Min cut

$G = (V, E)$: undirected graph, $n$ vertices, $m$ edges.

Interested in cuts in $G$.

Definition
cut in $G$: a partition of $V$: $S$ and $V \setminus S$.
Edges of the cut:
$$(S, V \setminus S) = \{uv \mid u \in S, v \in V \setminus S, \text{ and } uv \in E\},$$

$|(S, V \setminus S)|$ is size of the cut

minimum cut / mincut: cut in graph with min size.

Some definitions

1. conditional probability of $X$ given $Y$ is
   $$\Pr[X = x \mid Y = y] = \frac{\Pr[(X = x) \cap (Y = y)]}{\Pr[Y = y]}.$$

2. $X, Y$ events are independent, if
   $$\Pr[X = x \cap Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

Some more probability

Lemma
$\mathcal{E}_1, \ldots, \mathcal{E}_n$: $n$ events (not necessarily independent). Then,

$$\Pr[\cap_{i=1}^n \mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \mid \mathcal{E}_1] \cdot \Pr[\mathcal{E}_3 \mid \mathcal{E}_1 \cap \mathcal{E}_2] \cdot \ldots \cdot \Pr[\mathcal{E}_n \mid \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-1}].$$
**Edge contraction...**

![Graph G with edge xy](image1.png)

1. **edge contraction**: $e = xy$ in $G$.
2. ... merge $x, y$ into a single vertex.
3. ...remove self loops.
4. ... parallel edges – **multi-graph**.
5. ... weights/ multiplicities on the edges.

**Cuts under contractions**

**Observation**

1. A cut in $G/xy$ is a valid cut in $G$.
2. There $\exists$ cuts in $G$ are not in $G/xy$.
3. The cut $S = \{x\}$ is not in $G/xy$.
4. $\implies$ size mincut in $G/xy \geq$ mincut in $G$.

1. **Idea**: Repeatedly perform edge contractions (benefits: shrink graph)... 
2. Every vertex in contracted graph is a connected component in the original graph.

**Min cut in weighted graph**

![Graph G and G/xy](image2.png)

Edge contraction implemented in $O(n)$ time:

1. Graph represented using adjacency lists.
2. Merging the adjacency lists of the two vertices being contracted.
3. Fix adjacency list of vertices connected to $x, y$.
   [Use radix sort.]
4. Include edge weight in computing cut weight.

**Contraction**

![Graph contractions](image3.png)

(2) (3) (4)
The algorithm...

```
Algorithm MinCut(G)
    \( G_0 \leftarrow G \)
    \( i = 0 \)
    while \( G_i \) has more than two vertices do
        \( e_i \leftarrow \) random edge from \( E(G_i) \)
        \( G_{i+1} \leftarrow G_i / e_i \)
        \( i \leftarrow i + 1 \)
    Let \( (S, V \setminus S) \) be the cut in the original graph corresponding to the single edge in \( G_i \)
return \( (S, V \setminus S) \).
```

But...

1. Not min cut!
2. Contracted wrong edge somewhere...
3. If never contract an edge in the cut...
4. ...get min cut in the end!
5. We might still get min cut even if we contract edge min cut. Why???
How to pick a random edge?

**Lemma**

\[ X = \{x_1, \ldots, x_n\}: \text{elements, } \omega(x_i): \text{integer positive weight.} \]

Pick randomly, in \(O(n)\) time, an element \(x \in X\), with prob

\[ \text{picking } x_i \text{ being } \omega(x_i)/W, \text{ where } W = \sum_{i=1}^{n} \omega(x_i). \]

**Proof.**

Randomly choose \(r \in [0, W]\).

Precompute \(\beta_i = \sum_{k=1}^{i} \omega(x_k) = \beta_{i-1} + \omega(x_i).\)

Find first index \(i, \beta_{i-1} < r \leq \beta_i\). Return \(x_i\).

1. Edges have weight...
2. ...compute total weight of each vertex (adjacent edges).
3. Pick randomly a vertex by weight.
4. Pick random edge adjacent to this vertex.

---

**Lemma**

\( G: \text{mincut of size } k \text{ and } n \text{ vertices, then } |E(G)| \geq \frac{kn}{2}. \)

**Proof.**

Each vertex degree is at least \(k\), otherwise the vertex itself would form a minimum cut of size smaller than \(k\). As such, there are at least \(\sum_{v \in V} \text{degree}(v)/2 \geq nk/2\) edges in the graph.

---

**Lemma**

If we pick in random an edge \(e\) from a graph \(G\), then with probability at most \(\frac{2}{n}\) it belong to the minimum cut.

**Proof.**

There are at least \(nk/2\) edges in the graph and exactly \(k\) edges in the minimum cut. Thus, the probability of picking an edge from the minimum cut is smaller then \(k/(nk/2) = 2/n\).

---

**Lemma**

\( \text{MinCut} \text{ outputs the mincut with prob. } \geq \frac{2}{n(n-1)}. \)

**Proof**

1. \(E_i: \text{ event that } e_i \text{ is not in the minimum cut of } G_i.\)
2. \(\text{MinCut} \text{ outputs mincut if all the events } E_0, \ldots, E_{n-3} \text{ happen.}\)
3. \(\Pr[E_i | E_0 \cap E_1 \cap \ldots \cap E_{i-1}] \geq 1 - \frac{2}{|V(G_i)|} = 1 - \frac{2}{n-i}.\)

\[ \Delta = \Pr[E_0 \cap \ldots \cap E_{n-3}] = \Pr[E_0] \cdot \Pr[E_1 | E_0] \cdot \Pr[E_2 | E_0 \cap E_1] \cdot \ldots \cdot \Pr[E_{n-3} | E_0 \cap \ldots \cap E_{n-4}] \]
Proof continued...

