Graphs III - Network Flow
Flow network setup

- graph $G=(V,E)$
- edge capacity $w(u,v) \geq 0$
  - if edge does not exist, then $w(u,v)=0$
- special vertices: source vertex $s$; sink vertex $t$
  - no edges into $s$ and no edges out of $t$
- Assume every vertex $v$ is on a path from source to sink ($s \rightarrow v \rightarrow t$)
  - vertices $v$ that are not on such path can be ignored (deleted) along with all connecting edges
Flow network

- flow is a function $f: V \times V \rightarrow \mathbb{R}$ such that
  - flow from $u$ to $v$: $f(u,v) \leq w(u,v)$
  - symmetry $f(u,v) = -f(v,u)$
  - flow is conserved on all nodes except source $s$ and sink $t$

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

- total flow (from the source)

$$|f| = \sum_{v \in V} f(s, v)$$
Maximum Flow Problem

- determine the flow $f$ that realizes the maximum total flow
More on Flows

- $f(u,u)=0$
- total net flow into/out-to a vertex is 0

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

- except for source $s$ and sink $t$

- if edge $(u,v)$ is missing in $G$, there can be no net flow from $u$ to $v$
More on Flows

• positive net flow entering \( v \)
  \[
  \sum_{u \in V; f(u,v) > 0} f(u, v)
  \]

• positive net flow leaving \( v \)
  \[
  \sum_{u \in V; f(v,u) > 0} f(v, u)
  \]

• these two are equal
Cancellation

- positive flow on \((u,v)\) cancels positive flow on \((v,u)\) until only one is positive (the other becomes 0)
  - both flows decrease, so they still satisfy capacity constraints
  - flow conservation satisfied since both flow reduced by the same amount
Ford-Fulkerson

- want the max flow for source s to sink t
  - a class of algorithms, not a single one

- initialize flow with 0;
- repeat
  - find an augmenting path from s to t (that admits more flow)
  - send more flow on that path
- until no augmenting path exists

- have to prove that this termination condition implies the flow is max.
  - if an augmenting path exists, sending more flow to it increases the value of the existing flow
Residual network

- after sending some flow on a path from $s$ to $t$, the graph essentially changes
  - existing flow edges will have a different capacity in the residual network (because the flow uses some)
  - new edges can appear (in red): the possibilities of reversing the existing flow
Residual network

• residual capacity of edge $(u,v)$: $r(u,v) = w(u,v) - f(u,v)$

• residual network “$R$” induced by $f$ is given by the set of edges also called “$R$” with positive residual capacity
  - edges set $R = \{ (u,v) : r(u,v) > 0 \}$

• note that some new edges appear!
  - example $(u,v) \in E$; $w(u,v)=3$, $f(u,v)=1$
  - then $r(u,v) = 3-1 = 2$
  - edge $(v,u)$ not in the original graph
  - but $r(v,u) = 0 - f(v,u) = 0 - (-1)=1$; therefore edge $(v,u)$ is now part of the residual network.

• edge $(v,u)$ can be part of the residual network only if either $(u,v)$ or $(v,u)$ are edges in the original graph
  - thus $|R| \leq 2|E|$
Augmenting paths

- any path $p = s \rightarrow t$ in the residual network $R$
- the residual capacity of the path $p$ is the minimum ("bottleneck") edge residual capacity
  - $r(p) = \min \{ r(u,v) : (u,v) \in p \}$
- add the path $p$ as additional flow $f_p$ of size $r(p)$
  - to the existing flow $f$ that created $R$
  - new flow $f' = f + f_p$
  - increases the flow total by $r(p)$. Proof in the book.
Cuts in flow network

- Cut $C = (S,T)$ is a partition of vertices:
  - $S \cup T = V$ ; $S \cap T = \emptyset$ ; $S = V \setminus T$
  - $S$ contains the source and $T$ contains the sink $s \in S$; $t \in T$

- Net flow across cut is $f(S,T)$, the sum of all flows on edges from $S$ to $T$

$$f(S, T) = \sum_{u \in S; v \in T} f(u, v)$$

- Capacity of a cut is the sum of edges capacity from $S$ to $T$

$$w(S, T) = \sum_{u \in S; v \in T} w(u, v)$$
Max Flow – Min Cut theorem

- (S,T) is a cut, f a flow with total value |f|. Then
  - \( f(S,T) = |f| \) (the total flow value)
  - consequently \( |f| \leq w(S,T) \) : flow value is smaller than the capacity of any cut

- MAX-FLOW MIN-CUT theorem. The following are equivalent:
  - (a) f is a max flow
  - (b) residual network \( R=R_f \) has no augmenting paths
  - (c) there is a cut (S,T) such that \( |f| = w(S,T) \)
Max Flow – Min Cut proof intuition

