COMP4121 Advanced Algorithms

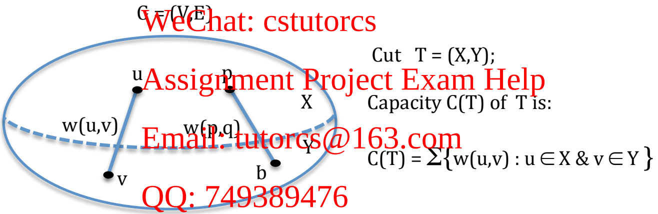
Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

More randomized algorithms: Karger's MinCut Algorithm

Karger's MinCut Algorithm

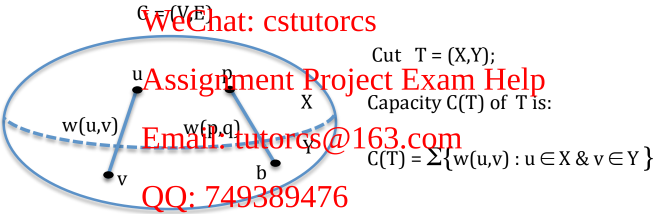
- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* T in G is the total sum of weights of all edges which have one end in X and the other in Y.



- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

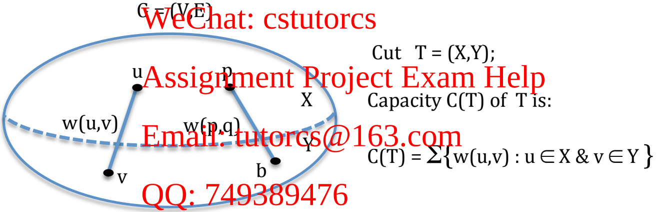
- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* $T = (X, Y)$ in G is the total sum of weights of all edges which have one end in X and the other in Y.



- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

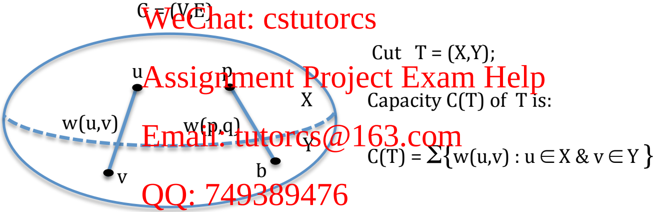
- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* $T = (X, Y)$ in G is the total sum of weights of all edges which have one end in X and the other in Y.



- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

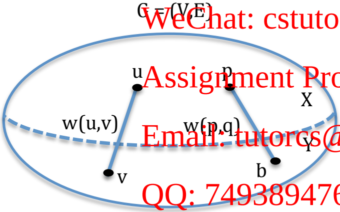
- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* $T = (X, Y)$ in G is the total sum of weights of all edges which have one end in X and the other in Y.



- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* $T = (X, Y)$ in G is the total sum of weights of all edges which have one end in X and the other in Y.



Cut T = (X,Y);

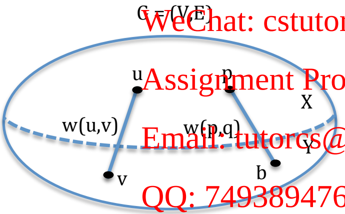
Capacity C(T) of T is:

C(T) = $\Sigma\{$w(u,v) : u $\in$ X & v $\in$ Y$\}$

- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

- Assume you are given an undirected, connected weighted graph $G = (V, E)$, with weights of all edges positive reals.
- A *cut* $T = (X, Y)$ in G is a partition of the set of vertices V into two non empty disjoint subsets X and Y such that $V = X \cup Y$.
- The capacity of a *cut* $T = (X, Y)$ in G is the total sum of weights of all edges which have one end in X and the other in Y.



Cut T = (X,Y);

Capacity C(T) of T is:

$C(T) = \Sigma\{w(u,v) : u \in X \,\&\, v \in Y\}$

- A cut $T = (X, Y)$ in G is a *minimal cut* if it has the lowest capacity among all cuts in G.
- We say that an edge $e(u, v)$ belongs to a cut $T = (X, Y)$ if one of its vertices belongs to X and the other belongs to Y.

Karger's MinCut Algorithm

- The task is, given a co[illegible]aph G find a cut of minimal possible capacity.
- Let (X, Y) be a cut in [illegible]inimal possible capacity, let x be any vertex in X and y any vertex [illegible]
- Since G is undirected, if we replace every edge of G with two directed edges in opposite directions, in such flow network this cut (X, Y) would also be the smallest capacity cut with x as the source and y as the sink.

- Thus, we could fix one arbitrary vertex x of G as the source and try all other vertices as possible sinks and run a Max Flow algorithm on such flow networks to find the corresponding Min Cuts and then among such produced cuts pick the one with the smallest capacity.

- However, this results in a very slow algorithm – the fastest Max Flow algorithm to date runs in time $O(|V|^3)$ so such algorithm would run in time $O(n^4)$ which is very slow for large n.

Karger's MinCut Algorithm

- The task is, given a co[illegible]aph G find a cut of minimal possible capacity.
- Let (X, Y) be a cut in [illegible]inimal possible capacity, let x be any vertex in X and y any vertex [illegible]
- Since G is undirected, if we replace every edge of G with two directed edges in opposite directions, in such flow network this cut (X, Y) would also be the smallest capacity cut with x as the source and y as the sink.

- Thus, we could fix one arbitrary vertex x of G as the source and try all other vertices as possible sinks and run a Max Flow algorithm on such flow networks to find the corresponding Min Cuts and then among such produced cuts pick the one with the smallest capacity.
- However, this results in a very slow algorithm – the fastest Max Flow algorithm to date runs in time $O(|V|^{[illegible]})$ so such algorithm would run in time $O(n^4)$ which is very slow for large n.

- The task is, given a co[illegible]aph G find a cut of minimal possible capacity.
- Let (X, Y) be a cut in [illegible]minimal possible capacity, let x be any vertex in X and y any vertex [illegible]
- Since G is undirected, if we replace every edge of G with two directed edges in opposite directions, in such flow network this cut (X, Y) would also be the smallest capacity cut with x as the source and y as the sink.

- Thus, we could fix one arbitrary vertex x of G as the source and try all other vertices as possible sinks and run a Max Flow algorithm on such flow networks to find the corresponding Min Cuts and then among such produced cuts pick the one with the smallest capacity.

- However, this results in a very slow algorithm – the fastest Max Flow algorithm to date runs in time $O(|V|^3)$ so such algorithm would run in time $O(n^4)$ which is very slow for large n.

