Algorithm Theory, Tutorial 7

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

January 2019

Ex 1

Let us consider the following modified version of the contraction algorithm presented in the lecture. Instead of choosing an edge uniformly at random and merging its endpoints, in each step the modified algorithm chooses a pair of nodes in graph G uniformly at random and merges the two nodes into a single node.

a) Give an example graph of size at least n where the above algorithm does not work well, that is, where the probability of finding a minimum cut is exponentially small in n.
b) Prove that property for the example you gave in (a), i.e., show that the modified contraction algorithm has probability of finding a minimum cut at most c^n for some constant $c < 1$.

Ex 1a, solution

- We now pick two nodes instead of an edge

Ex 1a, solution

- We now pick two nodes instead of an edge
- New: possibility that we choose unconnected nodes.

Ex 1a, solution

- We now pick two nodes instead of an edge
- New: possibility that we choose unconnected nodes.
- Observation: If the minimum cut after the merging is bigger than before, we have a problem (in both versions)

Ex 1a, solution

- We now pick two nodes instead of an edge
- New: possibility that we choose unconnected nodes.
- Observation: If the minimum cut after the merging is bigger than before, we have a problem (in both versions)
- We are thus looking for a graph where the probability of choosing a pair of nodes that make the min-cut bigger when merging is very high.

Ex 1a, solution

- We now pick two nodes instead of an edge
- New: possibility that we choose unconnected nodes.
- Observation: If the minimum cut after the merging is bigger than before, we have a problem (in both versions)
- We are thus looking for a graph where the probability of choosing a pair of nodes that make the min-cut bigger when merging is very high.
- Example: For simplicity we assume that n is dividable by four. Create a graph with two cliques of size $n/2$ that are connected by one edge in the middle.

Ex 1a, solution

- We now pick two nodes instead of an edge
- New: possibility that we choose unconnected nodes.
- Observation: If the minimum cut after the merging is bigger than before, we have a problem (in both versions)
- We are thus looking for a graph where the probability of choosing a pair of nodes that make the min-cut bigger when merging is very high.
- Example: For simplicity we assume that n is dividable by four. Create a graph with two cliques of size $n/2$ that are connected by one edge in the middle.
- Draw example.

1a, Example pictures

Which pairs of nodes are bad for us (destroy our min-cut?)

Solution 1b

- Let K_1, K_2 be our two cliques

Solution 1b

- Let K_1, K_2 be our two cliques
- We can never obtain the min-cut if we contract a pair of nodes u, v with $u \in K_1$ and $v \in K_2$.

Solution 1b

- Let K_1, K_2 be our two cliques
- We can never obtain the min-cut if we contract a pair of nodes u, v with $u \in K_1$ and $v \in K_2$.
- In order to obtain the min-cut we need to be lucky enough that we *always* randomly choose a pair u, v that are in the same clique.

Solution 1b

- Let K_1, K_2 be our two cliques
- We can never obtain the min-cut if we contract a pair of nodes u, v with $u \in K_1$ and $v \in K_2$.
- In order to obtain the min-cut we need to be lucky enough that we *always* randomly choose a pair u, v that are in the same clique.
- We analyze the chance that this happens in the first $T = n/4$ contractions of the modified algorithm.

Solution 1b

- Let K_1, K_2 be our two cliques
- We can never obtain the min-cut if we contract a pair of nodes u, v with $u \in K_1$ and $v \in K_2$.
- In order to obtain the min-cut we need to be lucky enough that we *always* randomly choose a pair u, v that are in the same clique.
- We analyze the chance that this happens in the first $T = n/4$ contractions of the modified algorithm.
- Let n_1, n_2 be the number of remaining nodes in K_1, K_2 after T contractions. Since we contract at most $n/4$ pairs, we have $n_1, n_2 \geq n/4$.

Solution 1b Continued

- The probability to draw a remaining node uniformly at random from a *fixed* clique in any of the first T contractions is therefore at least $\frac{n/4}{n} = 1/4$

Solution 1b Continued

- The probability to draw a remaining node uniformly at random from a *fixed* clique in any of the first T contractions is therefore at least $\frac{n/4}{n} = 1/4$
- Let $\mathcal{E}_i$ be the event that the algorithm randomly chooses *two* nodes u, v from the *same* clique in step $i \leq T$. We have

$$\mathbb{P}(\mathcal{E}_i) = 1 - \mathbb{P}(\overline{\mathcal{E}_i}) = 1 - \mathbb{P}(\text{“}u, v \text{ from different cliques''}) \leq 1 - 2 \cdot \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{8}$$

Solution 1b Continued

- The probability to draw a remaining node uniformly at random from a *fixed* clique in any of the first T contractions is therefore at least $\frac{n/4}{n} = 1/4$
- Let $\mathcal{E}_i$ be the event that the algorithm randomly chooses *two* nodes u, v from the *same* clique in step $i \leq T$. We have

$$\mathbb{P}(\mathcal{E}_i) = 1 - \mathbb{P}(\overline{\mathcal{E}_i}) = 1 - \mathbb{P}(\text{“}u, v \text{ from different cliques”}) \leq 1 - 2 \cdot \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{8}$$

- Let $\mathcal{E} = \bigcap_{i=1}^{n/4} \mathcal{E}_i$ be the event that we draw pairs of nodes from the same cliques in all of the first $T = n/4$ contractions.

