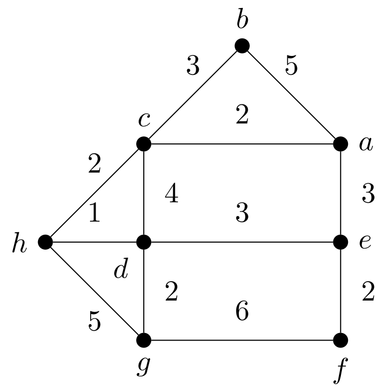
Minimum Cuts in Unidrected Graphs

Recall: $\delta(S)$ is the set of edges with exactly one endpoint in S and we also write $u(\delta(S)) = \sum_{e\in\delta(S)} u(e)$.
Note: $\delta(S) = \delta(V(G) \setminus S)$.

Minimum Cut Problem:
Input: Graph $G = (V, E)$ and cost function $u : E \to \mathbb{R}^+$. (cost for minimizing)
Output: Global Minimum Cut. That is $S \subset V$ which minimizes $u(\delta(S))$

1: Find a minimum cut in the following graph:



Notation:

$\lambda(G)$ is the cost of minimum cut of G, i.e.

$$\lambda(G) = \min_{\emptyset\neq S\subset V(G)} \sum_{e\in\delta(S)} u(e)$$

$\lambda(G; v, w)$ is the cost of minimum (v, w)-cut of G, i.e.

$$\lambda(G; v, w) = \min_{v\in S\subseteq V(G)\setminus\{w\}} \sum_{e\in\delta(S)} u(e)$$

2: Find an algorithm for Minimum Cut Problem using network flows.

Solution: Fix any vertex, find a maximum flow to every other vertex, and take the minimum. This max-flow gives a globally minimum cut. Why this works?

Node Identification Algorithm:

Let G_{uv} be a graph obtained from G by identifying u and v (delete loops, keep parallel edges).

Main idea:

$$\lambda(G) = \min(\lambda(G_{vw}), \lambda(G; v, w)) \tag{1}$$

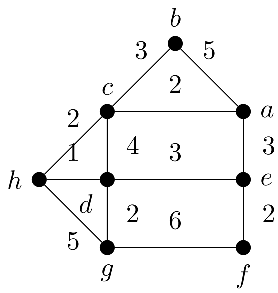
3: Explain (1).

Solution: A minimum cut in G either separates u from v or does not.

How can we make $\lambda(G; v, w)$ easy to calculate? By cleverly picking v and w?

A legal ordering of vertices starting at v_1 is $v_1, v_2, \ldots, v_n$ if for all i, v_i has the largest cost of edges joining it to $v_1, \ldots, v_{i-1}$.

4: Find a legal ordering starting with vertex a of the graph from the first exercise (redrawn below)



Solution: a, b, c, d, e, h, g, f

Main theorem: If $v_1, \ldots, v_n$ is a legal ordering of G, then $\delta(v_n)$ is a minimum v_n, v_{n-1} cut of G.

Node Identification Algorithm:

1. $M := \infty$ and A undefined
2. while G has more than 1 vertex
3. Find a legal ordering $v_1, v_2, \ldots, v_n$ of G
4. If $u(\delta(v_n)) < M$
5. $M := u(\delta(v_n))$ and $A := \delta(v_n)$
6. $G := G_{v_n v_{n-1}}$
7. return A

5: Run the node identification algorithm on the graph from the previous exercise.

Solution: Many figures needed here...

Random Contraction Algorithm:

1. while G has more than 2 vertices
2. Choose an edge e of G with probability $u(e)/u(E)$
3. $G := G_{vw}$, where $e = vw$
4. return the unique cut in G.

6: Let A be a minimum cut of an n-vertex graph G. Show that the random contraction algorithm returns A with probability at least $2/(n(n-1))$.
What is the probability that a random cut in G is a minimum cut? (The algorithm does something.)

Solution: Let $u(A) = \sum_{e\in A} u(A)$. Then

$$P(\text{edge of } A \text{ is picked for contraction}) = \frac{u(A)}{u(E)}$$

Notice that A is the minimum cut in G. Hence $u(A) \leq u(C)$ for any other cut. In particular, we consider cuts around each vertex. A cut around vertex v has cost $\sum_{e\in\delta(v)} u(e)$. The average cost of a cut around one vertex is

$$\frac{\sum_{e\in\delta(v)} u(e)}{n} = \frac{2\sum_{e\in E} u(e)}{n} = \frac{2u(E)}{n}.$$

Then picking an edge from A has lower probability than picking an edge from an average cut around a vertex

$$\frac{u(A)}{u(E)} \leq \frac{2u(E)}{n \cdot u(E)} = \frac{2}{n}.$$

After i rounds of the algorithm, G has $n - i$ edges and we get

$$\frac{u(A)}{u(E)} \leq \frac{2}{n-i}.$$

Now the probability that no edge of A was choses is at least

$$1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}$$

The algorithm is running for rounds with $i = 0, \ldots, n-2$ and we get that the probability no edge of A is ever chosen is at least

$$\frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)}.$$

7: Let $k \in \mathbb{N}$. Show that the probability that the random contraction algorithm does not return A in one of kn^2 runs is at most e^{-2k}.

Solution: We use the estimate from previous round kn^2 times.

$$\left(1 - \frac{2}{n(n-1)}\right)^{kn^2} \leq \left(1 - \frac{2}{n^2}\right)^{kn^2} \leq \left(e^{-\frac{2}{n^2}}\right)^{kn^2} = e^{-2k}.$$