- The task is, given a co[illegible]aph G find a cut of minimal possible capacity.
- Let (X, Y) be a cut in [illegible]inimal possible capacity, let x be any vertex in X and y any vertex [illegible]
- Since G is undirected, if we replace every edge of G with two directed edges in opposite directions, in such flow network this cut (X, Y) would also be the smallest capacity cut with x as the source and y as the sink.

- Thus, we could fix one arbitrary vertex x of G as the source and try all other vertices as possible sinks and run a Max Flow algorithm on such flow networks to find the corresponding Min Cuts and then among such produced cuts pick the one with the smallest capacity.

- However, this results in a very slow algorithm – the fastest Max Flow algorithm to date runs in time $O(|V|^3)$ so such algorithm would run in time $O(n^4)$ which is very slow for large n.

Karger's MinCut Algorithm

- We now design a much [illegible]domised algorithm in two stages, refining in the second stage the a[illegible]signed in the first stage.
- The basic operation: c[illegible]an edge $e(u, v)$ by fusing the two vertices u and v into a single vertex $[uv]$ and replacing edges $e(u, x)$ and $e(v, x)$ by a single edge $e([uv], x)$ of weight $w([uv], x) = w(u, x) + w(v, x)$:

- We denote thus obtained graph as G_{uv}

Karger's MinCut Algorithm

- We now design a much [illegible]domised algorithm in two stages, refining in the second stage the a[illegible]signed in the first stage.

- The basic operation: c[illegible]an edge $e(u, v)$ by fusing the two vertices u and v into a single vertex $[uv]$ and replacing edges $e(u, x)$ and $e(v, x)$ by a single edge $e([uv], x)$ of weight $\mathrm{w}([uv], x) = \mathrm{w}(u, x) + \mathrm{w}(v, x)$:

- We denote thus obtained graph as G_{uv}

Karger's MinCut Algorithm

- We now design a much ... domised algorithm in two stages, refining in the second stage the a... signed in the first stage.

- The basic operation: c... an edge $e(u, v)$ by fusing the two vertices u and v into a single vertex $[uv]$ and replacing edges $e(u, x)$ and $e(v, x)$ by a single edge $e([uv], x)$ of weight $\mathrm{w}([uv], x) = \mathrm{w}(u, x) + \mathrm{w}(v, x)$:

- We denote thus obtained graph as G_{uv}

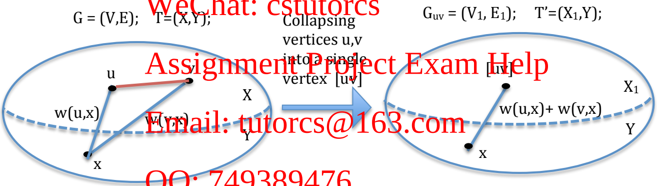
Karger's MinCut Algorithm

- We now design a much [simpler ran]domised algorithm in two stages, refining in the second stage the a[lgorithm de]signed in the first stage.

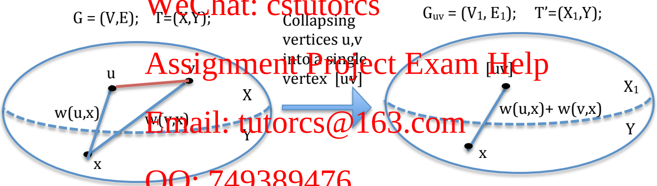
- The basic operation: c[ollapsing] an edge $e(u, v)$ by fusing the two vertices u and v into a single vertex $[uv]$ and replacing edges $e(u, x)$ and $e(v, x)$ by a single edge $e([uv], x)$ of weight $\mathrm{w}([uv], x) = \mathrm{w}(u, x) + \mathrm{w}(v, x)$:

- We denote thus obtained graph as G_{uv}

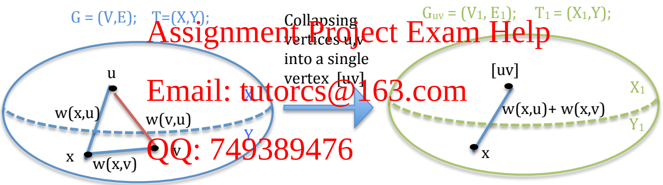
- **Claim1:** If two vertices u, v belong to the same side of a minimal cut (X, Y) then after collapsing u and v into a single vertex the capacity of the minimal cut in G_{uv} is the same as the capacity of the minimal cut in G.



- **Claim1:** If two vertices u and v belong to the same side of a minimal cut (X, Y) then after collapsing u and v into a single vertex the capacity of the minimal cut in G_{uv} is the same as the capacity of the minimal cut in G.

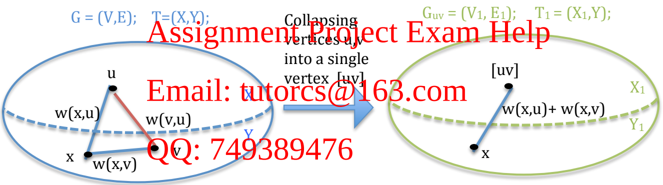


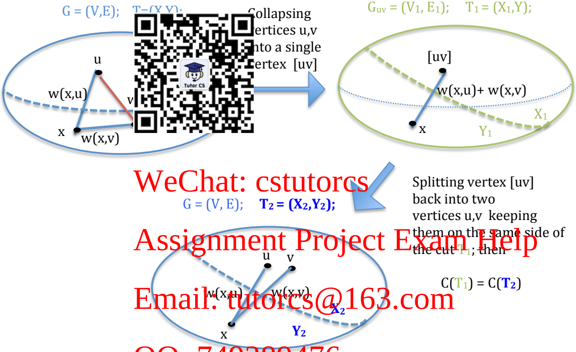
- **Claim2:** If two vertic… belong to the opposite sides of a minimal cut (X, Y) in G then after … u and v into a single vertex the capacity of the minimal cut in G_u… …r equal to the capacity of the minimal cut in G.



Karger's MinCut Algorithm

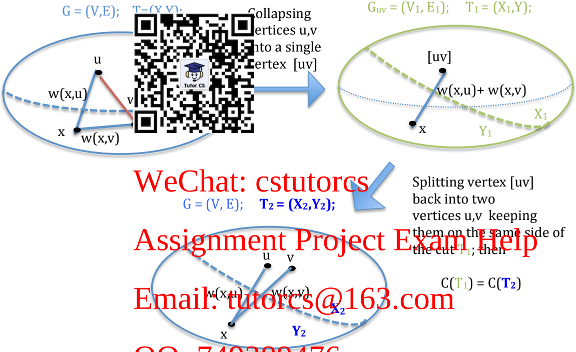
- **Claim2:** If two vertic[illegible] belong to the opposite sides of a minimal cut (X, Y) in G then after [illegible] u and v into a single vertex the capacity of the minimal cut in G_u[illegible]r equal to the capacity of the minimal cut in G.





