Last word on Max Flow
Outline for today

• Variants of the Ford-Fulkerson (FF)
• Analysis of a strongly polynomial time algorithm
• Another application: Project Selection
How to Choose Augmenting Paths?

FF(G) // “generic FF”
  // given a flow network G, return its max (s,t)-flow
initialize f(e)=0 for all e in G
WHILE there is a simple s-t path in in Gf
  let P be any such path
  f = augment(f, P)
ENDWHILE
return f

Exercise: What might be good ways to choose augmenting paths in the FF algorithm?
How to Choose Augmenting Paths?

Ideas:

• On each iteration of FF, choose the path in the residual graph with the *largest bottleneck*
  
  Running time: $O(m^2 \log m \log f^*)$, where $f^*$ is the value of the max flow

• On each iteration of FF, choose the *shortest* path in the residual graph
  
  Running time: $O(m^2 n)$
FF Variant: Augment Shortest Path

FF(G) // “shortest path FF”

// given a flow network G, return its max (s,t)-flow
initialize f(e)=0 for all e in G
WHILE there is a simple s-t path in in G_f
    let P be a shortest such path
    f = augment(f, P)
ENDWHILE
return f
Analysis of Shortest Path FF

Let’s start with an example
Analysis of Shortest Path FF

G

all horizontal edges have capacity 50
Let $G_i$ be the residual graph in the $i$th iteration of the algorithm.

Exercise: What is the shortest path $P_1$ in $G_1 (=G)$?
Exercise:
For $i=2,3$ what is the augmenting path $P_i$ in $G_i$?
What is the shortest path to $i$, HI, LO, and $t$ in $G_i$ for $i=1,2,3$?
Consider an execution of this algorithm (break ties arbitrarily) on a given input

Let $G_0 = G$ and for $i \geq 1$ let $G_i$ be the $i$th residual graph
Analysis of Shortest Path FF

Let $\text{level}_i(v)$ be the length of the shortest path from $s$ to $v$ in $G_i$, and infinity if there is no path.

Claim: For any $i > 0$, $\text{level}_i(v) \geq \text{level}_{i-1}(v)$, for all nodes $v$.

Proof: Induction on $\text{level}_i(v)$.
Analysis of Shortest Path FF

Let $\text{level}_i(v)$ be the length of the shortest path from $s$ to $u$ in $G_i$, and infinity if there is no path

Claim: For any $i > 0$, $\text{level}_i(v) \geq \text{level}_{i-1}(v)$, for all nodes $v$.
Proof: Induction on $\text{level}_i(v)$

Corollary: A edge can "appear" from the residual graph (i.e. be in $G_i$ but not in $G_{i-1}$) at most $n/2$ times
Analysis of Shortest Path FF

Let $\text{level}_i(v)$ be the length of the shortest path from $s$ to $u$ in $G_i$, and infinity if there is no path.

Claim: For any $i > 0$, $\text{level}_i(v) \geq \text{level}_{i-1}(v)$, for all nodes $v$.

Proof: Induction on $\text{level}_i(v)$

Corollary: A edge can "appear" from the residual graph (i.e. be in $G_i$ but not in $G_{i-1}$) at most $n/2$ times.

Claim: Shortest Path FF, takes at most $mn/2$ iterations, and thus has strongly polynomial running time of $O(m^2n)$.

(The proof uses the fact that at least one edge must “appear” in every iteration after the first.)
Recall reduction from Maximum Bipartite Matching to Max Flow

Exercise: Based on our bounds to date, what is our best running time bound for Maximum Bipartite Matching? (Our bounds: $O(mC)$, $O(m^2 \log m \log f^*)$, and $O(m^2 n)$ where $f^*$ is the value of the max flow and $C$ is the sum of the capacities from $s$)

From Jeff Erickson’s notes
Application: Project Selection
Application: Project Selection

Instance: Acyclic graph $G = (V = \{1,\ldots,n\}, E)$, where

- the nodes in $V$ represent projects
- profit $p_i$ is associated with project $i$ ($p_i$ may be negative)
- edge $(i,j)$ means $i$ depends on $j$: $i$ can't be started unless $j$ is done

A set $X$ of projects is valid if for every project in the set, the projects it depends on are also in $X$

Goal: Find a valid set of projects that maximizes the total profit
Application: Project Selection

Exercise: What is set of projects maximizes profit in this example?
Application: Project Selection

Exercise: Can you find an efficient algorithm for Project Selection?
Application: Project Selection

Min Cut (S,T)
\[ c(S,T) = 2 + 3 + 5 + 3 = 13 \]

Total capacity from S is
\[ C = 4 + 6 + 2 + 3 = 15 \]

Claim: \( C - c(S,T) \) is the max profit!
Summary: Max Flow

• The Ford-Fulkerson augmenting paths algorithm works well for small capacities; some of its variants solve the problem in strongly polynomial time.

• There are other quite different approaches to Max Flow, including the pre-flow push algorithm described in the Kleinberg-Tardos text.

• The best known algorithm has running time $O(mn)$ (Orlin, 2012, the previous best was $O(nm \log_{m/n} \log n)$, by Valerie King at UVic, with Rao and Tarjan).

• Many applications (see Kleinberg and Tardos).