Maxflow

Mincut problem

Input an edge-weighted digraph, source vertex s and target vertex t

* edge has positive capacity

Def A st-cut (cut) is a partition of the vertices into two disjoint sets with s in one set A and t in the other set B

Def Its capacity is the sum of capacities of the edges from A to B ($s \longrightarrow t$)

* find a cut of minimum capacity

Maxflow problem

Input an edge-weighted digraph, source vertex s and target vertex t

Def an st-flow is an assignment to the edges such that:

- capacity constraint: $0 \le$ edge's flow $\le$ edge's capacity
- local equilibrium: inflow = outflow at every vertex (except s and t)

Def The value of a flow is the inflow at t

* find a flow of maximum value
* these two problems are dual

Ford-Fulkerson Algorithm

<u>Init</u>: start with 0 flow

<u>Idea</u>: increase flow along augmenting paths

* augmenting path: find an undirected path from s to t such as:
  - can increase flow on forward edges (not full)
  - can decrease flow on backward edges (not empty)

* terminates when there are no more augmenting paths

=> all paths from s to t are blocked by either a:

- full forward edge
- empty backward edge

Ford-Fulkerson

-> start with 0 flow

-> while ∃ augmenting path

  -> find augmenting path

  -> compute bottleneck capacity

  -> increase flow on that path by bottleneck capacity

Maxflow - Mincut Theorem

* Relationship between a flow and a cut

Def A net flow across a cut (A, B) is the sum of the flows on its edges from $\boxed{A \text{ to } B}$ minus the sum of the flows on its edges from $\boxed{B \text{ to } A}$

Flow-value lemma: Let f be any flow and let (A, B) be any cut. $\Rightarrow$ the net flow across (A, B) equals the value f

Corollary outflow from s = inflow to t = value of the flow

Weak duality: Let f be any flow and let (A, B) be any cut $\rightarrow$ the value of flow $\leq$ the capacity of the cut

Pf

Value of flow f = net flow across cut (A, B)

$\leq$ capacity of cut (A, B)

- Augmenting path theorem: a flow f is a maxflow iff no augmenting paths
- Maxflow-mincut theorem: value of the maxflow = capacity of mincut

Computing a mincut from a maxflow

To compute mincut (A, B) from maxflow f

- by augmenting path theorem $\Rightarrow$ no augmenting paths with respect to f
- compute A = set of vertices connected to s by an undirected path with no full forward or empty backward edges

Running Time Analysis

- augmenting paths can be found using BFS
- important special case: edge capacities are integers between 1 and U

<u>Invariant</u>: flow is integer-valued throughout Ford-Fulkerson

<u>Proposition</u>

Number of augmentations $\leq$ the value of the maxflow

<u>Pf</u>: each augmentation increases the value by at least 1

<u>Integrality theorem</u>: there exists an integer-valued maxflow

Augmenting paths:

- shortest paths $\leq \frac{1}{2} EV$ paths (BFS)
- fattest path $\leq E \ln(E \cdot V)$ priority queue

<u>Java Implementation</u>

* Flow edge data type: f_e and C_e for edge $e = v \longrightarrow w$

* Flow network data type: need to process edge $e = v \to w$ in either direction: include (e) in both v and w's adjacency lists (similar to undirected graphs)

* <u>Residual capacity</u>
  - forward edge: $r_e = c_e - f_e$
  - backward edge: $r_e = f_e$

* <u>Augment flow</u>
  - forward edge: add Δ
  - backward edge: subtract Δ

* <u>Residual network</u> - a useful view of a flow network

  <u>key point</u>: augmenting path in original network is equivalent to directed path in residual network.

Flow edge API

public class FlowEdge

- int from()
- int to()
- int other(int v)
- double capacity()
- double flow()
- double residualCapacityTo(v)
- void addResidualFlowTo(int v, double)

public class FlowNetwork

- FlowNetwork(int V)
- void addEdge(FlowEdge e)
- Iterable<FlowEdge> adj(int v)
- Iterable<FlowEdge> edges()
- int V(), int E()

Maxflow Applications

* Bipartite matching problem
* where no perfect matching ⇒ mincut explains why

* Baseball Elimination Problem