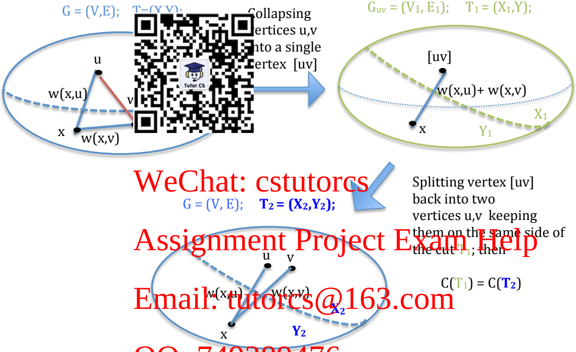
Proof:

- Let $T_1 = (X_1, Y_1)$ be a minimal cut in G_{uv} (T_1 can be completely unrelated to the minimal cut T in G).
- Split vertex $[uv]$ back into two vertices u and v but keep them on the same side of the minimal cut T_1. This produces a cut T_2 in G of the same capacity as the minimal cut T_1 in G_{uv}. Thus, the capacity of the minimal cut in G can only be smaller than the capacity of the minimal cut T_1 in G_{uv}.



Proof:

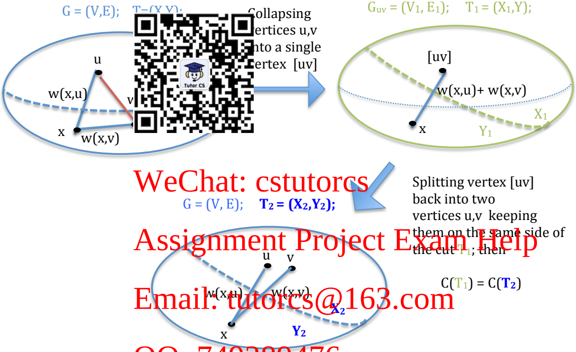
- Let $T_1 = (X_1, Y_1)$ be a minimal cut in G_{uv} (T_1 can be completely unrelated to the minimal cut T in G).
- Split vertex $[uv]$ back into two vertices u and v but keep them on the same side of the minimal cut T_1. This produces a cut T_2 in G of the same capacity as the minimal cut T_1 in G_{uv}. Thus, the capacity of the minimal cut in G can only be smaller than the capacity of the minimal cut T_1 in G_{uv}.



Proof:

- Let $T_1 = (X_1, Y_1)$ be a minimal cut in G_{uv} (T_1 can be completely unrelated to the minimal cut T in G).
- Split vertex $[uv]$ back into two vertices u and v but keep them on the same side of the minimal cut T_1. This produces a cut T_2 in G of the same capacity as the minimal cut T_1 in G_{uv}. Thus, the capacity of the minimal cut in G can only be smaller than the capacity of the minimal cut T_1 in G_{uv}.

Karger's MinCut Algorithm



Proof:

- Let $T_1 = (X_1, Y_1)$ be a minimal cut in G_{uv} (T_1 can be completely unrelated to the minimal cut T in G).
- Split vertex $[uv]$ back into two vertices u and v but keep them on the same side of the minimal cut T_1. This produces a cut T_2 in G of the same capacity as the minimal cut T_1 in G_{uv}. Thus, the capacity of the minimal cut in G can only be smaller than the capacity of the minimal cut T_1 in G_{uv}.

Karger's MinCut Algorithm - first attempt

Algorithm 1:

- Pick an edge to contract with probability proportional to the weight of that edge:

$$P(e(u,v)) = \frac{\mathrm{w}(u,v)}{\sum_{e(p,q)\in E} \mathrm{w}(p,q)}$$

- Continue until only one edge is left (we are assuming that the graph is connected).

- Take the capacity of that last edge to be the estimate of the capacity of the minimal cut in G.

Karger's MinCut Algorithm - first attempt

Algorithm 1:

- Pick an edge to contract with probability proportional to the weight of that edge:

$$P(e(u,v)) = \frac{\mathrm{w}(u,v)}{\sum_{e(p,q)\in E} \mathrm{w}(p,q)}$$

- Continue until only one edge is left (we are assuming that the graph is connected).
- Take the capacity of that last edge to be the estimate of the capacity of the minimal cut in G.

Karger's MinCut Algorithm - first attempt

Algorithm 1:

- Pick an edge to contract with probability proportional to the weight of that edge:

$$P(e(u,v)) = \frac{\mathrm{w}(u,v)}{\sum_{e(p,q)\in E} \mathrm{w}(p,q)}$$

- Continue until only one edge is left (we are assuming that the graph is connected).
- Take the capacity of that last edge to be the estimate of the capacity of the minimal cut in G.

Karger's MinCut Algorithm - first attempt

Algorithm 1:

- Pick an edge to contract with probability proportional to the weight of that edge:

$$P(e(u,v)) = \frac{\mathrm{w}(u,v)}{\sum_{e(p,q)\in E} \mathrm{w}(p,q)}$$

- Continue until only one edge is left (we are assuming that the graph is connected).

- Take the capacity of that last edge to be the estimate of the capacity of the minimal cut in G.

Karger's MinCut Algorithm - first attempt

- **Theorem 1:** Let G_{uv} the graph obtained from a graph G with n vertices by contracting an edge $e(u,v) \in E$. Then the probability that the capacity of a minimal cut in G_{uv} is [illegible] the capacity of a minimal cut in G is smaller than $2/n$:

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) < \frac{2}{n} \qquad (1)$$

- **Proof:** As we have shown, the capacity of the min cut can increase only if the vertices collapsed are on the opposite sides of every min cut in G.
- Let also $M = \{e(x,y) : x \in X, y \in Y\}$ be a minimum capacity cut in G; then

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) \leq P\Big(e(u,v) \in M\Big) \qquad (2)$$

Note that

$$P\Big(e(u,v) \in M\Big) = \frac{\sum\{\mathrm{w}(p,q) \ : \ e(p,q) \in M\}}{\sum\{\mathrm{w}(u,v) \ : \ e(u,v) \in E\}} \qquad (3)$$

Karger's MinCut Algorithm - first attempt

- **Theorem 1:** Let G_{uv} the graph obtained from a graph G with n vertices by contracting an edge $e(u, v) \in E$. Then the probability that the capacity of a minimal cut in G_{uv} is [illegible] the capacity of a minimal cut in G is smaller than $2/n$:

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) < \frac{2}{n} \qquad (1)$$

- **Proof:** As we have shown, the capacity of the min cut can increase only if the vertices collapsed are on the opposite sides of every min cut in G.