As such, we have

\[ \Delta \geq \prod_{i=0}^{n-3} \left( 1 - \frac{2}{n-i} \right) = \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n \cdot (n-1)}. \]

Some math restated...

\[ \alpha = \left( 1 - \frac{2}{n} \right) \left( 1 - \frac{2}{n-1} \right) \left( 1 - \frac{2}{n-2} \right) \cdots \left( 1 - \frac{2}{4} \right) \left( 1 - \frac{2}{3} \right) \]

\[ = \frac{n-2}{n} \cdot \frac{n-1}{n-1} \cdot \frac{n-2}{n-2} \cdots \frac{4}{4} \cdot \frac{3}{3} \]

\[ = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{2}{4} \cdot \frac{1}{3} \]

\[ = \frac{2}{n(n-1)} \]

Running time analysis...

Observation

MinCut runs in \( O(n^2) \) time.

Observation

The algorithm always outputs a cut, and the cut is not smaller than the minimum cut.

Definition

Amplification: running an experiment again and again till the things we want to happen, with good probability, do happen.

Getting a good probability

MinCutRep: algorithm runs MinCut \( n(n-1) \) times and return the minimum cut computed.

Lemma

probability MinCutRep fails to return the minimum cut is < 0.14.

Proof.

MinCut fails to output the mincut in each execution is at most \( 1 - \frac{2}{n(n-1)} \).

MinCutRep fails, only if all \( n(n-1) \) executions of MinCut fail.

\[ \left( 1 - \frac{2}{n(n-1)} \right)^{n(n-1)} \leq \exp \left( -\frac{2}{n(n-1)} \cdot n(n-1) \right) = \exp(-2) < 0.14, \] since \( 1 - x \leq e^{-x} \) for \( 0 \leq x \leq 1 \).
Result

Theorem
One can compute mincut in \( O(n^4) \) time with constant probability to get a correct result. In \( O(n^4 \log n) \) time the minimum cut is returned with high probability.

Faster algorithm

Why \textbf{MinCutRep} needs so many executions?
Probability of failure in first \( \nu \) iterations is
\[
\Pr[E_0 \cap \ldots \cap E_{\nu-1}] \geq \prod_{i=0}^{\nu-1} \left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{\nu-1} \frac{n-i-2}{n-i} \\
= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \ldots \\
= \frac{(n-\nu)(n-\nu-1)}{n \cdot (n-1)}.
\]

\( \implies \nu = n/2 \): Prob of success \( \approx 1/4 \).
\( \implies \nu = n - \sqrt{n} \): Prob of success \( \approx 1/n \).

Faster algorithm...

Insight
1. As the graph get smaller probability for bad choice increases.
2. Currently do the amplification from the outside of the algorithm.
3. Put amplification directly into the algorithm.

Contract...

\textbf{Contract}(G, t) shrinks G till it has only t vertices. \textbf{FastCut} computes the minimum cut using \textbf{Contract}.

\begin{verbatim}
FastCut(G = (V, E))
  // G -- multi-graph
  n \leftarrow |V(G)|
  if \ n \leq 6 then return mincut G (brute force)
  t \leftarrow \left\lceil 1 + n/\sqrt{2} \right\rceil
  H_1 \leftarrow \textbf{Contract}(G, t)
  H_2 \leftarrow \textbf{Contract}(G, t)
  /* Contract is randomized */
  X_1 \leftarrow \textbf{FastCut}(H_1),
  X_2 \leftarrow \textbf{FastCut}(H_2)
  return mincut of X_1 and X_2.
\end{verbatim}
Lemma...  

**Lemma**  
The running time of \( \text{FastCut}(G) \) is \( O(n^2 \log n) \), where \( n = |V(G)| \).

**Proof.**  
Well, we perform two calls to \( \text{Contract}(G, t) \) which takes \( O(n^2) \) time. And then we perform two recursive calls on the resulting graphs. We have:  
\[
T(n) = O(n^2) + 2T\left(\frac{n}{\sqrt{2}}\right)
\]

The solution to this recurrence is \( O(n^2 \log n) \) as one can easily (and should) verify.

Proof at each step  

**Lemma**  
Probability that mincut in contracted graph is original mincut is at least \( 1/2 \).

**Proof.**  
Plug in \( \nu = n - t = n - \left[1 + n/\sqrt{2}\right] \) into success probability:

\[
\Pr\left[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-t}\right] \geq \frac{t(t-1)}{n(n-1)} \geq \frac{1}{2}
\]

Probability of success...  

**Lemma**  
\( \text{FastCut} \) finds the minimum cut with probability larger than \( \Omega(1/\log n) \).

See class notes for a formal proof. We provide a more elegant direct argument shortly.

Amplification  

**Lemma**  
Running \( \text{FastCut} \) repeatedly \( c \cdot \log^2 n \) times, guarantee that the algorithm outputs mincut with probability \( \geq 1 - 1/n^2 \). \( c \) is a constant large enough.

**Proof.**  
1. \( \text{FastCut} \) succeeds with prob \( \geq c'/\log n \), \( c' \) is a constant.
2. ...fails with prob. \( \leq 1 - c'/\log n \).
3. ...fails in \( m \) reps with prob. \( \leq (1 - c'/\log n)^m \). But then
\[
(1 - c'/\log n)^m \leq (e^{-c'/\log n})^m \leq e^{-mc'/\log n} \leq \frac{1}{m^2},
\]
for \( m = (2\log^2 n)/c' \).
Theorem

One can compute the minimum cut in a graph $G$ with $n$ vertices in $O(n^2 \log^3 n)$ time. The algorithm succeeds with probability $\geq 1 - 1/n^2$.

Proof.
We do amplification on **FastCut** by running it $O(\log^2 n)$ times. The running time bound follows from lemma... \qed