FORD-FULKERSON ALGORITHM

2005095 - Tamim Hasan Saad
2005096 - Habiba Rafique
2005119 - Sadia afrin Sithi

Department of CSE
Bangladesh University Of Engineering and Technology

February 19, 2024

Table of Contents

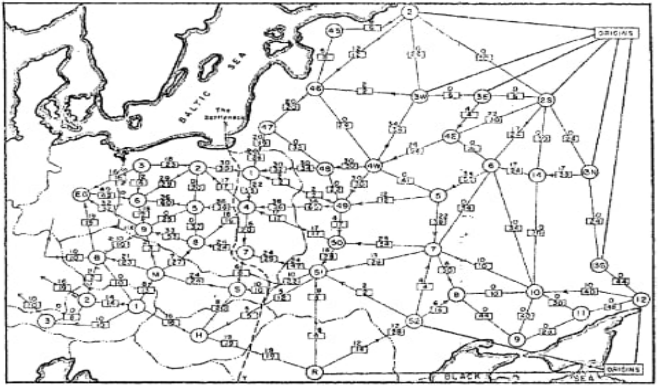
Scenarios

Some real life scenarios!!

$\rightarrow$ Network Traffic Optimization
$\rightarrow$ Water Distribution Systems
$\rightarrow$ Electric Power Grids
$\rightarrow$ Network reliability
$\rightarrow$ Airline scheduling
$\rightarrow$ Baseball elimination
$\rightarrow$ Distributed computing
and many many more . . .

An Example

Soviet Rail Network, 1955



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

Introduction

Network FLow

A Network is a directed graph **G**
Edges represent pipes that carry flow
Each edge $<$ **u,v** $>$ has a maximum capacity $c<$ **u,v** $>$
A source node **s** in which flow arrives
A sink node **t** out which flow leaves

Value of a Flow

$|f| = \sum_{v \in V}(s, v) = \sum_{v \in V}(v, t)$
This is the total flow leaving s = the total flow arriving in t.

Introduction

Definition

The Ford-Fulkerson algorithm efficiently solves the maximum flow problem in a network by determining the maximum amount of flow from a source to a sink, respecting edge capacity constraints.

History

Inventors

This algorithm was discovered in 1956 by Ford and Fulkerson.

L.R.Ford,Jr.

Delbert Ray Fulkerson

Scenarios

Introduction

Problem Description

Properties
- Residual Graph
- Augmenting Path
- Ford-Fulkerson

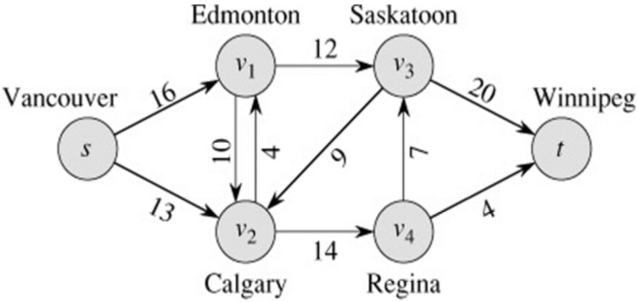
Example

Min-Cut

Running Time Complexity

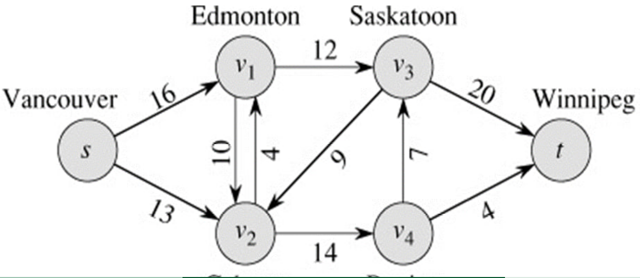
The Problem

Use a graph to model material that flows through conduits. Each edge represents one conduit, and has a capacity, which is an upper bound on the flow rate = units/time. Can think of edges as pipes of different sizes. Want to compute max rate that we can ship material from a designated source to a designated sink.