- Let also $M = \{e(x, y) : x \in X, y \in Y\}$ be a minimum capacity cut in G; then

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) \leq P\Big(e(u, v) \in M\Big) \qquad (2)$$

Note that

$$P\Big(e(u, v) \in M\Big) = \frac{\sum\{\mathrm{w}(p, q) \;:\; e(p, q) \in M\}}{\sum\{\mathrm{w}(u, v) \;:\; e(u, v) \in E\}} \qquad (3)$$

Karger's MinCut Algorithm - first attempt

- **Theorem 1:** Let G_{uv} the graph obtained from a graph G with n vertices by contracting an edge $e(u,v) \in E$. Then the probability that the capacity of a minimal cut in G_{uv} is larger than the capacity of a minimal cut in G is smaller than $2/n$:

$$P\Big(\textsc{min-cut-capacity}(G_{uv}) > \textsc{min-cut-capacity}(G)\Big) < \frac{2}{n} \tag{1}$$

- **Proof:** As we have shown, the capacity of the min cut can increase only if the vertices collapsed are on the opposite sides of every min cut in G.

- Let also $M = \{e(x,y) \ : \ x \in X, y \in Y\}$ be a minimum capacity cut in G; then

$$P\Big(\textsc{min-cut-capacity}(G_{uv}) > \textsc{min-cut-capacity}(G)\Big) \leq P\Big(e(u,v) \in M\Big) \tag{2}$$

Note that

$$P\Big(e(u,v) \in M\Big) = \frac{\sum\{\mathrm{w}(p,q) \ : \ e(p,q) \in M\}}{\sum\{\mathrm{w}(u,v) \ : \ e(u,v) \in E\}} \tag{3}$$

Karger's MinCut Algorithm - first attempt

- **Theorem 1:** Let G_{uv} the graph obtained from a graph G with n vertices by contracting an edge $e(u,v) \in E$. Then the probability that the capacity of a minimal cut in G_{uv} is [illegible] the capacity of a minimal cut in G is smaller than $2/n$:

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) < \frac{2}{n} \qquad (1)$$

- **Proof:** As we have shown, the capacity of the min cut can increase only if the vertices collapsed are on the opposite sides of every min cut in G.

- Let also $M = \{e(x,y) : x \in X, y \in Y\}$ be a minimum capacity cut in G; then

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) \leq P\Big(e(u,v) \in M\Big) \qquad (2)$$

Note that

$$P\Big(e(u,v) \in M\Big) = \frac{\sum\{\mathrm{w}(p,q)\ :\ e(p,q) \in M\}}{\sum\{\mathrm{w}(u,v)\ :\ e(u,v) \in E\}} \qquad (3)$$

Karger's MinCut Algorithm - first attempt

- **Claim:**

$$2\sum_{e\in E} \mathrm{w}(e) = \sum_{v\in V}\ \sum_{u\,:\,e(v,u)\in E} \mathrm{w}(v,u) \tag{4}$$

- **Proof:** In the sum on the right every edge is counted twice, once for each of its vertices.

- **Claim:** For every $v \in V$,

$$\sum_{u\,:\,e(v,u)\in E} \mathrm{w}(v,u) \geq \text{MIN-CUT-CAPACITY}(G) \tag{5}$$

- **Proof:** If we let $X = \{v\}$ and $Y = V\setminus\{v\}$ we get a cut $T = (X, Y)$ whose capacity must be larger or equal to the capacity of the minimal cut M.

- Since $|V| = n$, summing the inequalities (5) over all $v \in V$ and using (4) we now obtain

$$\sum_{e\in E} \mathrm{w}(e) \geq \frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G) \tag{6}$$

Karger's MinCut Algorithm - first attempt

- **Claim:**

$$2\sum_{e \in E} w(e) = \sum_{v \in V} \sum_{u\,:\,e(v,u) \in E} w(v,u) \tag{4}$$

- **Proof:** In the sum on the right every edge is counted twice, once for each of its vertices.

- **Claim:** For every $v \in V$,

$$\sum_{u\,:\,e(v,u) \in E} w(v,u) \geq \text{MIN-CUT-CAPACITY}(G) \tag{5}$$

- **Proof:** If we let $X = \{v\}$ and $Y = V \backslash \{v\}$ we get a cut $T = (X, Y)$ whose capacity must be larger or equal to the capacity of the minimal cut M.

- Since $|V| = n$, summing the inequalities (5) over all $v \in V$ and using (4) we now obtain

$$\sum_{e \in E} w(e) \geq \frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G) \tag{6}$$

Karger's MinCut Algorithm - first attempt

- **Claim:**

$$2\sum_{e\in E} w(e) = \sum_{v\in V} \sum_{u\,:\,e(v,u)\in E} w(v,u) \tag{4}$$

- **Proof:** In the sum on the right every edge is counted twice, once for each of its vertices.

- **Claim:** For every $v \in V$,

$$\sum_{u\,:\,e(v,u)\in E} w(v,u) \geq \text{MIN-CUT-CAPACITY}(G) \tag{5}$$

- **Proof:** If we let $X = \{v\}$ and $Y = V\setminus\{v\}$ we get a cut $T = (X, Y)$ whose capacity must be larger or equal to the capacity of the minimal cut M.

- Since $|V| = n$, summing the inequalities (5) over all $v \in V$ and using (4) we now obtain

$$\sum_{e\in E} w(e) \geq \frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G) \tag{6}$$

Karger's MinCut Algorithm - first attempt

- **Claim:**

$$2\sum_{e \in E} w(e) = \sum_{v \in V} \sum_{u\,:\,e(v,u) \in E} w(v,u) \tag{4}$$

- **Proof:** In the sum on the right every edge is counted twice, once for each of its vertices.

- **Claim:** For every $v \in V$,

$$\sum_{u\,:\,e(v,u) \in E} w(v,u) \geq \text{MIN-CUT-CAPACITY}(G) \tag{5}$$

- **Proof:** If we let $X = \{v\}$ and $Y = V \backslash \{v\}$ we get a cut $T = (X, Y)$ whose capacity must be larger or equal to the capacity of the minimal cut M.