Solution 1b Continued

- The probability to draw a remaining node uniformly at random from a *fixed* clique in any of the first T contractions is therefore at least $\frac{n/4}{n} = 1/4$
- Let $\mathcal{E}_i$ be the event that the algorithm randomly chooses *two* nodes u, v from the *same* clique in step $i \leq T$. We have

$$\mathbb{P}(\mathcal{E}_i) = 1 - \mathbb{P}(\overline{\mathcal{E}_i}) = 1 - \mathbb{P}(\text{“}u, v \text{ from different cliques”}) \leq 1 - 2 \cdot \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{8}$$

- Let $\mathcal{E} = \bigcap_{i=1}^{n/4} \mathcal{E}_i$ be the event that we draw pairs of nodes from the same cliques in all of the first $T = n/4$ contractions.
- This is a necessary condition for finding the min-cut.

Solution 1b Continued

- The probability to draw a remaining node uniformly at random from a *fixed* clique in any of the first T contractions is therefore at least $\frac{n/4}{n} = 1/4$
- Let $\mathcal{E}_i$ be the event that the algorithm randomly chooses *two* nodes u, v from the *same* clique in step $i \leq T$. We have

$$\mathbb{P}(\mathcal{E}_i) = 1 - \mathbb{P}(\overline{\mathcal{E}_i}) = 1 - \mathbb{P}(\text{“}u, v \text{ from different cliques”}) \leq 1 - 2 \cdot \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{8}$$

- Let $\mathcal{E} = \bigcap_{i=1}^{n/4} \mathcal{E}_i$ be the event that we draw pairs of nodes from the same cliques in all of the first $T = n/4$ contractions.
- This is a necessary condition for finding the min-cut.
- Since our bounds $\mathbb{P}(\mathcal{E}_i) \leq 7/8$ for the probabilities of the events $\mathcal{E}_i$ are independent from one another, we have

$$\mathbb{P}(\mathcal{E}) = \mathbb{P}\Big(\bigcap_{i=1}^{n/4} \mathcal{E}_i\Big) \leq (7/8)^{n/4} = \left(\sqrt[4]{7/8}\right)^n = c^n, \text{ where } c := \sqrt[4]{7/8} < 1.$$

Metric TSP with Small Edge Weights

Consider the family of complete, weighted, undirected graphs $G = (V, E, w)$ in which all edges have weight either 1 or 2.

Remark: TSP is the Traveling Salesperson Problem. The goal is to find a tour, i.e., a permutation $v_1, \ldots, v_n$ of nodes, that minimizes the total weight of edges on that tour $w(v_1, v_n) + \sum_{1=1}^{n} w(v_i, v_{i+1})$.

(a) Assume you have a subroutine that computes a *minimum* 2-matching for the above family of graphs in polynomial time. Describe an *efficient* algorithm that computes a 4/3-approximation for the TSP problem for graphs of this family.
Remark: A 2-matching is a subset $M \subseteq E$, so that every $v \in V$ is incident to exactly 2 edges in M.

(b) Prove that your algorithm computes a 4/3-approximation of TSP on the this family of graphs.

2-matchings

- 2-matchings are basically decompositions of the graph into cycles. (draw example).

2-matchings

- 2-matchings are basically decompositions of the graph into cycles. (draw example).
- All cycles have length at least three, thus we have at most $n/3$ cycles.

2-matchings

- 2-matchings are basically decompositions of the graph into cycles. (draw example).
- All cycles have length at least three, thus we have at most $n/3$ cycles.
- The minimum TSP is also a 2-matching.

2-matchings

- 2-matchings are basically decompositions of the graph into cycles. (draw example).
- All cycles have length at least three, thus we have at most $n/3$ cycles.
- The minimum TSP is also a 2-matching.
- $\Rightarrow$ Minimum T2-matching is lower bound for minimum TSP.

2-matchings

- 2-matchings are basically decompositions of the graph into cycles. (draw example).
- All cycles have length at least three, thus we have at most $n/3$ cycles.
- The minimum TSP is also a 2-matching.
- $\Rightarrow$ Minimum T2-matching is lower bound for minimum TSP.
- Idea for Algorithm: Calculate minimum 2-matching and then merge the cycles to a solution of the min tsp.