The Problem

- Each edge (u,v) has a nonnegative capacity c(u,v).
- If (u,v) is not in E, assume c(u,v)=0.
- We have a source s, and a sink t.
- Assume that every vertex v in V is on some path from s to t.
- c(s,v1)=16; c(v1,s)=0; c(v2,v3)=0



Scenarios

Introduction

Problem Description

Properties

- Residual Graph
- Augmenting Path
- Ford-Fulkerson

Example

Min-Cut

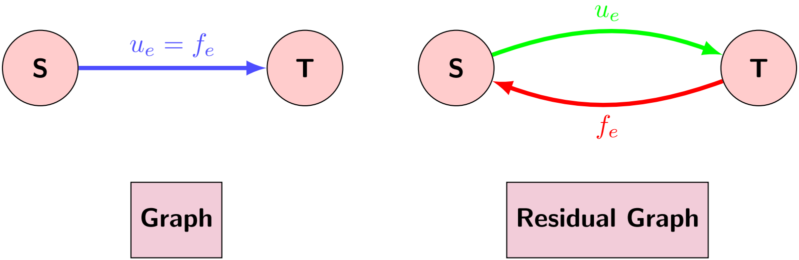
Running Time Complexity

Residual Network

Residual Graph

$\rightarrow$ Original edge: e = (u, v) $\in E$.
Flow f(e), capacity c(e).
$\rightarrow$ Create two residual edges
Forward edge
e = (u, v) with capacity c(e) - f(e)
Backward/reverse edge
e' = (v, u) with capacity f(e)
$\rightarrow$ Residual graph: G_f = (V, E_f)
where,
E_f = edges with positive residual capacity
$E_f = \{e : f(e) < c(e)\} \cup \{e' : f(e) > 0\}$

Residual Network



Why do we need residual networks?

- Allow us to reverse flow if necessary
- If we take a bad path then it will be helpful
- The bad path will overlap with too many paths.....

Augmenting Path

Definition

An augmenting path P is a simple path from s to t on the residual network.

Idea

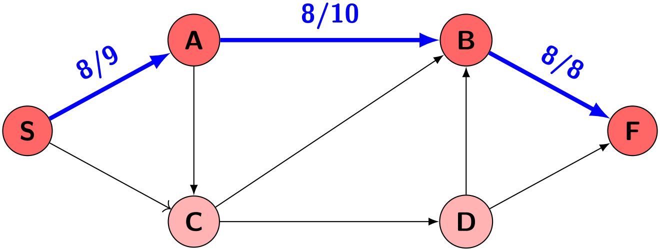
Increase flow on forward edges
Decrease flow on backward edges

Bottleneck capacity

let bottleneck(P, f) be the minimum residual capacity (i.e., capacity in G_f) of any edge in P

$$C_f(P) = \min\{(u, v) : (u, v) \text{ is on } P\}$$

Augmenting Path



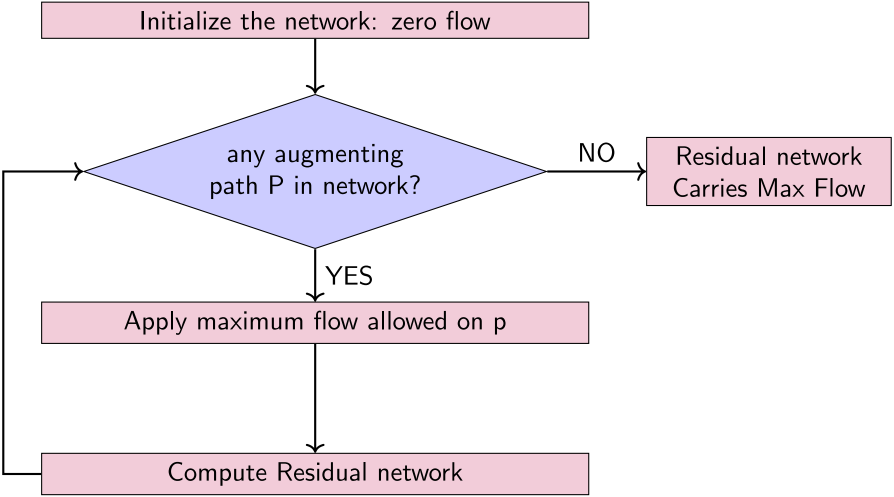
Augmenting path which is a simple path from the source to the sink.
Here is an augmenting path:
S → A → B → F
Bottleneck capacity is the minimum capacity of augmenting path.Here Bottleneck capacity is 8 which is the minimum of the three edges in augmenting path