- Since $|V| = n$, summing the inequalities (5) over all $v \in V$ and using (4) we now obtain

$$\sum_{e \in E} w(e) \geq \frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G) \tag{6}$$

Karger's MinCut Algorithm - first attempt

- **Claim:**

$$2\sum_{e\in E} w(e) = \sum_{v\in V} \sum_{u\,:\,e(v,u)\in E} w(v,u) \qquad (4)$$

- **Proof:** In the sum on the right every edge is counted twice, once for each of its vertices.

- **Claim:** For every $v \in V$,

$$\sum_{u\,:\,e(v,u)\in E} w(v,u) \geq \text{MIN-CUT-CAPACITY}(G) \qquad (5)$$

- **Proof:** If we let $X = \{v\}$ and $Y = V\backslash\{v\}$ we get a cut $T = (X, Y)$ whose capacity must be larger or equal to the capacity of the minimal cut M.

- Since $|V| = n$, summing the inequalities (5) over all $v \in V$ and using (4) we now obtain

$$\sum_{e\in E} w(e) \geq \frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G) \qquad (6)$$

- From (3) and (6) we n

$$\begin{aligned} P\Big(e(u,v) \in M\Big) &= \frac{\sum\{\mathrm{w}(p,q)\ :\ e(p,q) \in M\}}{\sum\{\mathrm{w}(u,v)\ :\ e(u,v) \in E\}} \\ &\leq \frac{\text{MIN-CUT-CAPACITY}(G)}{\frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G)} \\ &= \frac{2}{n} \end{aligned}$$

- Thus, we obtain

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) \leq P\Big(e(u,v) \in M\Big) \leq \frac{2}{n} \quad (7)$$

- From (3) and (6) we n

$$P\Big(e(u,v) \in M\Big) = \frac{\sum\{\mathrm{w}(p,q)\ :\ e(p,q) \in M\}}{\sum\{\mathrm{w}(u,v)\ :\ e(u,v) \in E\}}$$
$$\leq \frac{\text{MIN-CUT-CAPACITY}(G)}{\frac{n}{2} \cdot \text{MIN-CUT-CAPACITY}(G)}$$
$$= \frac{2}{n}$$

- Thus, we obtain

$$P\Big(\text{MIN-CUT-CAPACITY}(G_{uv}) > \text{MIN-CUT-CAPACITY}(G)\Big) \leq P\Big(e(u,v) \in M\Big) \leq \frac{2}{n} \quad (7)$$

- **Theorem 2:** If we run edge contraction procedure until we get a single edge, then the probability π that the capacity of that final edge is equal to the capacity of a minimal cut in G is $\Omega\left(\frac{1}{n^2}\right)$.

- **Proof:** Let G_i for $0 \leq i \leq n-2$ be the sequence of graphs obtained by successive edge contractions starting from $G_0 = G$. The probability π that the capacity of the final edge is equal to the capacity of a minimal cut in G is greater or equal to the probability that we never contracted an edge belonging to M.

- Thus, (7) implies

$$\begin{aligned}
\pi &= P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n-2})\Big) \\
&= \prod_{i=1}^{n-2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big) \\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{3}\right) \\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdot\ldots\cdot\frac{4}{6}\cdot\frac{3}{5}\cdot\frac{2}{4}\cdot\frac{1}{3} \\
&= \frac{2}{n(n-1)}, \quad \text{which implies the claim of the theorem.}
\end{aligned}$$

Karger's MinCut Algorithm - first attempt

- **Theorem 2:** If we run edge contraction procedure until we get a single edge, then the probability π that the capacity of that final edge is equal to the capacity of a minimal cut in G is $\Omega\left(\frac{1}{n^2}\right)$.
- **Proof:** Let G_i for $0 \leq i \leq n-2$ be the sequence of graphs obtained by successive edge contractions starting from $G_0 = G$. The probability π that the capacity of the final edge is equal to the capacity of a minimal cut in G is greater or equal to the probability that we never contracted an edge belonging to M.

- Thus, (7) implies

$$
\begin{aligned}
\pi &= P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n-2})\Big) \\
&= \prod_{i=1}^{n-2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big) \\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{3}\right) \\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdot\ldots\cdot\frac{4}{6}\cdot\frac{3}{5}\cdot\frac{2}{4}\cdot\frac{1}{3} \\
&= \frac{2}{n(n-1)}, \quad \text{which implies the claim of the theorem.}
\end{aligned}
$$

- **Theorem 2:** If we run edge contraction procedure until we get a single edge, then the probability π that the capacity of that final edge is equal to the capacity of a minimal [illegible] $\Omega\left(\frac{1}{n^2}\right)$.

- **Proof:** Let G_i for $0 \leq$ [illegible] be the sequence of graphs obtained by successive edge contrac[illegible]ing from $G_0 = G$. The probability π that the capacity of the final ed[illegible] to the capacity of a minimal cut in G is greater or equal to the probability that we never contracted an edge belonging to M.

- Thus, (7) implies

$$
\begin{aligned}
\pi &= P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n-2})\Big) \\
&= \prod_{i=1}^{n-2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big) \\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\ldots\left(1-\frac{2}{3}\right) \\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdot\ldots\cdot\frac{4}{6}\cdot\frac{3}{5}\cdot\frac{2}{4}\cdot\frac{1}{3} \\
&= \frac{2}{n(n-1)}, \qquad \text{which implies the claim of the theorem.}
\end{aligned}
$$

Karger's MinCut Algorithm - refinement

- However, $\pi = \Omega\left(\frac{1}{n^2}\right)$ is a very small probability for large n; somehow we have to boost it.

- Let us run our contraction algorithm only until the number of vertices is $\lfloor \frac{n}{2} \rfloor$.

- Then such an algorithm runs in time $O(n^2)$ and we have

$$
\begin{aligned}
&P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n/2})\Big)\\
&= \prod_{i=1}^{n/2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big)\\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n/2+1}\right)\cdot\left(1-\frac{2}{n/2}\right)\\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdots\frac{n/2}{n/2+2}\cdot\frac{n/2-1}{n/2+1}\cdot\frac{n/2-2}{n/2}\\
&= \frac{(n/2-1)(n/2-2)}{n(n-1)}\\
&\approx \frac{1}{4}
\end{aligned}
$$

Karger's MinCut Algorithm - refinement

- However, $\pi = \Omega\left(\frac{1}{n^2}\right)$ is a very small probability for large n; somehow we have to boost it.

