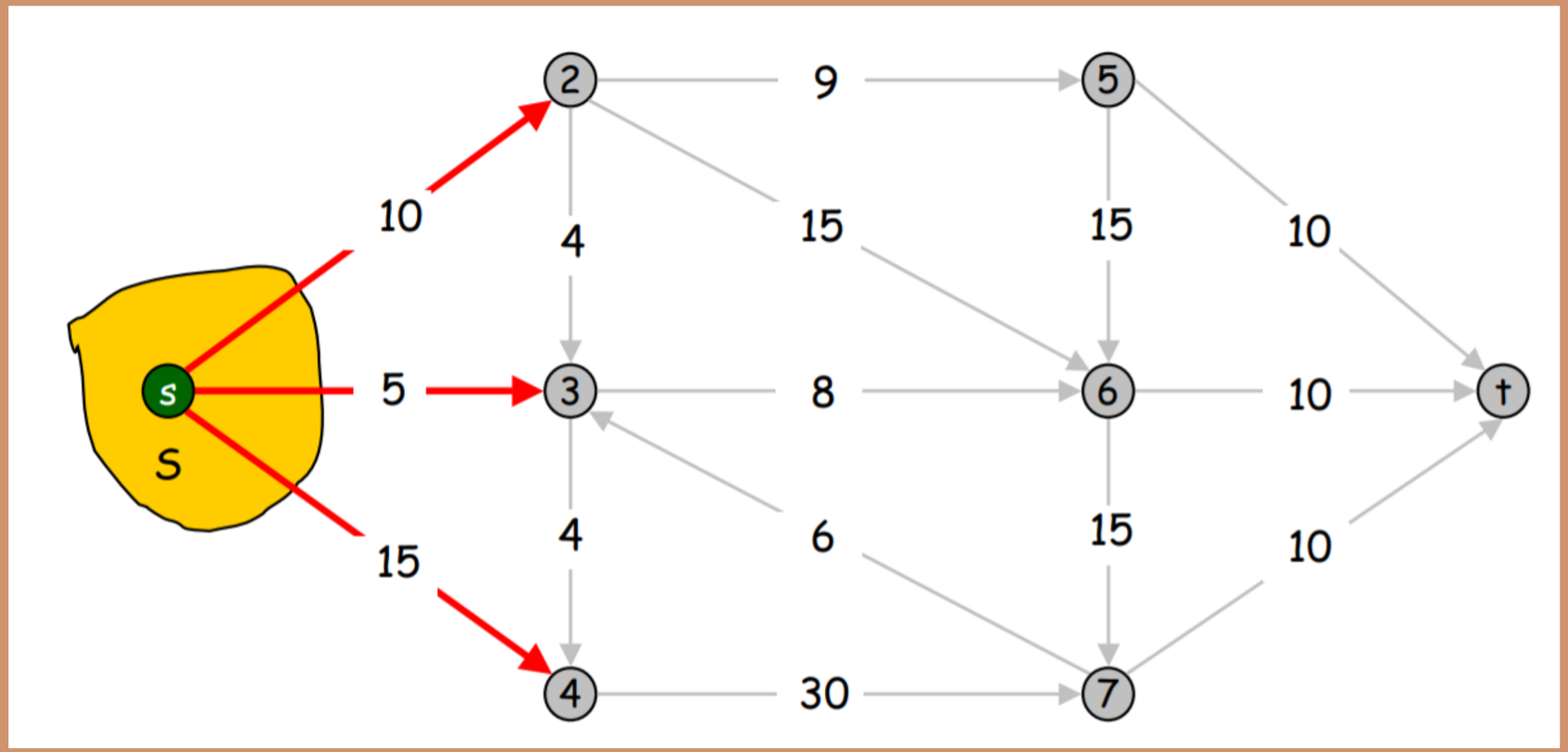


Let f be a flow, and let (S, T) be any s-t cut

Then, the net flow sent across the cut is equal to the amount reaching t.

