The following, is one of the most beautiful algorithms I know.

1 Min Cut

Compute the cut with minimum number of edges in the graph. Namely, find $S \subseteq V$ such that $(S \times (V \setminus S)) \cap E$ is as small as possible, and $S$ is neither empty nor all the vertices of the graph $G = (V, E)$.

1.1 Some Definitions

**Definition 1.1** The conditional probability of $X$ given $Y$ is

$$\Pr[X = x | Y = y] = \frac{\Pr[(X = x) \cap (Y = y)]}{\Pr[Y = y]}.$$ 

An equivalent, useful statement of this is that:

$$\Pr[(X = x) \cap (Y = y)] = \Pr[X = x | Y = y] \cdot \Pr[Y = y].$$

Two events $X$ and $Y$ are independent, if $P[X = x \cap Y = y] = P[X = x] \cdot P[Y = y]$. In particular, if $X$ and $Y$ are independent, then

$$\Pr[X = x | Y = y] = \Pr[X = x].$$

Let $\eta_1, \ldots, \eta_n$ be $n$ events which are not necessarily independent. Then,

$$\Pr[\cap_{i=1}^n \eta_i] = \Pr[\eta_1] \cdot \Pr[\eta_2 | \eta_1] \cdot \Pr[\eta_3 | \eta_1 \cap \eta_2] \cdots \cdot \Pr[\eta_n | \eta_1 \cap \ldots \cap \eta_{n-1}]$$
2 Algorithm

The basic operation, is edge contraction:

![Graph diagram]

We take an edge $e = xy$ and merge the two vertices into a single vertex. What we get:

![Graph diagram]

The new graph is denoted by $G/xy$.

Note, that we remove self loops.

Note, that the resulting graph is no longer a regular graph, it has parallel edges - namely, it is a multi-graph. We represent a multigraph, as a regular graph with multiplicities on the edges.

![Graph diagram]

The edge contraction operation can be implemented in $O(n)$ time for a graph with $n$ vertices. This is done by Merging the adjacency lists of the two vertices being contracted, and then using hashing to do the fixups (i.e., we need to fix the adjacency list of the vertices that are connected to the two vertices).

Q: What is the size of the min-cut in $G/xy$?

![Graph diagram]

Note, that the cut is now computed counting multiplicities (i.e., if an edge is in the cut, and it has weight $w$ we add $w$ to the weight of the cut.

![Graph diagram]
Observation 2.1 The size of the min-cut in $G/xy$ is at least as large as the min-cut in $G$ (as long as $G/xy$ has at least one edge). Since any cut in $G/xy$ has a corresponding cut of the same cardinality in $G$.

Idea: Contraction is good because it shrinks the graph. So contract repeatedly, and compute the min-cut on the resulting graph.

Observation 2.2 Let $e_1, \ldots, e_{n-2}$ be a sequence of edges in $G$, such that none of them is in the min-cut, and such that $G' = G / \{e_1, \ldots, e_{n-2}\}$ is a single multi-edge. Then, this
multi-edge correspond to the min-cut in $G$.

Problem: Argumentation is circular, how can we find a sequence of edges that are not in the cut without knowing what the cut is???

**Lemma 2.3** If a graph $G$ has a min-cut of size $k$, and it has $n$ vertices, then $|E(G)| \geq \frac{kn}{2}$.

*Proof:* Vertex degree is at least $k$. Now count the number of edges... ■

**Lemma 2.4** If we pick in random an edge $e$ from a graph $G$, then with probability at most $\frac{2}{n}$ it belong to the min-cut.

*Proof:* There are at least $nk/2$ edges in the graph and exactly $k$ edges in the cut. Thus, the probability of picking an edge from the min-cut is small then $k/(nk/2) = 2/n$. ■

### Algorithm MinCut

1. $G_0 \leftarrow G$
2. $i = 0$
3. While $G_i$ has more than two vertices do:
   a. Pick randomly an edge $e_i$ from the edges of $G_i$
   b. $G_{i+1} \leftarrow G_i/e_i$
   c. $i \leftarrow i + 1$
4. Let $(S, V - S)$ be the cut in the original graph corresponding to the single edge in $G_i$

### Observation 2.5

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

### Observation 2.6

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

### Lemma 2.7

MinCut outputs the min cut in probability $\geq 2/n(n - 1)$.