- Let us run our contrac[illegible]hm only until the number of vertices is $\lfloor \frac{n}{2} \rfloor$.

- Then such an algorith[illegible]ime $O(n^2)$ and we have

$$\begin{aligned}
&P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n/2})\Big) \\
&= \prod_{i=1}^{n/2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big) \\
&\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right)\left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{n/2+1}\right) \cdot \left(1 - \frac{2}{n/2}\right) \\
&= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \frac{n-5}{n-3} \cdots \frac{n/2}{n/2+2} \cdot \frac{n/2-1}{n/2+1} \cdot \frac{n/2-2}{n/2} \\
&= \frac{(n/2-1)(n/2-2)}{n(n-1)} \\
&\approx \frac{1}{4}
\end{aligned}$$

Karger's MinCut Algorithm - refinement

- However, $\pi = \Omega\left(\frac{1}{n^2}\right)$ is a very small probability for large n; somehow we have to boost it.
- Let us run our contraction algorithm only until the number of vertices is $\lfloor \frac{n}{2} \rfloor$.
- Then such an algorithm runs in time $O(n^2)$ and we have

$$
\begin{aligned}
&P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n/2})\Big)\\
&= \prod_{i=1}^{n/2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big)\\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n/2+1}\right)\cdot\left(1-\frac{2}{n/2}\right)\\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdots\frac{n/2}{n/2+2}\cdot\frac{n/2-1}{n/2+1}\cdot\frac{n/2-2}{n/2}\\
&= \frac{(n/2-1)(n/2-2)}{n(n-1)}\\
&\approx \frac{1}{4}
\end{aligned}
$$

Karger's MinCut Algorithm - refinement

- However, $\pi = \Omega\left(\frac{1}{n^2}\right)$ is a very small probability for large n; somehow we have to boost it.
- Let us run our contraction algorithm only until the number of vertices is $\lfloor \frac{n}{2} \rfloor$.
- Then such an algorithm runs in time $O(n^2)$ and we have

$$
\begin{aligned}
&P\Big(\text{MIN-CUT-CAPACITY}(G) = \text{MIN-CUT-CAPACITY}(G_{n/2})\Big)\\
&= \prod_{i=1}^{n/2} P\Big(\text{MIN-CUT-CAPACITY}(G_i) = \text{MIN-CUT-CAPACITY}(G_{i-1})\Big)\\
&\geq \left(1-\frac{2}{n}\right)\left(1-\frac{2}{n-1}\right)\left(1-\frac{2}{n-2}\right)\cdots\left(1-\frac{2}{n/2+1}\right)\cdot\left(1-\frac{2}{n/2}\right)\\
&= \frac{n-2}{n}\cdot\frac{n-3}{n-1}\cdot\frac{n-4}{n-2}\cdot\frac{n-5}{n-3}\cdots\frac{n/2}{n/2+2}\cdot\frac{n/2-1}{n/2+1}\cdot\frac{n/2-2}{n/2}\\
&= \frac{(n/2-1)(n/2-2)}{n(n-1)}\\
&\approx \frac{1}{4}
\end{aligned}
$$

Karger's MinCut Algorithm - refinement

- This shows that the probability of not picking an edge which belongs to a min cut M is fairly large after $n/2$ many contractions, but drops fast afterwards. This suggests the following algorithm:

4-CONTRACT(G)

$G_0 = (V_0, E_0) \leftarrow G = (V, E)$
while $|V_0| > 2$
  for $i = 1$ to 4
    run the randomised edge contraction algorithm on G_0 until you get a graph $G_i = (V_i, E_i)$ with $|V_i| = |V_0|/2$ many vertices;
  end for
  4-CONTRACT(G_1)
  4-CONTRACT(G_2)
  4-CONTRACT(G_3)
  4-CONTRACT(G_4)
end while
return the smallest capacity among the capacities of all thus produced single edges.

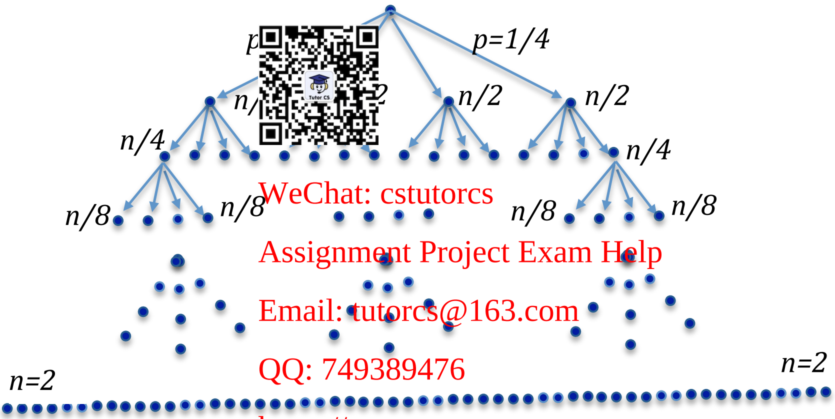
Karger's MinCut Algorithm - refinement

- This shows that the probability of not picking an edge which belongs to a min cut M is fairly large after $n/2$ many contractions, but drops fast afterwards. This suggests the following algorithm:

4-Contract(G)

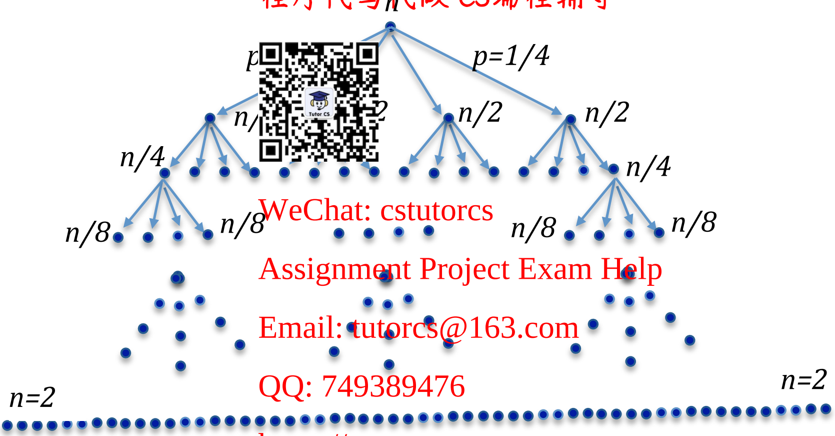
$G_0 = (V_0, E_0) \leftarrow (V, E)$
while $|V_0| > 2$
  for $i = 1$ to 4
    run the randomised edge contraction algorithm on G_0 until you get a graph $G_i = (V_i, E_i)$ with $|V_i| = |V_0|/2$ many vertices;
  end for
  4-Contract(G_1)
  4-Contract(G_2)
  4-Contract(G_3)
  4-Contract(G_4)
end while
return the smallest capacity among the capacities of all thus produced single edges.

