2473 Algorithms: Lecture 17

logistics

less time

- more examples + homework in labs
- bijections, counting

more for applications

I. Edge flow

Prop: \( f: E \rightarrow \mathbb{R}_{>0} \) flow \( \Rightarrow \) \( f \) is a single \( s \rightarrow t \) path in \( G \)?

exists \( g: P \rightarrow \mathbb{R}_{>0} \) \( \Rightarrow \) \( C = \{ \} \) simple cycles in \( G \)?

- \( \forall e \in E \quad f(e) = \sum_{p \ni e} g(p) \)

- \( \sum_{p \ni e} g(p) \geq 0 \) \( \Rightarrow \) \( \sum_{C \ni e} g(C) \geq 0 \)

- if \( f \) integral \( \Rightarrow g \) integral

- \( g \) computable in \( O(mn) \) time

Sketch: Ford Fulkerson barracks

algo: find simple path/cycle and subtract from \( f \), until \( f = 0 \)

remove all flows from \( f \) that\( \Rightarrow G \) in \( m \) paths/cycle

computation: \( O(\sqrt{m}) \) easy

use \( O(mn) \) instead for adjacency list, edges of positive flow

integrality: by algorithm

def: circulation \( \mathcal{C} = (V,E) \) capacites <

A circulation is a flow \( \psi \) with \( \psi \) \ni 0

- \( \mathcal{C} \)

A circulation is a positive sum of simple cycles

Sketch: flow decomposition \( \mathcal{C} \) at each \( \psi \ni 0 \)

Q: \( \mathcal{C} = (V,E) \) directed \( s \rightarrow t \) \( V \). What is the maximum number of edge disjoint \( s \rightarrow t \) paths? \( \mathcal{C} \) full \( \Rightarrow \)

def: \( E_1 \subseteq E \) is an \((s,t)-\)edge-cut \( \Rightarrow G - E_1 \) has no \( s \rightarrow t \) path

ex: \( V = \{s,t\} \) \( s \leftarrow t \) \( \mathcal{C} = (V,E) \) edge cut \( \circ(s,t) = \{ (u,v) \in E \mid u \in S, v \in T \} \)
\[ \text{lo} = E' \leq E \text{ (\text{adj. cut}), } \exists V = S, T \leq E' \text{ s.t.} \\
\text{lo} = \exists S \subseteq V : S \leq V \text{ path in } G - E' \text{ s.t.} \\
(S, T) \in E, (V, T) \in E, (V, V) \in E - E' \text{ as } S \subseteq V \text{ s.t. } S \cap T = \emptyset \]

\[ \text{Thm (Menger): } G = (V, E) \text{ directed, } s \neq t \in V, \]

\[ \min \{ |E'| \} = \max \text{ # edge-disjoint } s \rightarrow t \text{ paths} \]

\[ E' \leq E \text{ (adj. cut)} \]

\[ \text{Sketch: view } G \text{ as network, unit capacities, } \text{ capacity } 2 \]

\[ \min \text{ } c(S, T) = \min \text{ } c(S, T') \]

\[ \max \text{ #f} \]

\[ \text{X edge disjoint paths in } G \rightarrow \text{flow value } f \text{ in } G \text{ (clear)} \]

\[ \text{flow value } f \text{ in } G \rightarrow \text{X edge disjoint paths} \]

\[ \text{Apply flow decomposition to gain paths} \]

\[ \text{Integrality in every unit capacity } = \text{X paths are edge disjoint} \]

\[ \text{rule: Menger's thm predicts } \max \text{flow = min cut} \]

\[ \text{also works for undirected graphs} \]

\[ \text{Q: who will make the world series? I think San this year... II} \]

\[ \text{Fact: } \leq 1969: \text{two baseball leagues (National, American) advanced their team} \]

\[ \text{to world series based on best overall season record} \]

\[ \geq 1969: \text{Richer or } \text{"penchant race"} \]

\[ \text{Q: can Baltimore Orioles still win the pennant (with tie?)} \]

<table>
<thead>
<tr>
<th>Team</th>
<th>Wins</th>
<th>Played</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>91</td>
<td>4</td>
</tr>
<tr>
<td>Boston</td>
<td>91</td>
<td>4</td>
</tr>
<tr>
<td>Baltimore</td>
<td>88</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ A: \text{no, cannot exceed } 90 < 91 \]

\[ \text{Q: still win?} \]

<table>
<thead>
<tr>
<th>Team</th>
<th>Wins</th>
<th>Played</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYC</td>
<td>91</td>
<td>4</td>
</tr>
<tr>
<td>BOS</td>
<td>91</td>
<td>4</td>
</tr>
<tr>
<td>BUC</td>
<td>90</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ A: \text{no, cannot exceed } 90 < 91 \]
A: 

- Suppose BU1 wins both games.
  - BU1: 92
- BU2 needs BOS, NYC to lose all remaining games.