Augmenting Path

Use path P in G_f to to update flow f

Augment$(f, P)\{$ // edge on P with least residual capacity
  $b =$ bottleneck(P, f)
  foreach $e = (u, v) \in P\{$
    if e is a forward edge
      $f(e) = f(e) + b$ //forward edge: increase flow
    else
      let $e' = (v, u)$
      $f(e') = f(e') - b$ //backward edge: decrease flow
  $\}$
  return f
$\}$

Ford-Fulkerson Method

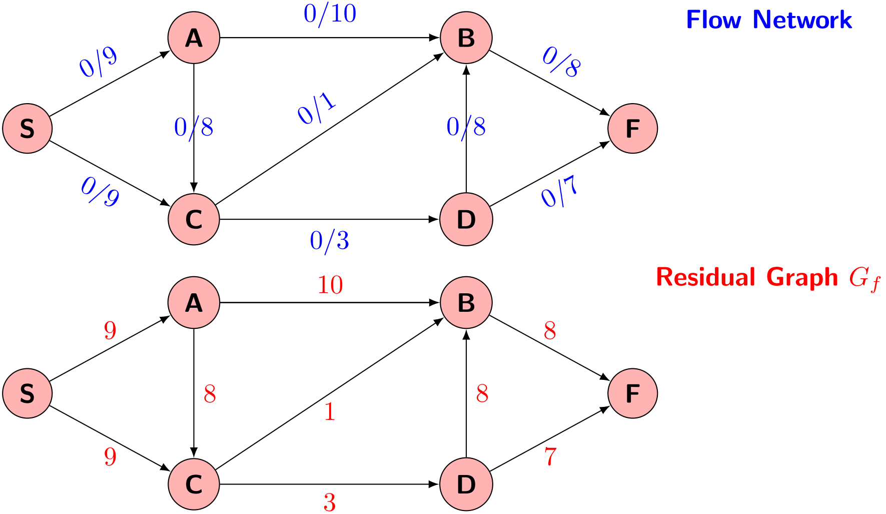


Ford-Fulkerson Algorithm

Repeat: find an augmenting path, and augment!

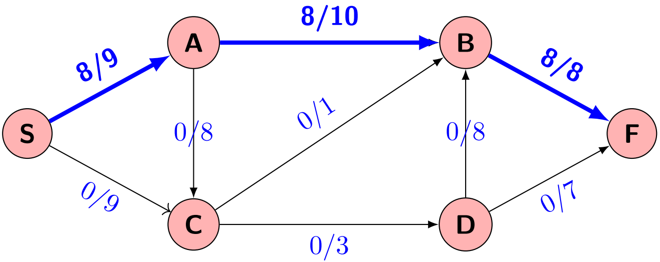
```
Ford-Fulkerson(G, s, t){
   foreach e ∈ E   f(e) = 0   //initially, no flow
   G_f = copy of G   //residual graph = original graph
   while (there exists an s-t path P in G_f){
      f = Augment(f, P)   //change the flow
      update G_f   //build a new residual graph
   }
   return f
}
```

An Example

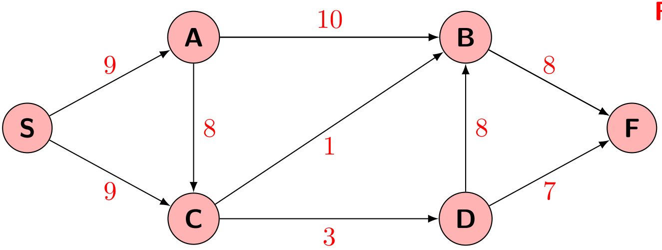


Flow Network
A
B
S
F
C
D
0/10
0/9
0/8
0/1
0/8
0/8
0/9
0/7
0/3
Residual Graph G_f
A
B
S
F
C
D
10
9
8
8
1
8
9
7
3

