EECS 336: Lecture 8: Introduction to Algorithms

Network Flow: Ford-fulkerson, duality, minimum cut

Reading: 7.0-7.5

Announcements: midterm thursday

- closed book, closed notes.
- one handwritten cheat sheet.
- dynamic programming.
- focus:
  - writing Parts I-II.
  - writing Parts III-IV (given Parts I-II.)

Last Time:

- reduction
- Network flow defn
- Bipartite matching
- reduction: matching $\Rightarrow$ flow.

Today:

- Network flow
- duality: max flow = min cut

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

- $c(e)$ = **capacity** if 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:** $\forall v \neq s, t$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

Recall: the **value** of a flow is:

$$|f| = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$$

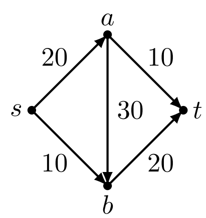
Recall: Max Network Flow Probem

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

output: flow f with maximum value.

Network Flow

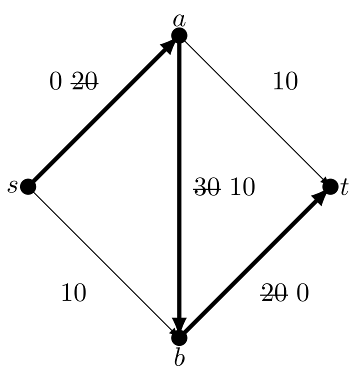
Example:



Max flow = 30.

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

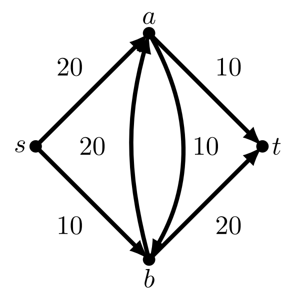
Example: Push 20 on $P = (s, a, b, t)$



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

Def: the <u>residual graph</u> G_f for flow f on G is the graph that represents capacity constraints for flows after pushing f.

Example: G_f



Construction: $G_f = (V, E_f), c_f(\cdot)$:

For each $e = (u, v) \in E$,

(if $f(e) = c(e)$ discard e)

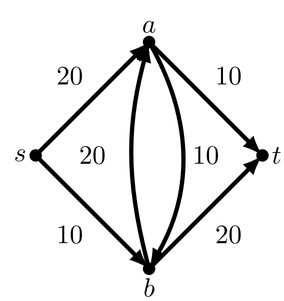
- if $f(e) < c(e)$,
  - add e to E_f
  - $c_f(e) = c(e) - f(e)$
- if $f(e) > 0$
  - let $e' = (v, u)$
  - add e' to E_f
  - $c_f(e') = c(e') + f(e)$

Def: the <u>residual capacity</u> of e in E_f is $c_f(e)$.

Def: the <u>bottleneck</u> capacity of s-t path P in G_f is minimum residual capacity of any edge in P.

Def: an <u>augmenting path</u> 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:

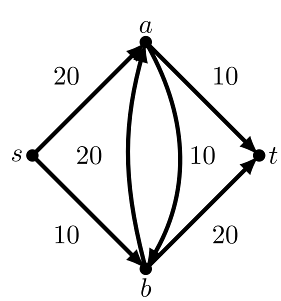
- $f(s, b) \leftarrow f(s, b) + 10$
- $f(a, b) \leftarrow f(a, b) - 10$
- $f(a, t) \leftarrow f(a, t) + 10$

Note: can find augmenting paths with BFS.

Algorithm: Augment f with P

- $b = \text{bottleneck}(P, G_f)$.
- 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 $P = (s, b, a, t)$



No more augmenting paths!

Algorithm: Ford-fulkerson

- $f \leftarrow$ null flow.
- $G_f \leftarrow G$.
- while exists s-t path P in G_f (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 terminate in $O(C)$ iterations with runtime $O(mC)$.

Proof: (by "measure of progress")

1. bottleneck capacities integral:
   - current residual capacities intergal
     - $\Rightarrow$ integral bottleneck capacity
     - $\Rightarrow$ next residual capacities integral
   - induction!
2. bottleneck capacities ≥ 1
3. flow increases by 1 each iteration
4. terminate in $\leq C$ iterations.

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!

Max flow = min cut

"duality: for maximization problem there is a corresponding minimization problem"

Recall: an s-t cut (A, B) is partion of V into A and B with $s \in A$ and $t \in B$.

Def: the capacity of cut (A, B) is

$$c(A, B) = \sum_{e \text{ from } A \text{ to } B} c(e)$$

Goal: flow algorithm is optimal

Proof Approach: primal = dual.

Claim 1: any flow f and any cut (A, B) then $\underbrace{|f|}_{\text{value of flow}} \leq c(A, B)$.

Claim 2: for flow f^* with no augmenting path in G_{f^*} then exists cut (A^*, B^*) with $|f^*| = c(A^*, B^*)$

Picture:

```
 *           c u t s          * *
   * * * *                 * *
         * * *   * * * * *
               *
      * * * * *   * *
   * *                 * * *
 *        f l o w s        * * *
```

Proof: (of theorem)

- all flows

$$|f| \underbrace{\leq}_{\text{by Claim 1}} c(A^*, B^*) \underbrace{=}_{\text{by Claim 2}} |f^*|$$

.

Corollary: value of max flow = capacity of min cut

Lemma: for any flow f, cut (A, B) then, $|f| = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$

Proof: (by picture, see text for formal proof)

Proof: (of Claim 1)

From Lemma:

$$\begin{aligned} |f| &= \textstyle\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \\ &\leq \textstyle\sum_{e \text{ out of } A} f(e) \\ &\leq \textstyle\sum_{e \text{ out of } A} c(e) \\ &= c(A, B). \end{aligned}$$

Proof: (of Claim 2) no s-t path in G_f:

- let A^* be vertices connected to s. $> (B^* = V \setminus A^*)$
- (A^*, B^*) is cut:
  - $s \in S^*$
  - $t \in B^*$
- for all $e = (u, v)$ out of A^* in G:
  - $e \notin G_f$

    $\Rightarrow f^*(e) = c(e)$
- for all $e = (u, v)$ in to A^* in G:
  - $e' = (v, u) \notin G_f$

    $\Rightarrow f^*(e) = 0$
- Lemma

$$\begin{aligned} \Rightarrow |f| &= \textstyle\sum_{e \text{ out of } A^*} f(e) - \sum_{e \text{ into } A^*} f(e) \\ &= \textstyle\sum_{e \text{ out of } A^*} c(e) - 0 \\ &= c(A^*, B^*). \end{aligned}$$

Summary

- algorithm: augmenting paths in residual graph.
- correctness: max-flow min-cut theorem.
- many problems can be reduced to network flows.
- entire courses on network flows.