Karger's MinCut Algorithm - refinement



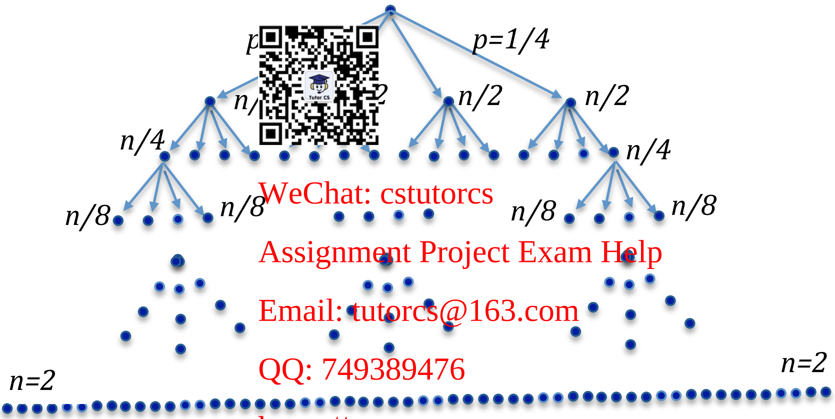
- Run time: $T(n) = 4T(n/2) + O(n^2)$
- By the Master Theorem (case 2), $T(n) = O(n^2 \log n)$.

Karger's MinCut Algorithm - refinement



- Run time: $T(n) = 4T(n/2) + O(n^2)$
- By the Master Theorem (case 2), $T(n) = O(n^2 \log n)$.

Karger's MinCut Algorithm - refinement



- Run time: $T(n) = 4T(n/2) + O(n^2)$
- By the Master Theorem (case 2), $T(n) = O(n^2 \log n)$.

Karger's MinCut Algorithm - refinement

- What is the probability that at least one of the edges will have the capacity of the min cut of G, and hence the algorithm will produce the correct value of MIN-CUT-CAPACITY(G)?

$$
\begin{aligned}
P(\text{success for a graph of size } n) &= 1 - P(\text{failure on all 4 branches}) \\
&= 1 - P(\text{failure on one branch})^4 = 1 - (1 - P(\text{success on one branch}))^4 \\
&= 1 - \left(1 - \frac{1}{4} P\left(\text{success for a graph of size } \frac{n}{2}\right)\right)^4
\end{aligned}
$$

Let $p(n) = P(\text{success for a graph of size } n)$; then

$$
p(n) = 1 - \left(1 - \frac{1}{4} p\left(\frac{n}{2}\right)\right)^4
$$

Karger's MinCut Algorithm - refinement

- What is the probability that at least one of the edges will have the capacity of the min cut of G, and thus the algorithm will produce the correct value of $\textsc{min-cut-capacity}(G)$?

$$
\begin{aligned}
&P(\text{success for a graph of size } n) = 1 - P(\text{failure on all 4 branches})\\
&= 1 - P(\text{failure on one branch})^4 = 1 - (1 - P(\text{success on one branch}))^4\\
&= 1 - \left(1 - \frac{1}{4}P\left(\text{success for a graph of size } \frac{n}{2}\right)\right)^4
\end{aligned}
$$

Let $p(n) = P(\text{success for a graph of size } n)$; then

$$p(n) = 1 - \left(1 - \frac{1}{4}p\left(\frac{n}{2}\right)\right)^4$$

Karger's MinCut Algorithm - refinement

- What is the probabilit[illegible]ast one of the edges will have the capacity of the min cut of G, and [illegible]he algorithm will produce the correct value of MIN-CUT-CAPACITY(G)?

$$P(\text{success for a graph of size } n) = 1 - P(\text{failure on all 4 branches})$$

$$= 1 - P(\text{failure on one branch})^4 = 1 - (1 - P(\text{success on one branch}))^4$$

$$= 1 - \left(1 - \frac{1}{4} P\left(\text{success for a graph of size } \frac{n}{2}\right)\right)^4$$

Let $p(n) = P(\text{success for a graph of size } n)$; then

$$p(n) = 1 - \left(1 - \frac{1}{4} p\left(\frac{n}{2}\right)\right)^4$$

Karger's MinCut Algorithm - refinement

- Note that

$$
\begin{aligned}
p(n) &= 1 - \left(1 - \frac{1}{4} p\left(\frac{n}{2}\right)\right)^4 \\
&= p\left(\frac{n}{2}\right) - \frac{3}{8} p\left(\frac{n}{2}\right)^2 + \frac{1}{16} p\left(\frac{n}{2}\right)^3 - \frac{1}{256} p\left(\frac{n}{2}\right)^4 \\
&> p\left(\frac{n}{2}\right) - \frac{3}{8} p\left(\frac{n}{2}\right)^2 \qquad (8)
\end{aligned}
$$

- We now use an induction of type

$$\phi(1) \& \forall n (\phi(\lfloor n/2 \rfloor) \to \phi(n)) \to \forall n \phi(n)$$

and prove that the assumption $p(n/2) > \frac{1}{\log(n/2)}$ implies $p(n) > \frac{1}{\log n}$.

- Note that

$$
\begin{aligned}
p(n) &= 1 - \left(1 - \frac{1}{4}p\left(\frac{n}{2}\right)\right)^4 \\
&= p\left(\frac{n}{2}\right) - \frac{3}{8}p\left(\frac{n}{2}\right)^2 + \frac{1}{16}p\left(\frac{n}{2}\right)^3 - \frac{1}{256}p\left(\frac{n}{2}\right)^4 \\
&> p\left(\frac{n}{2}\right) - \frac{3}{8}p\left(\frac{n}{2}\right)^2 \qquad (8)
\end{aligned}
$$

- We now use an induction of type

$$\phi(1)\&\forall n(\phi(\lfloor n/2 \rfloor) \to \phi(n)) \to \forall n \phi(n)$$

and prove that the assumption $p(n/2) > \frac{1}{\log(n/2)}$ implies $p(n) > \frac{1}{\log n}$.