An Example



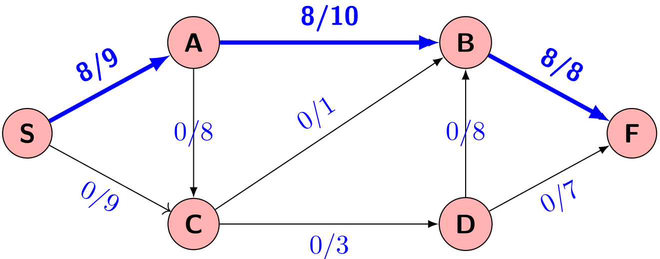
Flow Network



Residual Graph G_f

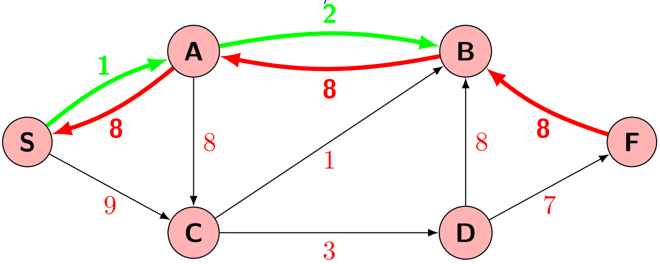
Start:
F(u,v) = 0 for all edges
$F_m = 0$

An Example



Flow Network

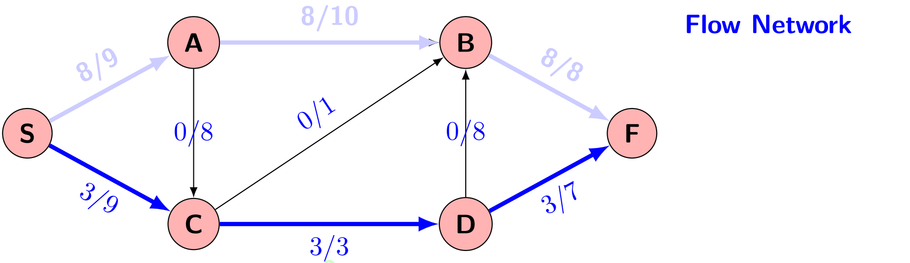
$F_m = 0 + 8 = 8$
Augmenting Path:
$S \rightarrow A \rightarrow B \rightarrow F$

