Reading: 7.1,7.5

Last time:

- Interval Pricing

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

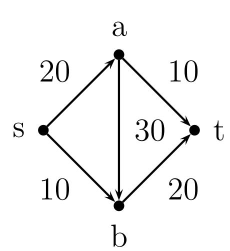
Def: the **value** of a flow is
$$|f| = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$$

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 with same value.
4. prove flow soln optimal iff matching soln optimal.
   (a) (convert flow soln to matching soln with same value)
   (b) convert matching soln to flow soln with same value.

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 $M = \{e \in E \;:\; f(e) = 1\}$

- $|M| = |f|$
- (capacity constraints imply matching)

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 $|f'| = |M'|$

  (Step 3)

$\Rightarrow$ size of output matching = value of max flow = size of max matching.

Runtime

$$T_{\text{matching}}(n, m) = O(n + m) + T_{\text{max flow}}(n, m)$$

Reductions

Def: <u>Y reduces to X</u> in polynomial time (notation: $Y \leq_P X$ if any instance of Y can be solved in a polynomial number of computational steps and a polynomial number of calls to <u>black-box</u> that solves instances of X.

Note: to prove correctness of general reduction, must show that correctness (e.g., optimality) of algorithm for X implies correctness of algorithm for Y.

Def: <u>one-call reduction</u> maps instance of Y to instance of X, solution of Y to solution of X. (also called a Karp reduction)

Note: a one-call reduction gives two algorithms:

(a) construction of X^Y instance from Y instance.

(b) construction of Y solution from X^Y solution (with same value).

Note: the proof of correctness of a one-call reduction gives one algorithm:

(c) construction of X^Y solution from Y solution (with same value).

(Only need to consider X^Y instances not general X instances.)

Theorem: reduction from "(a) and (b)" is correct if (a), (b), and (c) are correct.

Proof:

- for instance y of Y, let instance $x^y of X^Y$ be outcome of (a).
- (b) correct $\Rightarrow$ OPT$(y) \geq$ OPT(x^y).
- (c) correct $\Rightarrow$ OPT$(x^y) \geq$ OPT(y).

$\Rightarrow$ OPT$(y) =$ OPT(x^y)

$\Rightarrow$ output of reduction has value OPT(y).