- Using the fact that function $f(x) = x - \frac{3}{8} \cdot x^2$ is monotonically increasing on $[0,1]$, we obtain from the induction hypothesis and (8)

$$p(n) \geq p\left(\frac{n}{2}\right) - \frac{3}{8}\, p\left(\frac{n}{2}\right)^2 > \frac{1}{\log \frac{n}{2}} - \frac{3}{8} \frac{1}{\left(\log \frac{n}{2}\right)^2}$$

$$= \frac{1}{\log n - 1} - \frac{3}{8} \frac{1}{(\log n - 1)^2}$$

- We now use the fact that

$$\frac{1}{x-1} - \frac{3}{8(x-1)^2} \geq \frac{1}{x}$$

for all $x \geq 8/5$ to finally obtain $p(n) > \frac{1}{\log n}$ which proves the induction hypothesis and we conclude that $p(n) > \frac{1}{\log n}$ for all $n \geq 4$.

Karger's MinCut Algorithm - refinement

- Using the fact that function $f(x) = x - \frac{3}{8} \cdot x^2$ is monotonically increasing on $[0, 1]$, we obtain from the induction hypothesis and (8)

$$p(n) > p\left(\frac{n}{2}\right) - \frac{3}{8}\, p\left(\frac{n}{2}\right)^2 > \frac{1}{\log \frac{n}{2}} - \frac{3}{8} \frac{1}{\left(\log \frac{n}{2}\right)^2}$$

$$= \frac{1}{\log n - 1} - \frac{3}{8} \frac{1}{(\log n - 1)^2}$$

- We now use the fact that

$$\frac{1}{x-1} - \frac{3}{8(x-1)^2} \geq \frac{1}{x}$$

for all $x \geq 8/5$ to finally obtain $p(n) > \frac{1}{\log n}$, which proves the induction hypothesis and we conclude that $p(n) > \frac{1}{\log n}$ for all $n \geq 4$.

Karger's MinCut Algorithm - refinement

- Thus, if we run our 4-CONTRACT(G) algorithm $(\log n)^2$ many times and take the smallest capacity estimate produced, probability π that this estimate will be correct is

$$\pi \approx 1 - \left(1 - \frac{1}{\log n}\right)^{(\log n)^2}$$

- We now use the fact that for all reasonably large k we have $(1 - 1/k)^k \approx e^{-1}$

- Thus,

$$\pi \approx 1 - e^{-\log n} = 1 - 1/n$$

- So, for large n (which is when other algorithms for Min Cut are slow) we get the correct value with probability $1 - 1/n$, i.e., almost certainly!

- To run our algorithm $(\log n)^2$ times it takes the total number of steps of only

$$O(n^2 \log n \times (\log n)^2) = O(n^2 (\log n)^3) << O(n^4).$$

- Thus, our randomised algorithm runs much faster than the deterministic algorithm which runs in time $O(n^4)$ and yet it succeeds with a high probability!

Karger's MinCut Algorithm - refinement

- Thus, if we run our 4-CONTRACT(G) algorithm $(\log n)^2$ many times and take the smallest capacity estimate produced, probability π that this estimate will be correct is

$$\left(1 - \frac{1}{\log n}\right)^{(\log n)^2}$$

- We now use the fact that for all reasonably large k we have $(1 - 1/k)^k \approx e^{-1}$

- Thus,

$$\pi \approx 1 - e^{-\log n} = 1 - 1/n$$

- So, for large n (which is when other algorithms for Min Cut are slow) we get the correct value with probability $1 - 1/n$, i.e., almost certainly!

- To run our algorithm $(\log n)^2$ times it takes the total number of steps of only

$$O(n^2 \log n \times (\log n)^2) = O(n^2 (\log n)^3) << O(n^4).$$

- Thus, our randomised algorithm runs much faster than the deterministic algorithm which runs in time $O(n^4)$ and yet it succeeds with a high probability!

Karger's MinCut Algorithm - refinement

- Thus, if we run our 4-CONTRACT(G) algorithm $(\log n)^2$ many times and take the smallest capacity estimate produced, probability π that this estimate will be correct is

$$\left(1 - \frac{1}{\log n}\right)^{(\log n)^2}$$

- We now use the fact that for all reasonably large k we have $(1 - 1/k)^k \approx e^{-1}$

- Thus,

$$\pi \approx 1 - e^{-\log n} = 1 - 1/n$$

- So, for large n (which is when other algorithms for Min Cut are slow) we get the correct value with probability $1 - 1/n$, i.e., almost certainly!
- To run our algorithm $(\log n)^2$ times it takes the total number of steps of only

$$O(n^2 \log n \times (\log n)^2) = O(n^2 (\log n)^3) << O(n^4).$$

- Thus, our randomised algorithm runs much faster than the deterministic algorithm which runs in time $O(n^4)$ and yet it succeeds with a high probability!

Karger's MinCut Algorithm - refinement

- Thus, if we run our 4-CONTRACT(G) algorithm $(\log n)^2$ many times and take the smallest capacity estimate produced, probability π that this estimate will be correct is

$$\left(1 - \frac{1}{\log n}\right)^{(\log n)^2}$$

- We now use the fact that for all reasonably large k we have $(1 - 1/k)^k \approx e^{-1}$

- Thus,

$$\pi \approx 1 - e^{-\log n} = 1 - 1/n$$

- So, for large n (which is when other algorithms for Min Cut are slow) we get the correct value with probability $1 - 1/n$, i.e., almost certainly!

- To run our algorithm $(\log n)^2$ times it takes the total number of steps of only

$$O(n^2 \log n \times (\log n)^2) = O(n^2 (\log n)^3) << O(n^4).$$

- Thus, our randomised algorithm runs much faster than the deterministic algorithm which runs in time $O(n^4)$ and yet it succeeds with a high probability!

Karger's MinCut Algorithm - refinement

- Thus, if we run our 4-CONTRACT(G) algorithm $(\log n)^2$ many times and take the smallest capacity estimate produced, probability π that this estimate will be correct is

$$\left(1-\frac{1}{\log n}\right)^{(\log n)^2}$$

- We now use the fact that for all reasonably large k we have $(1-1/k)^k \approx e^{-1}$

- Thus,

$$\pi \approx 1 - e^{-\log n} = 1 - 1/n$$

- So, for large n (which is when other algorithms for Min Cut are slow) we get the correct value with probability $1-1/n$, i.e., almost certainly!

- To run our algorithm $(\log n)^2$ times it takes the total number of steps of only

$$O(n^2 \log n \times (\log n)^2) = O(n^2(\log n)^3) << O(n^4).$$

- Thus, our randomised algorithm runs much faster than the deterministic algorithm which runs in time $O(n^4)$ and yet it succeeds with a high probability!