*Proof:* Let $\eta_i$ be the event that $e_i$ is not in the min-cut of $G_i$. Clearly, MinCut outputs the minimum cut if the events $\eta_0, \ldots, \eta_{n-3}$ all happen (namely, all edges picked are outside the min cut).

By the above lemma,

$$\Pr[\eta_i | \eta_1 \cap \ldots \cap \eta_{i-1}] \geq 1 - \frac{2}{|V(G_i)|} = 1 - \frac{2}{n - i}$$

Thus,

$$\Pr[\eta_0 \cap \ldots \cap \eta_{n-2}] = \Pr[\eta_0] \cdot \Pr[\eta_1 | \eta_0] \cdot \Pr[\eta_2 | \eta_0 \cap \eta_1] \cdot \ldots \cdot \Pr[\eta_{n-3} | \eta_0 \cap \ldots \cap \eta_{n-4}]$$

4
Thus,
\[
\Pr[\eta_0 \cap \ldots \cap \eta_{n-2}] \geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{n-3} \frac{n - i - 2}{n - i} = \frac{n - 2}{n} \cdot \frac{n - 3}{n - 1} \cdot \frac{n - 4}{n - 2} \cdot \ldots
\]
\[
= \frac{2}{n \cdot (n-1)}.
\]

**Definition 2.8** (informal) Amplification is the process of running an experiment again and again till the things we want to happened with good probability, do happened.

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

**Lemma 2.9** The probability that MinCutRep fails to return the min-cut is < 0.14.

**Proof:** The probability of failure is at most
\[
\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)} \leq \exp\left(-\frac{2}{n(n-1)} \cdot n(n-1)\right) = \exp(-2) < 0.14,
\]
since \(1 - x \leq e^{-x}\) for \(0 \leq x \leq 1\), as you can (AND SHOULD) verify.

**Theorem 2.10** One can compute the min-cut in \(O(n^4)\) time with constant probability to get a correct result. In \(O(n^4 \log n)\) time the min-cut is returned with high probability.

Note: Algorithm is extremely simple, can we complicate things, and get a faster algorithm?

So, why is the algorithm needs so many executions? Well, the probability of success in the first \(l\) iterations, is
\[
\Pr[\eta_0 \cap \ldots \cap \eta_{l-1}] \geq \prod_{i=0}^{l-1} \left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{l-1} \frac{n - i - 2}{n - i} = \frac{n - 2}{n} \cdot \frac{n - 3}{n - 1} \cdot \frac{n - 4}{n - 2} \cdot \ldots = \frac{(n - l)(n - l - 1)}{n \cdot (n-1)}.
\]

Namely, this probability deteriorates very quickly toward the end of the execution, when the graph become small enough.

**Observation 2.11** As the graph get smaller, the probability to make a bad choice increases. So, run the algorithm more times when the graph get small.

<table>
<thead>
<tr>
<th>Contract( G, t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. While</td>
</tr>
<tr>
<td>(a) Pick a random edge e in G.</td>
</tr>
<tr>
<td>(b) G ← G/e</td>
</tr>
<tr>
<td>2. Return G</td>
</tr>
</tbody>
</table>

5
Namely, Contract(\( G, t \)) shrinks \( G \) till it has only \( t \) vertices.

|--FastCut(\( G = (V, E) \))
|\( G \)- multigraph
|1. \( n \leftarrow |V(G)| \)
|2. If \( n \leq 6 \) then
|   compute min-cut of \( G \) using brute force and return cut.
|3. \( t \leftarrow n/\sqrt{2} \)
|4. \( H_1 \leftarrow \text{Contract}(G, t) \)
|5. \( H_2 \leftarrow \text{Contract}(G, t) ~/\text{Contract is randomized!!} ~/\)
|6. \( X_1 \leftarrow \text{FastCut}(H_1), X_2 \leftarrow \text{FastCut}(H_2) \)
|7. Return the smaller cut out of \( X_1 \) and \( X_2 \).