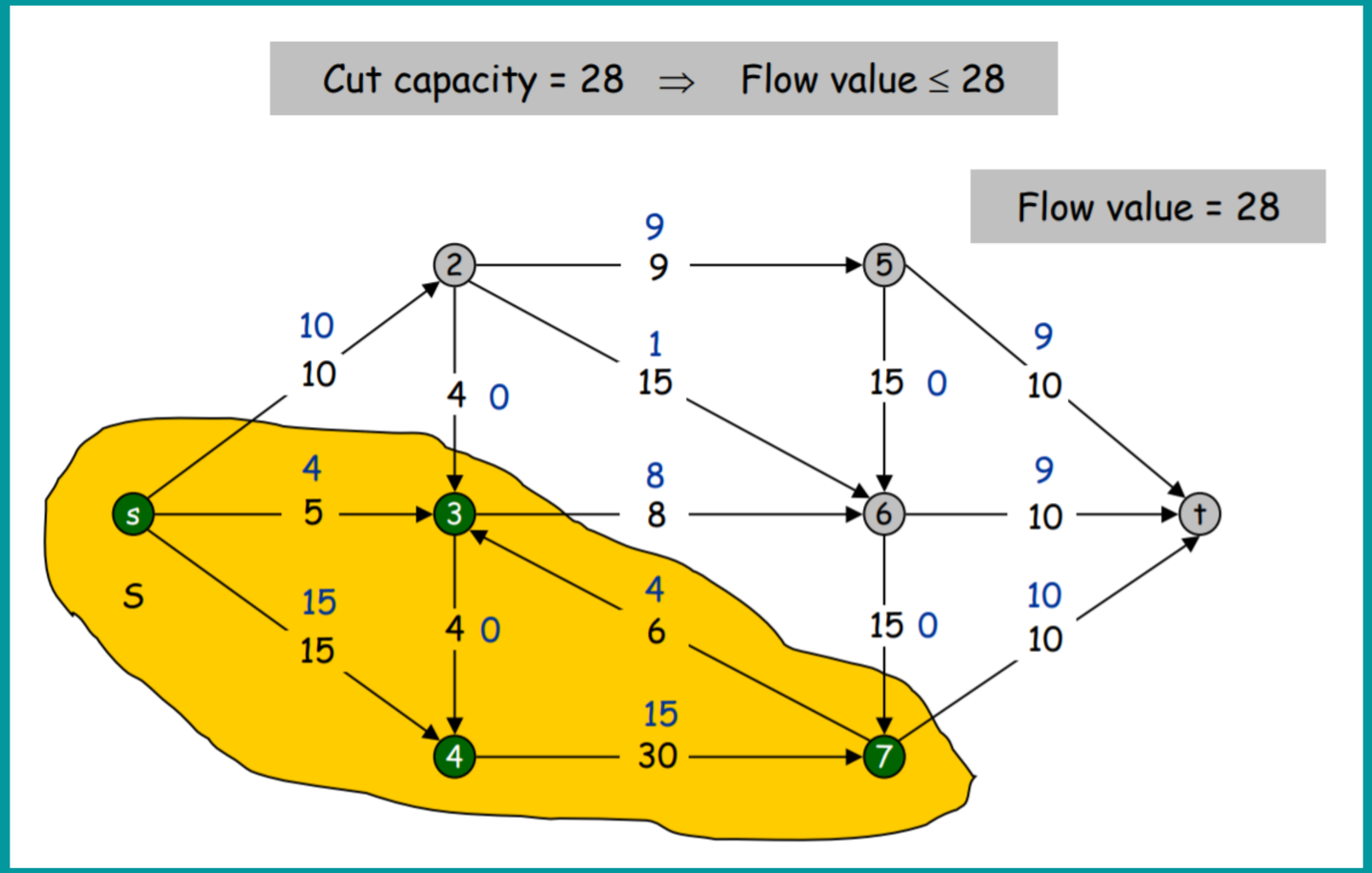


Let f be a flow, and let (S, T) be any s-t cut. Then the value of the flow is at most the capacity of the cut.



Cut capacity = 30 => Flow value ≤ 30

Let f be a flow, and let (S, T) be an s-t cut whose capacity equals the value of f. Then f is a max flow and (S, T) is a min cut.