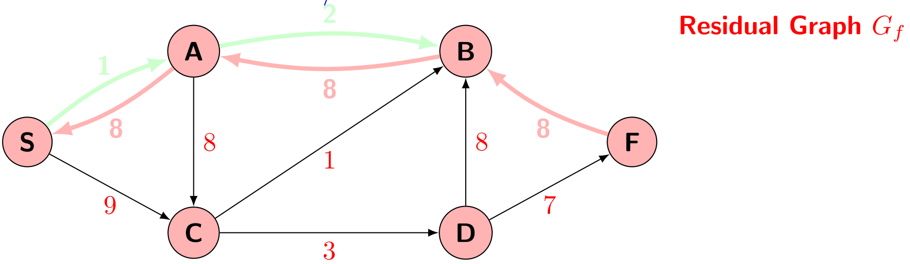


Residual Graph G_f

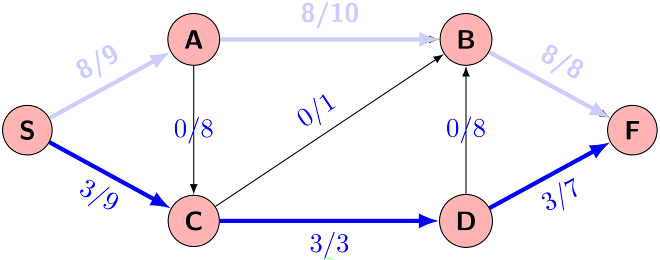
Residual Capacity : 8

An Example


Flow Network
S
A
B
C
D
F
8/9
8/10
8/8
0/8
0/1
0/8
3/9
3/3
3/7


Residual Graph G_f
S
A
B
C
D
F
1
8
2
8
8
8
1
8
9
3
7

An Example

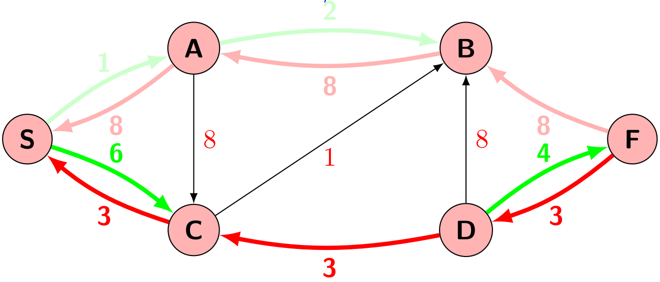


Flow Network

$F_m = 8 + 3 = 11$

Augmenting Path:

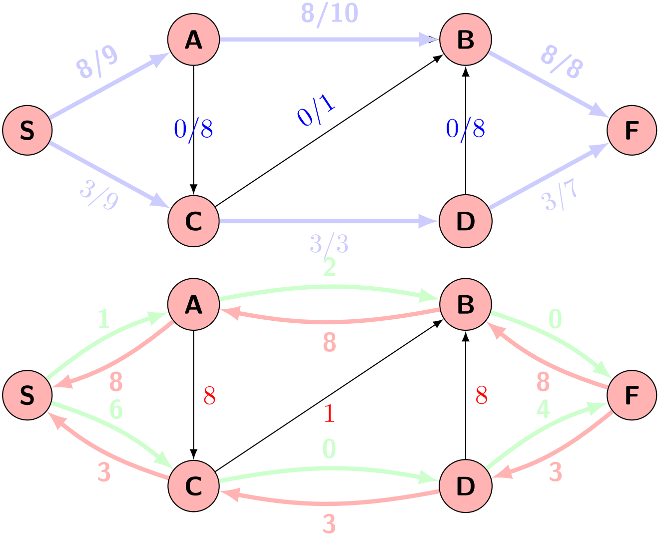
$S \to C \to D \to F$



Residual Graph G_f

Residual Capacity : 3

An Example



Flow Network

Residual Graph G_f

An Example

But is there any flow remaining?

The Answer is no!

An Example

But is there any flow remaining?

The Answer is no!

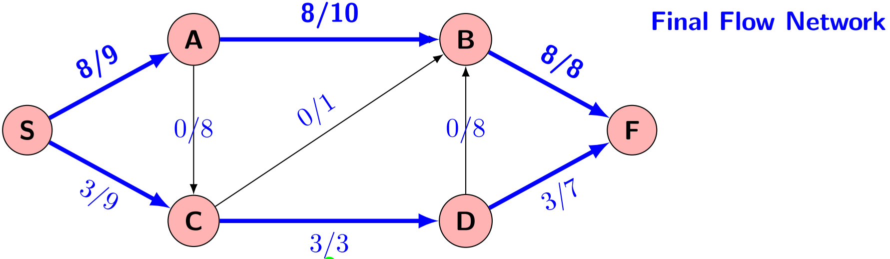
An Example

THINK AGAIN!

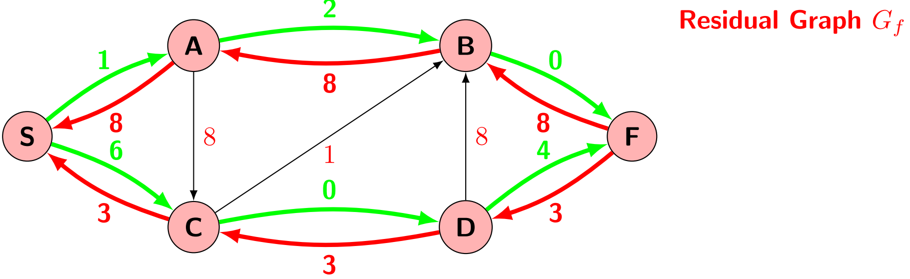
An Example

THE ANSWER IS STILL NO !!!