  - BOS: 91 + \( \frac{2}{3} \)
  - NYC: 91 + \( \frac{2}{3} \)

  \[ \text{not possible as system must } \min \geq 2 \quad \text{(num. 3g)} \]

\[ \Rightarrow \text{91, 92 > 93 > 92} \quad \text{winnier.} \]

\[ \Rightarrow \text{BU1 cannot (outside) win pennant even if tie} \]

\[ \text{def: The baseball elimination problem} \]

- Teams: 0, 1, ..., n
- Current win: \( w_i \geq 0 \)
- Remaining game: \( g_{ij} \geq 0 \)

\[ \text{want to know if team } 0 \text{ can possibly achieve larger \# wins (or tie)} \]

\[ \text{prop: team } 0 \text{ can win tie percent if} \]

\[ \text{team } 0 \text{ wins all remaining games, totals } W = w_0 + \sum g_{0j} \text{ possible wins} \]

\[ \text{team } k \text{ wins } < W \text{ games, total } < \text{ wins } < W - \text{ remaining games} \]

\[ \text{prop 2 - baseball elimination reduces to max flow} \]

\[ \text{sketch: draw } G = (V, E) \quad V = 5 \quad \text{sink} \]

\[ \text{sink} \quad \sum_{j=1}^{n} g_{ij} \quad \text{sink} \quad 0 \quad \text{sink} \quad \text{sink} \quad \text{sink} \]

\[ \text{Cl: max flow value } = \sum_{j=1}^{n} g_{ij} \quad \text{if team } 0 \text{ can win } \]

\[ = \min \text{ cut} = \sum_{j=1}^{n} g_{ij} \quad \text{via: max flight assignment (0-1)} \]

\[ \text{value of } V = s \oplus \sum_{j=1}^{n} w_j (V / s) \quad \text{can say all games can be assigned} \]

\[ \text{add no one wins too much} \]

\[ \text{frosh: small min cut gives suspect proof that team } U \text{ cannot win unless in challenging } Z \]

\[ \text{def: A rather weak proxy: } \quad G = (V, E) \quad \text{directed} \]

\[ s \neq t \in V \]

\[ c: E \rightarrow \mathbb{R}^+ \quad \text{capacities} \]

\[ \eta: E \rightarrow R \quad \text{prices} \quad \text{possibly negative?} \]

\[ \text{The cost of a flow } f: R \rightarrow \mathbb{R}^+ \quad \text{is } \sum_{e \in E} \eta(e) f(e) \]
Q: find min cost max flow?

rank: generalizes min weight perfect bipartite matching
- generalizes shortest s-t path, if acyclic flow, cycle free
- not known to be reducible to unweighted max flow
- special case of linear programming, II generalizes max flow

then: min cost max flow has polynomial (C, P) time - hope can be improved

Sketch: II like Ford-Fulkerson
create residual graph

Key lem: f min cost flow if & only if G_f has no augmenting cost cycle

Sketch: T \subseteq D: augmenting cost cycle D
- f + D - \text{path from s to t}
- \text{specifies unit cost path from s to t}
- has less cost than D: organic cycle

\[ T + D \leq f \text{ min cost } P(f) < P(f + D) \]

f is flow in G_f, e in G_f \text{ primal}
- value 0 = circulation = \text{sum of simple cycles}
and
- organic cost cycle = \text{some organic cost cycle}

algo:
- find max flow f in G

initialize \( G_f \)
- \text{while organic cost cycle } D \text{ in } G_f \text{ exits }

\[ f \leftarrow f + D \]

return \( f \leftarrow C.P \)

\[ \text{max} = \left( \text{max cost max flow} \right) - \left( \text{min cost max flow} \right) \]
\[ \leq 2 \cdot C.P \]

Remarks: finding organic cost cycle uses Bellman-Ford
- finding max cost-min flow cycle is NP-hard
- efficient algo via augmenting chain \text{Hamiltonian cycle problem}
- cycle of minimum mean cost
- per edge

Issues: flow + edge weight
- flow decomposition
- edge disjoint paths
- base case elimination
- min cost max flow
- \text{Ford-Fulkerson}
- \text{generalizes max flow} II