• (a)=>(b) already discussed

• (b)=>(c): consider $S = \{ v \mid \exists$ path $s \rightarrow v$ in residual $R\}$
  - $s \in S$
  - $T = V \setminus S; t \in T$. If $t \in S$, then there would be an augmenting path in $R$
  - $R$ can't have an edge $(v \in S, u \in T)$ because that would mean $u \in S$
  - thus existing flow saturates the cut $(S, T)$

• (c)=>(a): no flow can be bigger than capacity of a cut, so $f$ must be a maximum flow (since it saturates the cut described above)
Ford-Fulkerson

for each edge \((u,v)\)
- init: \(f(u,v)=0; f(v,u)=0\)

\(R = G\)

while exists path \(p(s\rightarrow t)\) in residual \(R\)
- \(c(p) = \min \{ r(u,v); (u,v) \in p \} \) //path capacity, used as new flow

for each \((u,v)\in p\)
- \(f(u,v) = f(u,v) + c(p) ; f(v,u)= - f(u,v)\)

recompute residual network \(R=R_f\)
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$ = total flow value
  - At most $|f|$ iterations; every iteration increases the flow by 1 or more
  - Total $O(E|f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - For real-value edge capacities, Ford-Fulkerson can be arbitrarily slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in R is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E \cdot |f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

• Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E|f|)$

• problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E\times|f|)$

- problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$ = total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E \cdot |f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrarily slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$ = total flow value
  - At most $|f|$ iterations; every iteration increases the flow by 1 or more
  - Total $O(E \cdot |f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - For real-value edge capacities, Ford-Fulkerson can be arbitrarily slow
Ford-Fulkerson

- **Running time with integer capacities**
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E \times |f|)$

- **Problem:** $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$ = total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E|f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrarily slow
**Ford-Fulkerson**

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$ = total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E*|f|)$

- problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E*|f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E*|f|)$

- problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrary slow
Ford-Fulkerson

- Running time with integer capacities
  - Finding a path in $R$ is $O(E)$ (say with DFS)
  - $|f|$=total flow value
  - at most $|f|$ iterations; every iteration increases the flow by 1 or more
  - total $O(E|f|)$

- Problem: $|f|$ can be very large, thus the algorithm very slow
  - for real-value edge capacities, Ford-Fulkerson can be arbitrarily slow
Edmonds-Karp

- same as FF, but find the augmenting path with BFS

  - for each edge \((u,v)\)
    - init: \(f(u,v)=0; f(v,u)=0\)
  
  - \(R = G\)

  - while BFS finds path \(p(s \rightarrow t)\) in residual \(R\)
    - \(c(p) = \min \{ r(u,v); (u,v) \in p \}\) //path capacity, used as new flow
    - for each \((u,v)\) \(\in p\)
      - \(f(u,v) = f(u,v) + c(p) ; f(v,u)= - f(u,v)\)
    - recomputes residual network \(R=R_f\)
Analysis of Edmonds-Karp

- BFS will find the augmenting path with fewest number of edges
- note that previous toy bad example would find max flow after two iterations

![Graph Diagram]
Analysis of Edmonds-Karp

- BFS will find the augmenting path with **fewest number of edges**
- Note that previous toy bad example would find max flow after two iterations
Analysis of Edmonds-Karp

- BFS will find the augmenting path with **fewest** number of edges
- Note that previous toy bad example would find max flow after two iterations
Analysis of Edmonds-Karp

- BFS will find the augmenting path with fewest number of edges
- Note that previous toy bad example would find max flow after two iterations
Analysis of Edmonds–Karp

- BFS will find the augmenting path with fewest number of edges
- Note that previous toy bad example would find max flow after two iterations
Analysis of Edmonds-Karp

• How many augmenting paths can EK find?
  
  – augmenting path p has critical edge (u,v), if (u,v) is the minimum residual capacity edge on the path
  
  – any edge can be critical at most |V| times during EK. Proof in the book
  
  – there are E edges, so at most |V|*|E| critical edges for the entire execution
  
  – thus at most $O(VE)$ augmenting paths (each path has at least one critical edge)

• BFS takes $O(E)$ to find each augmenting path

• total $O(VE^2)$
Push-Relabel (Optional reading)

- Advanced material not covered
  - optional reading from book

- intuition: flood the network, using vertex heights
  - nodes can accumulate flow
  - the more flow they accumulate, the “higher” they go
  - flow goes downhill

- practical / fast implementation: $O(V^3)$ running time.