An Example



Final Flow Network
S
A
B
C
D
F
8/9
3/9
8/10
0/8
0/1
0/8
8/8
3/7
3/3



Residual Graph G_f
S
A
B
C
D
F
1
8
6
3
2
8
8
1
0
3
8
0
8
4
3

What is Cut?

Cuts

An s-t cut is a partition (A, B) of V with $s \in$ A and t $\in B$.

The capacity of a cut (A, B) is c(A,B) $= \sum_{\text{e out of A}}$ c(e)

Let f be any flow, and let (A, B) be any s-t cut.
Then $\sum_{\text{e out of A}}$ f(e) - $\sum_{\text{e into A}}$ f(e) $=$ v(f).

Max Flow - Min Cut

Min-Cut

Min-cut is the one with the minimum total capacity. It has the smallest sum of capacities among all cuts that separate the source and sink nodes.
The value of the minimum cut is equal to the maximum flow in the network, which is a fundamental property exploited by the Ford-Fulkerson algorithm for finding the maximum flow.

Maxflow-Mincut Theorem

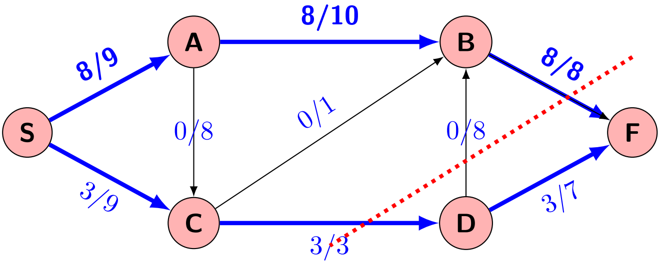
Conditions

If f is a flow in a flow network G=(V,E), with source s and sink t, then the following conditions are equivalent:

- $\Rightarrow$ f is a maximum flow in G.
- $\Rightarrow$ The residual network G_f contains no augmented paths.
- $\Rightarrow$ $|f| = c(S, T)$ for some cut (S,T) (a min-cut).

It is a flow since there is no augmented paths It is maximum since the sink is not reachable from the source

Min-Cut Example



mincut value = 11

See!!

Mincut Value v(f) $\Leftrightarrow$ Maxflow $|M_f| = 11$

Running Time?

The running time depends on

- ✓ Number of augmenting paths needed to find a maxflow
- ✓ Time needed to find each augmenting path

Running Time

Find a residual path $\rightarrow$ O(m+n)
Compute bottleneck capacity $\rightarrow$ O(m)
Update flow $\rightarrow$ O(m)
Update residual graph $\rightarrow$ O(m)
Total running time $\rightarrow$ O(C(m+n))

Running Time?

Analysis

$$\begin{cases} \textbf{for} \text{ each edge (u,v) } \in E[G] \\ \qquad \textbf{do } \text{f[u,v]} \leftarrow 0 \\ \qquad \text{f[u,v]} \leftarrow 0 \end{cases}$$

while there exits a path p from s to t in the residual network G_f

$$\begin{cases} \textbf{do } c_f(\text{p}) \leftarrow min\ c_f\{\text{(u,v):(u,v)is in p}\} \\ \quad \textbf{for} \text{ each edge (u,v) in p} \\ \qquad \textbf{do } \text{f[u,v]} \leftarrow \text{f[u,v]} + c_f(\text{p}) \\ \qquad \text{f[u,v]} \leftarrow \text{-f[u,v]} \end{cases}$$

O(E) for both cases 'indicated by {'

Running Time?

Analysis

- If capacities are all integers, then each augmenting path raises $|f|$ by ≥ 1.
- If maxflow is $f^\star$, then need $\leq |f^\star|$ iterations.
- So the time complexity is $O(E|f^\star|)$.
- This running time is not polynomial in input size. It depends on $f^\star$, which is not a function of $|V|$ or $|E|$.
- If capacities are rational, you can scale them to integers.
- If capacities are irrational, the Ford-Fulkerson algorithm might never terminate!


THANK
YOU


Any
questions