THEOREM

Max-flow min-cut theorem

Theorem: For any network, the value of the maximum flow is equal to the capacity of the minimum cut.

We'll describe the max-flow min-cut theorem and present an algorithm to find the maximum flow of a graph.

There are many specific algorithms that implement this theorem in practice. The most famous algorithm is the Ford-Fulkerson algorithm, named after the two scientists that discovered the max-flow min-cut theorem in 1956.

The Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm for finding the maximum flow:

a — Construct the Residual Graph

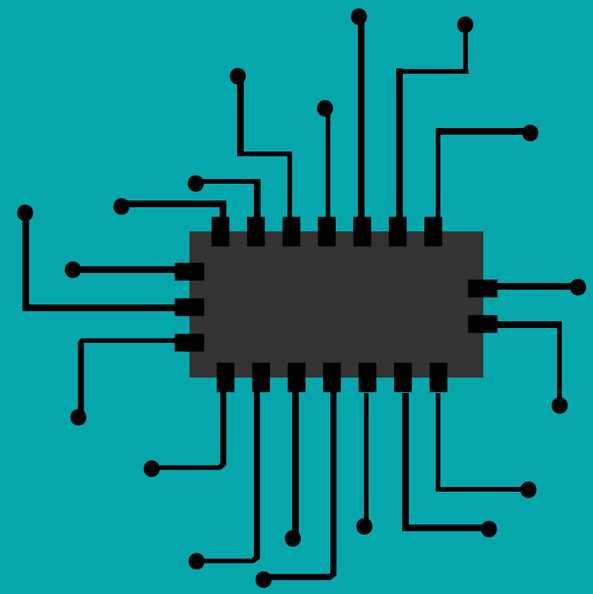
b — Find a path from the source to the sink with strictly positive flow

c — If this path exists, update flow to include it. Go to Step a

d — Else, the flow is maximal

e — The (s,t)-cut has as S all vertices reachable from the source, and T as V - S

The algorithm starts with a workable flow through the graph, and the flow is improved iteratively.
If this flow is maximum, it makes it possible to determine the flow function satisfying this value as well as the minimum cut.
If the flow is not maximum, its objective is to highlight an improving path corresponding to this flow.



Initially, the algorithm starts by setting the flow value between the source and sink node to . At each iteration, we find an augmented path and increase the flow value. We'll terminate the algorithm and return the flow value when no more augmented paths can be found.

Let's see the pseudocode of the Ford-Fulkerson algorithm:

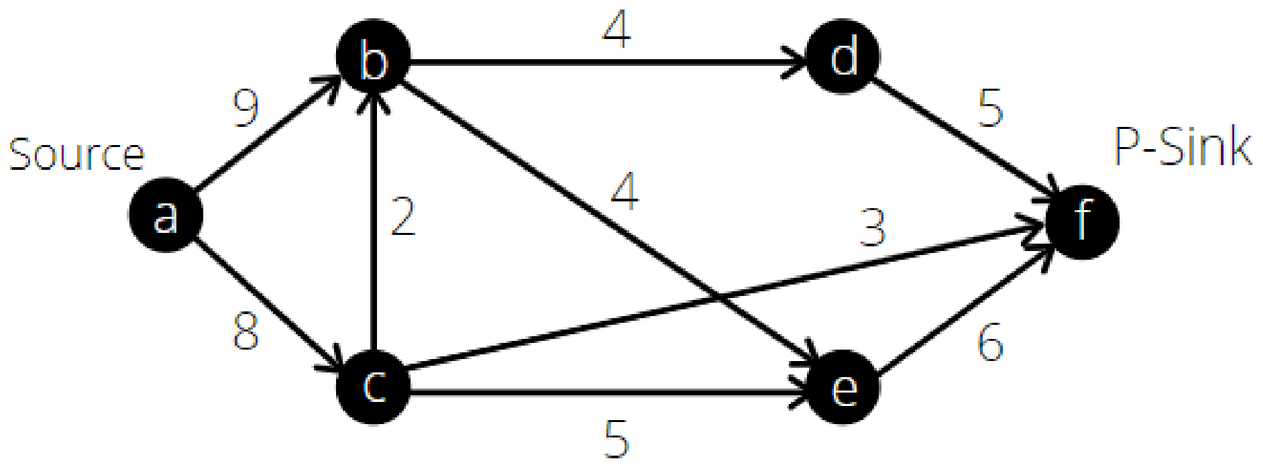
Algorithm 1: The Ford-Fulkerson algorithm for finding maximum flow of a graph

