Lecture 20:
• Applications of Network Flow

Nov 27, 2018
Midterm II Stats
Midterm II Grade Distribution

Midterm II Grades were really good!

Mean ≈ 83
Homework Grade Distribution

I have dropped your lowest grade (so far)
Applications of Network Flow
Applications of Network Flow

If I have seen further than others, it is by standing upon the shoulders of giants.

Isaac Newton
Applications of Network Flow

• Algorithms for maximum flow can be used to solve:
  • Bipartite Matching ✓
  • Disjoint Paths ✓
  • Survey Design
  • Matrix Rounding
  • Auction Design
  • Fair Division
  • Project Selection
  • Baseball Elimination
  • Airline Scheduling
  • ...

...
**Reduction**

- **Definition:** A reduction is an efficient algorithm that solves problem A using calls to a library function that solves problem B.

**Diagram:**

```
Given

Input x for problem A  →  Input u for problem B

Solution y for problem A  ←  Solution v for problem B
```
Mechanics of Reductions

• What exactly is a problem?
  • A set of legal inputs $X$
  • A set $A(x)$ of legal outputs for each $x \in X$

• Example: integer maximum flow

  $X = \{ \text{all arrays of numbers} \}$

  $A(1 \ 3 \ 8 \ 5 \ 7) = \{ 3 \ 5 \ 7 \ 8 \}$
Mechanics of Reductions

• What exactly is a **problem**?
  • A set of legal inputs $X$
  • A set $A(x)$ of legal outputs for each $x \in X$

• **Example:** integer maximum flow

legal inputs: $x = (V, E, s, t, \{c(e)\})$

- $E \subseteq V \times V$
- $s, t \in V$
- $c(e) \in \mathbb{N}$ for every $e \in E$

legal outputs for $x$: Any flow $f$ that maximizes $\text{val}(f)$
  - satisfies capacity, conservation
  - $f(e) \in \mathbb{Z}$
Mechanics of Reductions

- **Input x for Problem B**
- **Output y in B(x) for Problem B**
- **Input u for Problem A**
- **Output v in A(u) for Problem A**
- **SolveA**

 Encode: If $u$ is a legal input for $A$, then $v$ is in the set of legal outputs $A(u)$.

Your job is to design the two pieces in red.
When is a Reduction Correct?

1. Assume that SolveA is a correct algorithm for A (do not assume anything else)
2. Argue that for every legal x for B, u is legal for A
3. Argue that for every x, u_x and every v \in A(u_x), y \in B(x) depends on x
What is the Running Time?

Input $x$ for Problem B

Output $y$ in $B(x)$ for Problem B

Input $u$ for Problem A

Output $v$ in $A(u)$ for Problem A

SolveA

Total time: $1 + 2 + 3$

- $\mathcal{O}$ is the time to solve $A$ on an input of size $|u|$
Example: Minimum Cut

Problem B is min cut
Problem A is max flow

Input x for Problem B

Input u for Problem A

Output v in A(u) for Problem A

Output y in B(x) for Problem B

SolveA

Given $G=(V, E, s, t, \delta(e))$
Find a min s-t cut $(A, B)$

Given $G=(V, E, s, t, \delta(e))$
Find a max s-t flow f
Example: Minimum Cut

\[ G = (V, E, s, t, \xi_c(e)) \]

1. Input \( x \) for Problem B
2. Input \( u \) for Problem A
3. Output \( y \) in \( B(x) \) for Problem B
4. Output \( v \) in \( A(u) \) for Problem A

\[ \text{max flow } f \text{ for } G \]

1. Find residual graph \( G_f \)
2. Run BFS from \( s \) to find nodes reachable in \( G_f \), call it \( A \)
3. \( B = V \setminus A \)
Running Time:

1. $O(m+n)$ to write down the graph

2. $O(m) + O(n+m) + O(n) = O(n+m)$
   - find $G_f$
   - BFS $G_f$
   - write $A, B$

3. Time to find max flow on a graph with $n$ nodes and $m$ edges. $O(mn)$ time.

$O(mn)$
Bipartite Matching

- **Input:** bipartite graph $G = (V, E)$ with $V = L \cup R$
- **Output:** a maximum cardinality matching
  - A matching $M \subseteq E$ is a set of edges such that every node $v$ is an endpoint of at most one edge in $M$
  - Cardinality $= |M|$

Models any problem where one type of object is assigned to another type:
- doctors to hospitals
- jobs to processors
- advertisements to websites

Similar to stable matching
Bipartite Matching

• There is a reduction that uses integer maximum s-t flow to solve maximum bipartite matching.
Step 1: Transform the Input

Input $G$ for MCBM → $G'$ for MAXFLOW

Input $G$ for MCBM

Input $G'$ for MAXFLOW

Time $O(m+n)$

$c(e) = 1$ for every $e \in E$

$n' = n + 2$

$m' = m + n$
Step 2: Receive the Output

Every node on the left has \( \leq 1 \) unit leaving, (Similar for the right)

Red arrow means \( f(e) = 1 \)
Black means \( f(e) = 0 \)

Input \( G' \) for MAXFLOW

Output \( f \) for MAXFLOW

Integer Max Flow

Red arrow means \( f(e) = 1 \)
Black means \( f(e) = 0 \)
Step 3: Transform the Output

$O(m+n)$

Output $M$ for MCBM

Output $f$ for MAXFLOW

$M$ is all edges from $L$ to $R$ s.t. $f(e) = 1$
Reduction Recap

• Step 1: Transform the Input
  • Given G = (L,R,E), produce G’ = (V,E,{c(e)},s,t) by...
    • ... orient edges e from L to R
    • ... add a node s with edges from s to every node in L
    • ... add a node t with edges from every not in R to t
    • ... set all capacities to 1

• Step 2: Receive the Output
  • Find an integer maximum s-t flow in G’

• Step 3: Transform the Output
  • Given an integer s-t flow f(e)...
    • Let M be the set of edges e going from L to R that have f(e)=1
Correctness

• Need to show:

✓  (1) This algorithm returns a matching
  • (2) This matching is a maximum cardinality matching
Correctness

• This algorithm returns a matching
Correctness

• **Claim:** G has a matching of cardinality at least k if and only if G’ has an s-t flow of value at least k

• **Proof (⇒):**

matching w/ k edges
Correctness

• **Claim:** G has a matching of cardinality at least k if and only if G’ has an s-t flow of value at least k

• **Proof (\(\leq\)):**

![Diagram showing flow with value k]
Running Time

- Need to analyze the time for:
  1. Producing $G'$ given $G$ \( O(m+n) \)
  2. Finding a maximum flow in $G'$
  3. Producing $M$ given $G'$ \( O(m+n) \)

\[
T(n', m') = T(n+2, m+n)
\]

$T$ is time to solve max flow in graph with $n'$ nodes, $m'$ edges.

\[
T(n', m') = O(n'm') = O((n+2)(m+n))
= O(mn + 2m + 2n + n^2) = O(mn)
\]
Summary

Solving maximum s-t flow in a graph with \( n+2 \) nodes and \( m+n \) edges and \( c(e) = 1 \) in time \( T \)

\[
\downarrow
\]

Solving maximum bipartite matching in a graph with \( n \) nodes and \( m \) edges in time \( T + O(m+n) \)

- Can solve maximum bipartite matching in time \( O(nm) \) using Ford-Fulkerson
  - Improvement for maximum flow gives improvement for maximum bipartite matching!