**Lemma 2.12** The running time of FastCut(\( G \)) is \( O(n^2 \log n) \), where \( n = |V(G)| \).

*Proof:* Well, we perform two calls to 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.

**Exercise 2.13** Show that one can modify FastCut so that it uses only \( O(n^2) \) space.

**Lemma 2.14** The probability that Contract(\( G, n/\sqrt{2} \)) had NOT contracted the min-cut is at least \( 1/2 \).

*Proof:* Just plug in \( t = n/\sqrt{2} \) into the equation:

\[
\Pr[\eta_0 \cap \cdots \cap \eta_{n-t}] \geq \frac{t(t-1)}{n \cdot (n-1)},
\]

that we computed above.

**Theorem 2.15** FastCut finds the min-cut with probability larger than \( \Omega(1/\log n) \).

*Proof:* Let \( P(n) \) be the probability that the algorithm succeeds on a graph with \( n \) vertices.

The probability to succeed in the first call on \( H_1 \) is the probability that contract did not hit the min cut (this probability is larger than 1/2 by the above lemma), times the probability that the algorithm succeeded on \( H_1 \) (those two events are independent). Thus, the probability to succeed on the call on \( H_1 \) is at least \((1/2) \cdot P(n/\sqrt{2})\), Thus, the probability to fail on \( H_1 \) is \( \leq 1 - \frac{1}{2} P\left(\frac{n}{\sqrt{2}}\right) \).
The probability to fail on both $H_1$ and $H_2$ is smaller than

$$\left(1 - \frac{1}{2} P\left(\frac{n}{\sqrt{2}}\right)\right)^2.$$ 

And thus, the probability for the algorithm to succeed is

$$P(n) \geq 1 - \left(1 - \frac{1}{2} P\left(\frac{n}{\sqrt{2}}\right)\right)^2 = P\left(\frac{n}{\sqrt{2}}\right) - \frac{1}{4} \left( P\left(\frac{n}{\sqrt{2}}\right) \right)^2.$$ 

We need to solve this recurrence. Divide both sides of the equation by $P(n/\sqrt{2})$ we have:

$$\frac{P(n)}{P(n/\sqrt{2})} \geq 1 - \frac{1}{4} P(n/\sqrt{2}).$$

It is now easy to verify that this inequality holds for $P(n) \geq c/\log n$ (since the worst case is $P(n) = c/\log n$ we verify this inequality for this value). Indeed,

$$\frac{c/\log n}{c/\log(n/\sqrt{2})} \geq 1 - \frac{c}{4 \log(n/\sqrt{2})}.$$ 

$$\frac{\log n - \log \sqrt{2}}{\log n} \geq \frac{4(\log n - \log \sqrt{2}) - c}{4(\log n - \log \sqrt{2})}.$$ 

Let $\Delta = \log n$

$$\frac{\Delta - \log \sqrt{2}}{\Delta} \geq \frac{4(\Delta - \log \sqrt{2}) - c}{4(\Delta - \log \sqrt{2})}$$

and

$$4(\Delta - \log \sqrt{2})^2 \geq 4\Delta(\Delta - \log \sqrt{2}) - c\Delta.$$ 

Which implies

$$-8\Delta \log \sqrt{2} + 4\log^2 \sqrt{2} \geq -4\Delta \log \sqrt{2} - c\Delta$$

$$c\Delta - 4\Delta \log \sqrt{2} + 4\log^2 \sqrt{2} \geq 0,$$

which clear holds for $c \geq 4 \log \sqrt{2}$.

We conclude, that the algorithm succeeds in finding the min-cut in probability $\geq 2 \log 2/\log n$. (Note that the base of the induction holds because we use brute force, and then $P(i) = 1$ for small $i$.)

**Exercise 2.16** Prove, that running FastCut $c \cdot \log^2 n$ times, guarantee that the algorithm outputs the min-cut with probability $\geq 1 - 1/n^2$, say, for $c$ a constant large enough.