Data: G: a directed connected graph, s: source node, t: sink node
Result: Maximum flow of the graph G
Procedure Ford-Fulkerson(B, s, t)

```
for each edge (u, v) ∈ E(G) do
    f(u, v) ← 0;
    f(v, u) ← 0;
end
while ∃ a augmenting path p ∈ G_f do
    c_f(p) ← min{c_f(u, v) : (u, v) ∈ p};
    for each edge (u, v) ∈ p do
        f(u, v) ← f(u, v) + c_f(p);
        f(v, u) ← −f(u, v);
    end
end
```

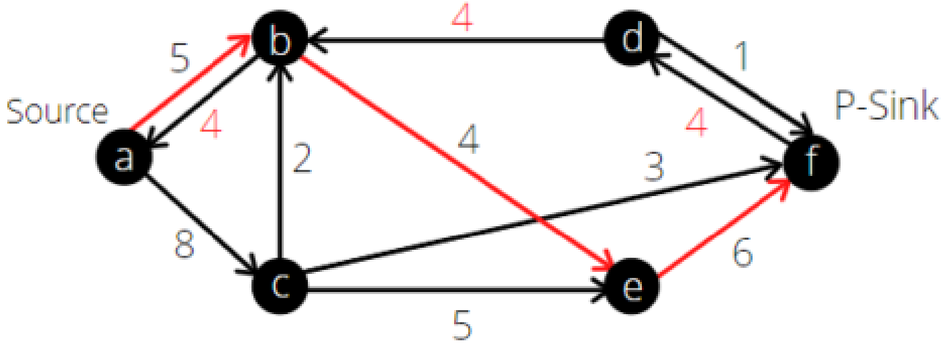
Example

Initially, we're taking a directed connected graph, and we'll run the Ford-Fulkerson algorithm on it. In each step, we pick an augmented path and present the residual graph and Netflow graph:

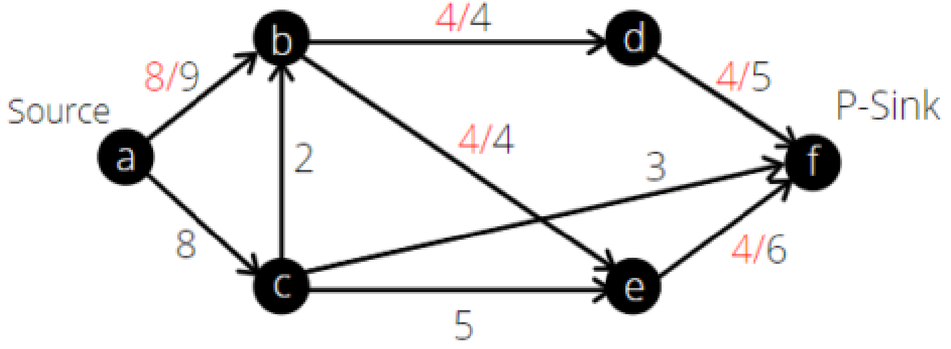


First, we choose the path a-b-d-f. In this path, the minimum capacity is 4. Now we'll construct a residual graph and a Netflow graph:

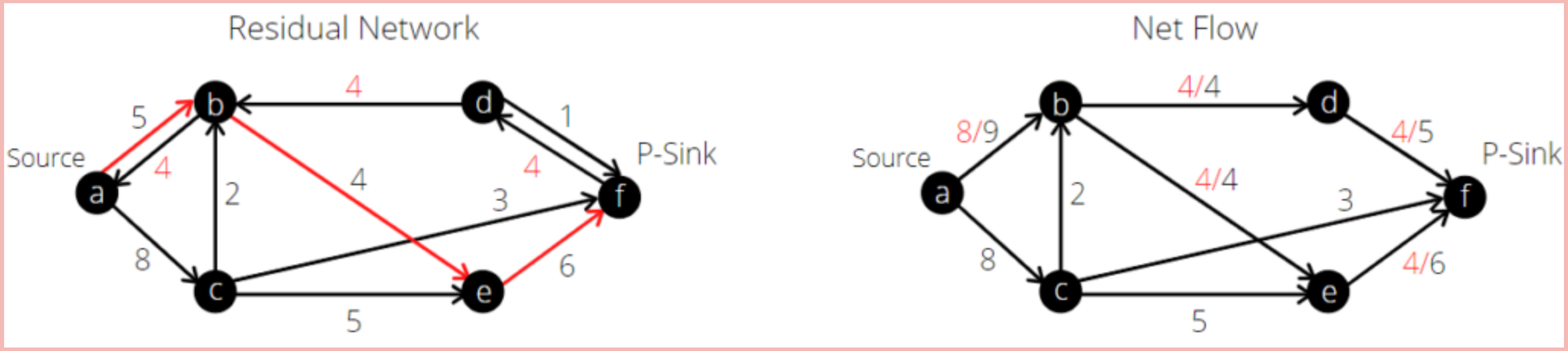
Residual Network



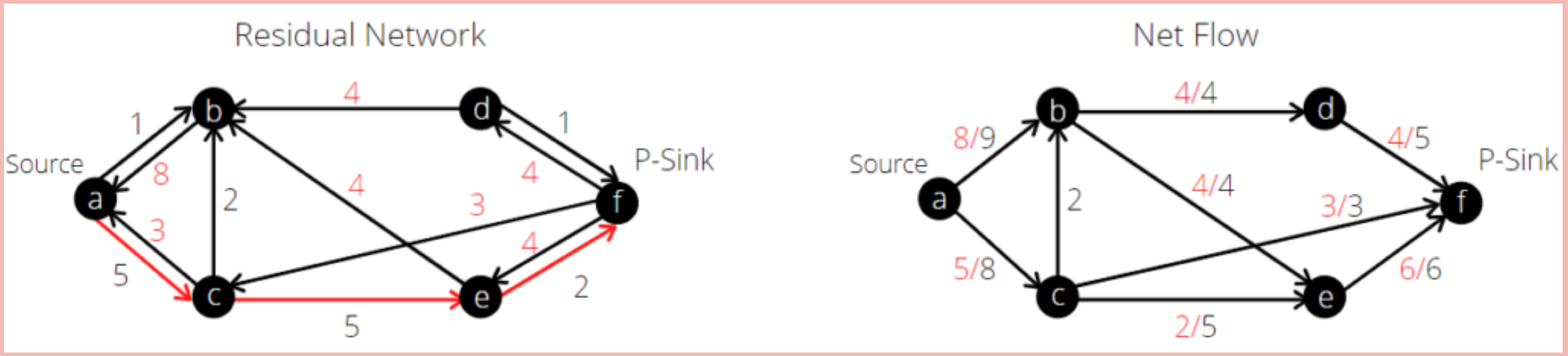
Net Flow



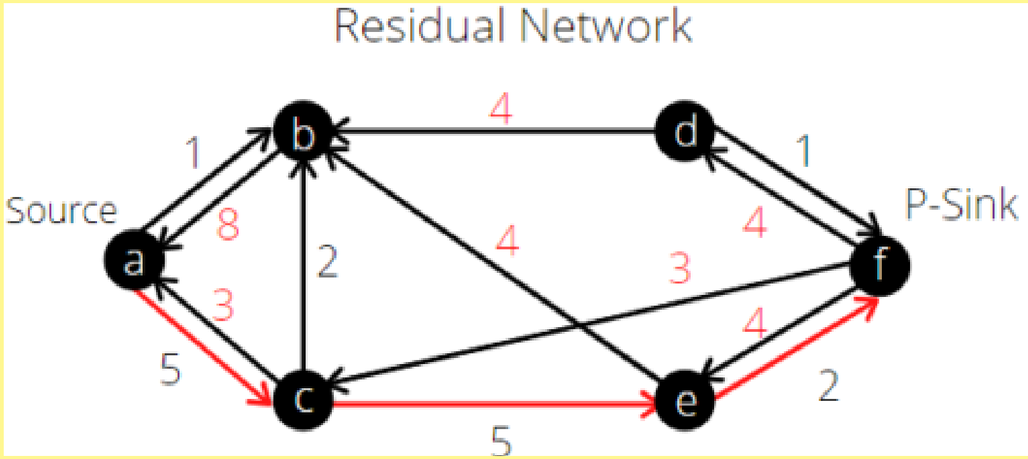
We'll continue the algorithm and select the next augmented path a-b-e-f:

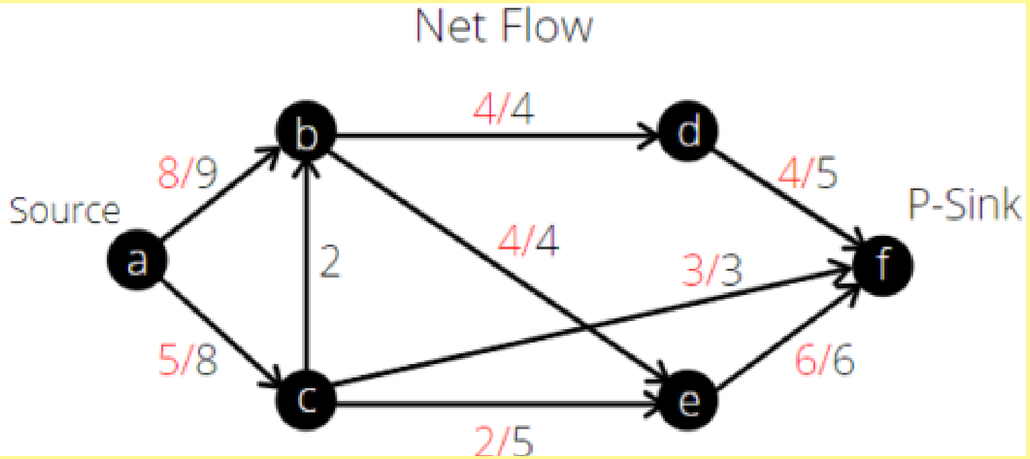


Our next pick is a-c-f:

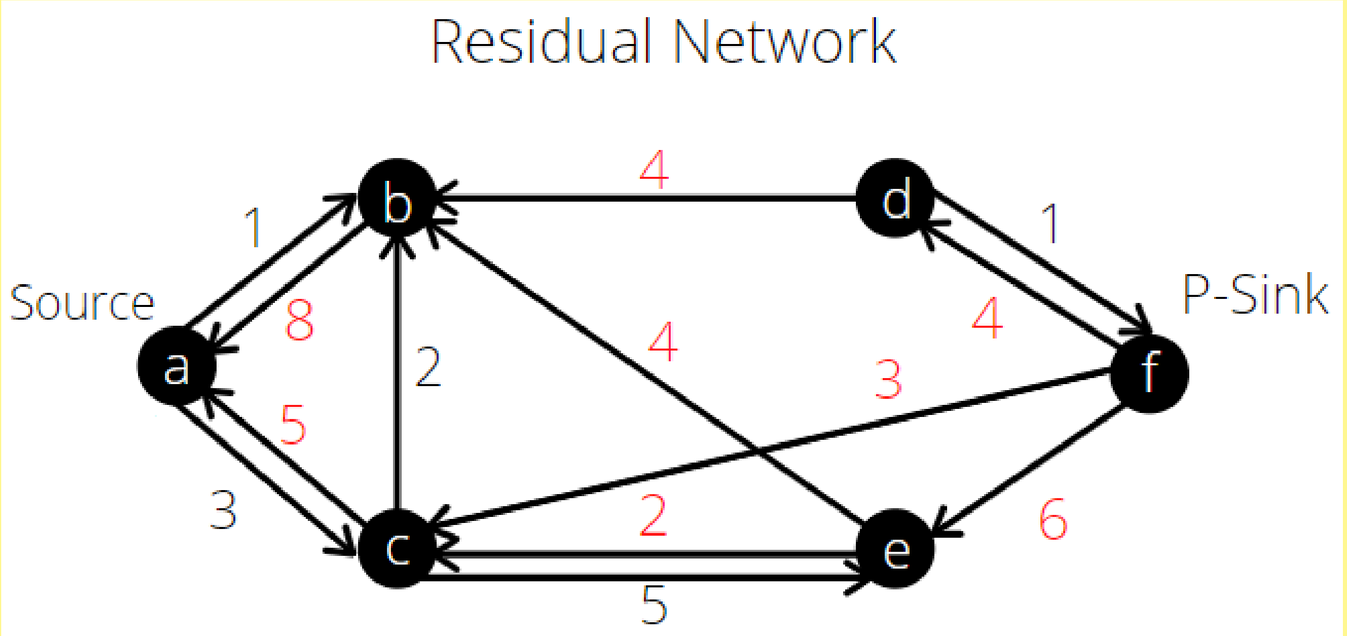


Still, we can pick more augmented paths. We pick the path a-c-e-f:





Now let's observe the residual graph, and let's see if we can find more augmented paths:



If we see the current residual graph from the source node a, we can't reach the sink node f via a path. Therefore, we terminate the algorithm as we can't find any more augmented paths. Now according to the algorithm, the maximum outgoing flow from the sink node should be equal to the maximum incoming flow of the source node. Let's verify this.

The maximum outgoing flow from the node f is 13 and also the maximum incoming flow for the source node f is 13, Hence, we verified that the Ford-Fulkerson algorithm works fine and provides the maximum flow of a graph correctly.
Now according to the maximum-flow minimum-cut theorem, the minimum cut of this graph would be equal to the maximum flow of the graph.
Therefore, the minimum cut of this example graph is 13.

Time Complexity Analysis

The Ford-Fulkerson algorithm depends heavily on the method used to find an augmented path. An augmented path can be found using a Breadth-first search (BFS) or Depth-first search (DFS). If we choose an augmented path using BFS or DFS, the algorithm runs in polynomial time.

The execution time of BFS is equal to O(E'), where E' is the number of edges in the residual graph G_f. For each edge, the flow will be increased, and in the worst case, it reaches its maximum flow value f*. Therefore the algorithm will be iterated at most f* times. Hence, the overall time complexity of the Ford-Fulkerson algorithm would be O(f* × E').

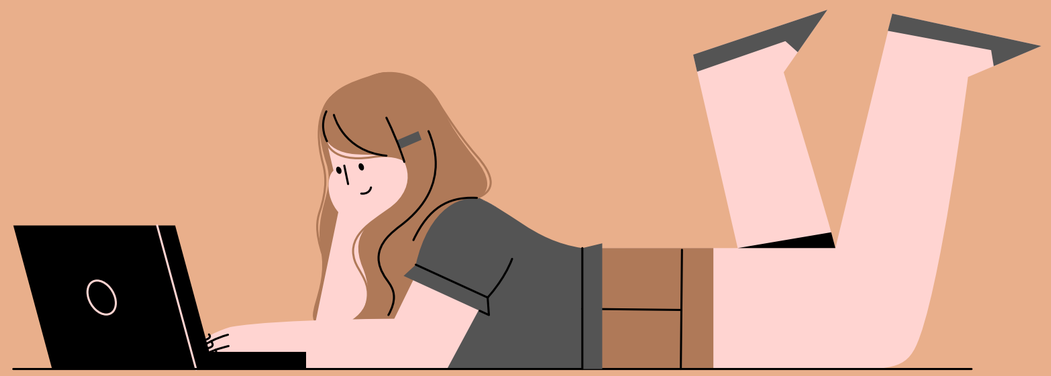
Results

Max Flow & Min Cut Result

General Idea

In computer science and optimization theory, the max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source to the sink is equal to the total weight of the edges in a minimum cut, i.e. the smallest total weight of the edges which if removed would disconnect the source from the sink.

