EECS 336: Lecture 7: Introduction to Algorithms

Reductions: network flow, reduction, bipartite matching

Reading: 7.1, 7.5

Last Time:

- Interval Pricing

Today:

- Reductions
- Network flow
- Bipartite matching

Reductions

"to solve problem Y given solution to problem X, transform instances from problem Y into instances of X, solve, transform solution back"

Problem X: 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** 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:** $\forall v \neq s, t$,

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

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

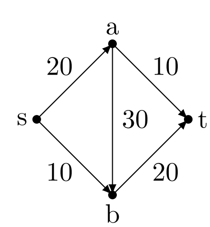
$$|f| = \sum_{e \text{ out of } s} f(e) = \sum_{e \text{ into } t} f(e)$$

Problem: Network Flow

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

output: flow f with maximum value.

Example:



Maxflow = 30.

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

Theorem 2: if capacities are integral, then there is a max flow that is is integral (on each edge) and algorithm finds it.

Problem Y: 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 cardinatlity.

Problem: bipartite matching

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

output: a maximum matching M.

Reducing bipartite matching to max flow

"use max flow alg to solve bipartite matching."

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; see step 3)

(b) convert matching soln to flow soln with same value.

Note: (a) and (b) imply value of max flow = size of max matching.

Step 1:

i. connect s to each $v \in A$ with capacity 1.

ii. connect t to each $u \in B$ with capacity 1.

iii. set capacity of each edge $e \in E$ to 1.

Step 2: compute (integral) max flow f

Step 3: matching $M = e \in E : f(e) = 1$

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

Step 4: Proof:

- any matching M' can be turned into a flow f' with $|f'| = |M'|$

  (send from s to each matched edge to t one unit of flow)

- any integral flow f' can be turned into a matching M' with $|f'| = |M'|$

$\Rightarrow$ 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 besolved in a polynomial number of computational steps and a polynomial number of calls to black-box 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:

I. contruction of X^Y instance from Y instance.

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

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

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

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

Theorem: reduction from "I and II" is correct if I, II, and III are correct.

Proof:

- for instance y of Y, let instance of x^y of X^Y be outcome of I.
- II correct $\Rightarrow$ OPT$(y) \geq$ OPT(x^y).
- III correct $\Rightarrow$ OPT$(x^y) \geq$ OPT(y).

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

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