WALKTHROUGH OF FORD-FULKERSON

A flow network

The residual graph

An augmenting path

\[ \delta = 4 \]
WALKTHROUGH OF FORD-FULKERSON
WALKTHROUGH OF FORD-FULKERSON

We cannot find an augmenting path in the residual graph. So, terminate.
```python
def ford_fulkerson(g, s, t):
    # Let f be a flow, initially empty
    for u → v in g.edges:
        f(u → v) = 0

    # Define a helper function for finding an augmenting path
    def find_augmenting_path():
        # Define the residual graph h on the same vertices as g
        for u → v in g.edges:
            if f(u → v) < c(u → v):
                give h an edge u → v labelled “inc u → v”
            if f(u → v) > 0:
                give h an edge v → u labelled “dec u → v”
        if h has a path from s to t:
            return some such path, together with the labels of its edges
        else:
            # Let S be the set of vertices reachable from s (used in the proof)
            return None

    # Repeatedly find an augmenting path and add flow to it
    while True:
        p = find_augmenting_path()
        if p is None:
            break
        else:
            compute δ, the amount of flow to apply along p, and apply it
            # Assert: δ > 0
            # Assert: f is still a valid flow
```

The diagram shows a network of nodes and edges, with arrows indicating the direction of flow. The vertices are labeled as follows: s, a, b, c, and t. The text describes a function that implements Ford-Fulkerson algorithm to find the maximum flow in the network.
SECTION 6.3

Max-flow min-cut
ORIGINS

The Bottleneck

EG

Fig. 7 — Traffic pattern: entire network available

Legend:
- = international boundary
- = railway operating division
- = capacity [12 each way per day]
- = required flow of 9 per day toward destinations (in direction of arrow) with equivalent number of returning trains in opposite direction

All capacities in [1000's of tons] each way per day

Origins: Divisions 2, 3, 5, 7, 9, 13, 14, 15, 16, 52 (USN), and Roumania

Destinations: Divisions 3, 5, 9 (Poland); B (Czechoslovakia); and 2, 5 (Austria)

Alternative destinations: Germany or East Germany

Note: EK at Division 9, Poland
The Bottleneck

Total capacity 228 trains/day

Total capacity 163 trains/day

Total capacity 276 trains/day
A cut is a partition of the vertices into two sets, \( V = S \cup \tilde{S} \), with the source vertex \( s \in S \) and the sink vertex \( t \in \tilde{S} \).

The capacity of the cut is

\[
\text{capacity}(S, \tilde{S}) = \sum_{u \in S, \; v \in \tilde{S}: \; u \rightarrow v} c(u \rightarrow v)
\]

**MAX-FLOW MIN-CUT THEOREM**

For any flow \( f \) and any cut \((S, \tilde{S})\),

\[
\text{value}(f) \leq \text{capacity}(S, \tilde{S})
\]
MAX-FLOW MIN-CUT THEOREM
For any flow $f$ and any cut $(S, \bar{S})$, 
\[ \text{value}(f) \leq \text{capacity}(S, \bar{S}) \]
MAX-FLOW MIN-CUT THEOREM

For any flow $f$ and any cut $(S, \bar{S})$,

\[
\text{value}(f) \leq \text{capacity}(S, \bar{S})
\]
CORRECTNESS THEOREM

Suppose Ford-Fulkerson terminates, producing a flow $f^*$. Then $f^*$ is a maximum flow.

1. Let $S^* = \{\text{vertices reachable from } s\}$ in the residual graph, at termination.
2. The algorithm terminated, so $t \notin S^*$, so $(S^*, \bar{S}^*)$ is a cut.
3. The residual graph has no edges from $S^*$ to $\bar{S}^*$, hence
   - on edges $S^* \rightarrow \bar{S}^*$ in the flow network, flow=capacity
   - on edges $S^* \leftarrow \bar{S}^*$ in the flow network, flow=0
4. From the inequalities in the max-flow min-cut theorem, $\text{value}(f^*) = \text{capacity}(S^*, \bar{S}^*)$; hence $f^*$ is a maximum flow.
SECTION 6.4

Matchings
DEFINITIONS

▪ A bipartite graph is an undirected graph in which the vertices are split into two sets, and all edges go between these sets

▪ A matching in a bipartite graph is a selection of edges, such that no vertex is connected to more than one of the edges

▪ The size of a matching is the number of edges it includes

▪ A maximum matching is one with the largest possible size

PROBLEM STATEMENT

Given a bipartite graph, find a maximum matching
0. Given a bipartite graph...

1. Build a helper graph:
   • add source $s$ and sink $t$
   • add edges from $s$ and to $t$

2. Solve max-flow on the helper graph, to find a maximum flow $f^*$

3. Interpret the flow $f^*$ as a matching

What's the bug in my thinking?
0. Given a bipartite graph...

1. Build a helper graph:
   - add source $s$ and sink $t$
   - add edges from $s$ and to $t$

2. Solve max-flow on the helper graph, to find a maximum flow $f^*$

3. Interpret the flow $f^*$ as a matching

wtf?! This isn’t the sort of flow I expected!
0. Given a bipartite graph...

1. Build a helper graph:
   • add source $s$ and sink $t$
   • add edges from $s$ and to $t$

2. Solve max-flow on the helper graph, to find a maximum flow $f^*$

3. Interpret the flow $f^*$ as a matching

I’ll set up a flow problem where the goal is to pick edges to discard.

Hold on! The max-flow solution actually leads to a worse matching.
THE TRANSLATION STRATEGY

**CLAIM1:** We can find a max flow $f^*$ that can be translated into a matching, call it $m^*$

**CLAIM2:** If there were a larger-size matching $m'$ then it would translate to a larger-value flow $f'$. But there cannot be such a $f'$, because $f^*$ is a maximum flow. Therefore there is no such $m'$, thus $m^*$ is a maximum matching.
THE TRANSLATION STRATEGY

**CLAIM1:** We can find a max flow $f^*$ that can be translated into a matching, call it $m^*$

**CLAIM2:** If there were a larger-size matching $m'$ then it would translate to a larger-value flow $f'$

But there cannot be such a $f'$, because $f^*$ is a maximum flow. Therefore there is no such $m'$, thus $m^*$ is a maximum matching.

*Ford-Fulkerson will produce an integer flow, since all capacities are integer. Indeed, the flow on each edge must be either 0 or 1.*

*The capacity constraints tell us that, when we translate $f^*$ into an edge selection, it meets the definition of “matching”.*

*When we did the translation $f^* \rightarrow m^*$, value($f^*$) = size($m^*$)*

*When we translate any matching to a flow, in the obvious way, value(flow)=size(matching)*

*So if we had a larger-size matching $m'$ it would translate to a larger-value flow $f'$.*
Q. A signal failure can prevent travel in both directions between a pair of stations. How many signal failures it would take to prevent travel from Kings Cross to Embankment?