The algorithm starts with a workable flow through the graph, and the flow is improved iteratively.
If this flow is maximum, it makes it possible to determine the flow function satisfying this value as well as the minimum cut. If the flow is not maximum, its objective is to highlight an improving path corresponding to this flow.
Initially, the algorithm starts by setting the flow value between the source and sink node to . **At each iteration, we find an augmented path and increase the flow value.** We'll terminate the algorithm and return the flow value when no more augmented paths can be found.



In this presentation, we discussed how to find a minimum cut by calculating the maximum flow value of a graph. We presented the Ford-Fulkerson algorithm to solve the maximum flow problem in a graph.

To find the minimum cut of a graph, we discuss the max-flow min-cut theorem. Finally, we verified the Ford-Fulkerson algorithm with an example and analyzed the time complexity.

An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision

Interactive Object Segmentation

In this section we describe experimental tests that compare min-cut/max-flow algorithms on Interactive Graph Cuts segmentation technique in . The method in allows for the segmentation of an object of interest in N-D images/volumes. This technique generalizes the MAP-MRF method of Greig at. al. by incorporating additional hard constraints into the minimization of the Potts energy

E(L) = X

p∈P

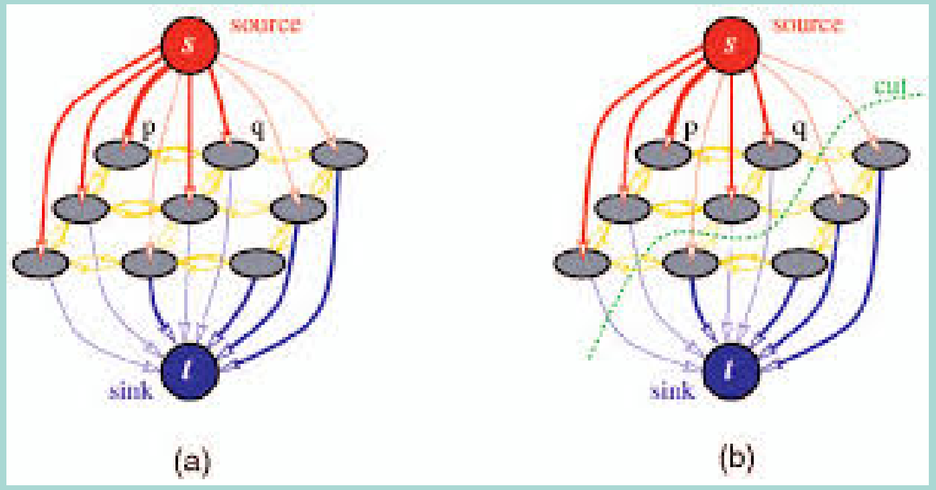
Dp(Lp) + X

(p,q)∈N

K(p,q)

· T (Lp 6= Lq)

over binary (object/background) labelings of image. The hard constrains come from a user placing some object and background seeds



Conclusion

The algorithm begins with a working flow through a chart, and the flow is improved iteratively.

If this stream is maximum, it allows you to determine the flow function that satisfies this value, as well as the minimum cut. If the stream is not maximum, its goal is to highlight the improvement path that corresponds to this stream.

Initially, the algorithm begins with setting the flow value between the source and recipient node in \ mathsf {0}. On each iteration, we find an enlarged path and increase the value of the stream. We will stop the algorithm and refund the value of the flow when the enlarged paths are no longer found.

The analysis of Ford-Fulkerson depends heavily on how the augmenting paths are found. The typicIn this guide, we discussed how to find a minimum edge, having calculated the maximum flow rate value. We presented the Ford-Fulkerson algorithm to solve the maximum flow problem in the graph.

GENERAL

To find the minimum incision of the graph, we will discuss the Max-Flow Min theorem. Finally, we checked the Ford-Fulkerson algorithm with an example and analyzed temporary complexity. al method is to use breadth-first search to find the path. If this method is used, Ford-Fulkerson runs in polynomial time.

TO FIND THE MINIMUM CUT OF THE GRAPH

The task of finding the maximum stream of minimum cost is to find the stream of the greatest value, and among all such - with minimal cost. In the particular case, when the weights of all the edges are the same, this task becomes an equivalent conventional task of the maximum stream.

MIN COST