Reading: 7.0-7.5

Last time:
  • Shortest-paths (Bellman-Ford Alg)

Today:
  • Reductions
  • Network flow
  • Bipartite matching
Reductions

“to solve problem $B$ given solution to problem $A$, transform instances from problem $B$ into instances of $A$, solve, transform solution back”

Problem $A$: Network Flow

“given a network with bandwidth constraints on links, how much data can we send from source to sink”

Def: a flow graph $G = (V, E)$ is a directed graph with:

- $c(e) =$ capacity of edge $e$
- $s \in V$ is source.
- $t \in V$ is sink.

Def: a flow $f$ in $G$ is an assignment of flow to edges “$f(e)$” satisfying:

- capacity: $\forall e, f(e) \leq c(e)$
- conservation: $\forall v \neq s, t, \sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

Def: the value of a flow is $|f| = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$

Problem: Network Flow

input: flow graph $G, s, t, c(\cdot)$.

output: flow $f$ with maximum value.

Example:

Max flow = 30.

Theorem 1: there is an algorithm to compute the max flow in polynomial time.

Theorem 2: if capacities are integral, then max flow is integral (on each edge).

Problem $B$: bipartite matching

Def: $G = (V, E)$ is a bipartite if exists partitioning of $V$ into $A$ and $B$ s.t.,

- $u, v \in A \Rightarrow (u, v) \notin E$,
- $u, v \in B \Rightarrow (u, v) \notin E$,

Recall: a matching is a set of edges $M \subseteq E$ each node is connected by at most one edge in $M$

- a perfect matching is one where all nodes are connected by exactly one edge.
- a maximum matching is one with maximum cardinality.

Problem: bipartite matching

input: bipartite graph $G = (A, B, E)$

output: a maximum matching $M$. 
Reducing bipartite matching to max flow

“use max flow alg to solve bipartite matching.”

Steps:
1. convert matching instance into flow instance.
2. run flow alg flow instance.
3. convert flow soln to matching soln.
4. prove flow soln optimal iff matching soln optimal.

Step 1:
(a) connect \( s \) to each \( v \in A \) with capacity 1.
(b) connect \( t \) to each \( u \in B \) with capacity 1.
(c) set capacity of each edge \( e \in E \) to 1.

Step 2: compute (integral) max flow \( f \)

Step 3: matching is \( M = \{e \in E : f(e) = 1\} \)

Step 4: Proof:
- any matching \( M' \) can be turned into a flow \( f' \) with \( |f'| = |M'| \)
  (send form \( s \) to each matched edge to \( t \) one unit of flow)
- any integral flow \( f' \) can be turned into a matching \( M' \) with \( |f'| = |M'| \)
  (capacity constraints imply matching)
\[ \Rightarrow \text{size of output matching} = \text{value of max flow} = \text{size of max matching}. \]

Runtime

\[ T_{\text{matching}}(n,m) = O(n + m) + T_{\text{max flow}}(n,m) \]
Network Flow

Example:

\[
\begin{array}{ccc}
& a & \\
s & 20 & 10 \\
& 30 & t \\
10 & & b \\
\end{array}
\]

Max flow = 30.

Idea: repeatedly push flow on s-t paths until can’t push anymore.

Example: Push 20 on \( P = (s, a, b, t) \)

\[
\begin{array}{ccc}
& a & \\
s & 20 & 10 \\
& 30 & t \\
10 & & b \\
\end{array}
\]

Note: when pushing flow, we can undo flow already pushed.

Def: the residual graph \( G_f \) for flow \( f \) on \( G \) is the graph that represents capacity constraints for flows after pushing \( f \)

Example: \( G_f \)

\[
\begin{array}{ccc}
& a & \\
s & 20 & 10 \\
& 30 & t \\
10 & & b \\
\end{array}
\]

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)

Def: the residual capacity of \( e \) in \( E_f \) is \( c_f(e) \).

Def: the bottleneck capacity of s-t path \( P \) in \( G_f \) is minimum residual capacity of any edge in \( P \).

Def: an augmenting path \( P \) in a residual graph \( G_f \) is a path with positive bottleneck capacity.

Example: \( G_f \) after pushing 20 on \( P = (s, a, b, t) \)

Augmenting path \( P = (s, b, a, t) \) with bottleneck capacity 10.

Augment \( f \) with flow of 10 on \( P \):

\[
\begin{align*}
& f(s, b) \leftarrow f(s, b) + 10 \\
& f(a, b) \leftarrow f(a, b) - 10 \\
& f(a, t) \leftarrow f(a, t) + 10
\end{align*}
\]

Note: can find augmenting paths with BFS.

Algorithm: Augment \( f \) with \( P \)
• for e in P:
  • if e a forward edge:
    \[ f(e) \leftarrow f(e) + b \]
  • if e a back edge:
    let \( e' = \) back edge
    \[ f(e') \leftarrow f(e) - b. \]

**Example:** \( G_f \) after augmenting with \( P = (s, b, a, t) \)

![Diagram](image)

No more augmenting paths!

**Algorithm:** Ford-Fulkerson

- \( f \leftarrow \) null flow.
- \( G_f \leftarrow G. \)
- while exists \( s-t \) path \( P \) in \( G \) (by BFS)
  - augment \( f \) with \( P \).
  - \( G_f \leftarrow \) residual graph for \( G \) and \( f \).
- return \( f \).

**Runtime**

Each iteration:
- construct \( G_f \): \( O(m) \).
- find \( P \): \( O(m) \).
- augmentation: \( O(n) \).
- (Total: \( O(m) \))

**Fact:** the value of flow increases by bottleneck capacity in each iteration.

**Theorem:** if \( C \) is upper bound on max flow and all capacities are integral then algorithm terminates in \( O(C) \) iterations with runtime \( O(nC) \)

**Proof:** (by “measure of progress”)

1. bottleneck capacities integral:
   - current residual capacities integral
   \[ \Rightarrow \] integral bottleneck capacity
   \[ \Rightarrow \] next residual capacities integral
   - induction!

2. bottleneck capacities \( \geq 1 \)

3. flow increases by 1 each iteration

4. terminates in \( \leq C \) iterations.

QED

**Note:** \( C \leq \sum_{e \text{ out of } s} c(e) \).

**Note:** Clever choice of augmenting paths gives runtime \( O(m^2 \log C) \).

**Correctness**

1. \( f \) is feasible.

2. \( f \) is optimal.

**Lemma:** \( f \) is feasible.

**Proof:** induction!