Algorithm

- Calculate the min 2-matching.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.
- Connect the chains to form one big cycle.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.
- Connect the chains to form one big cycle.
- We add at most 1 unit of weight per cycle/chain (worst case: only remove edges with weight 1 and add edges with weight two)

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.
- Connect the chains to form one big cycle.
- We add at most 1 unit of weight per cycle/chain (worst case: only remove edges with weight 1 and add edges with weight two)
- Let S be the weight or our Solution, M the size of our min 2-matching and O the optimal solution.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.
- Connect the chains to form one big cycle.
- We add at most 1 unit of weight per cycle/chain (worst case: only remove edges with weight 1 and add edges with weight two)
- Let S be the weight or our Solution, M the size of our min 2-matching and O the optimal solution.
- $O \geq n$ is obvious in the given graph.

Algorithm

- Calculate the min 2-matching.
- We now have at most $n/3$ cycles.
- Remove one edge from each cycle. We now have at most $n/3$ disjoint chains that still contain all nodes.
- Connect the chains to form one big cycle.
- We add at most 1 unit of weight per cycle/chain (worst case: only remove edges with weight 1 and add edges with weight two)
- Let S be the weight or our Solution, M the size of our min 2-matching and O the optimal solution.
- $O \geq n$ is obvious in the given graph.
- We compute

$$S \leq M + \frac{n}{3} \leq O + \frac{n}{3} \leq \frac{4}{3}O$$

Ex 3

Consider an undirected, unweighted Graph $G = (V, E)$ with n nodes. We are given a set $\mathcal{P}$ of p *simple* paths in G, where each path has exactly ℓ nodes. We say a path $P \in \mathcal{P}$ is *covered* by a set Q of nodes if P has a node in Q. Then, the goal is to find a set $Q \subseteq V$ of nodes with minimum cardinality such that *every* path in $\mathcal{P}$ is covered by Q.

Now consider the following simple greedy algorithm: It starts with $Q = \emptyset$. As long as there is a path not covered by Q, the node that covers the most uncovered paths is added to Q.

3a

- Argue why the above greedy algorithm provides a $(1 + \ln p)$-approximation of an minimal solution.

3a

- Argue why the above greedy algorithm provides a $(1+\ln p)$-approximation of an minimal solution.
- Reduce this problem to the minimum set cover problem (lecture).

3a

- Argue why the above greedy algorithm provides a $(1+\ln p)$-approximation of an minimal solution.
- Reduce this problem to the minimum set cover problem (lecture).
- We need to cover the set of all paths and each node covers a subset of paths.

3a

- Argue why the above greedy algorithm provides a $(1+\ln p)$-approximation of an minimal solution.
- Reduce this problem to the minimum set cover problem (lecture).
- We need to cover the set of all paths and each node covers a subset of paths.
- Let the set of paths $\mathcal{P}$ be the base set, and let $\mathcal{S} = \{S_1, \ldots, S_n\}$ where S_j contains all paths that node j covers.

3a

- Argue why the above greedy algorithm provides a $(1+\ln p)$-approximation of an minimal solution.
- Reduce this problem to the minimum set cover problem (lecture).
- We need to cover the set of all paths and each node covers a subset of paths.
- Let the set of paths $\mathcal{P}$ be the base set, and let $\mathcal{S} = \{S_1, \ldots, S_n\}$ where S_j contains all paths that node j covers.
- Result from lecture: Greedy algorithm has approximation ratio of $1 + \ln s$ where s is the maximum cardinality of any S_j.

3a

- Argue why the above greedy algorithm provides a $(1+\ln p)$-approximation of an minimal solution.
- Reduce this problem to the minimum set cover problem (lecture).
- We need to cover the set of all paths and each node covers a subset of paths.
- Let the set of paths $\mathcal{P}$ be the base set, and let $\mathcal{S} = \{S_1, \ldots, S_n\}$ where S_j contains all paths that node j covers.
- Result from lecture: Greedy algorithm has approximation ratio of $1 + \ln s$ where s is the maximum cardinality of any S_j.
- Since $s \leq p$ the claim follows.

3b

Show that after selecting i nodes, there are at most $p(1-\frac{\ell}{n})^i$ uncovered paths left.

Proof by induction

- Let u_k the number of uncovered paths after k steps.

Proof by induction

- Let u_k the number of uncovered paths after k steps.
- Let $i = 1$, i.e., we selected the first path

Proof by induction

- Let u_k the number of uncovered paths after k steps.
- Let $i = 1$, i.e., we selected the first path
- The sum of all nodes of all paths is $p\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path

Proof by induction

- Let u_k the number of uncovered paths after k steps.
- Let $i = 1$, i.e., we selected the first path
- The sum of all nodes of all paths is $p\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only n nodes in total, on average a node covers $p\ell / n$ paths

Proof by induction

- Let u_k the number of uncovered paths after k steps.
- Let $i = 1$, i.e., we selected the first path
- The sum of all nodes of all paths is $p\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only n nodes in total, on average a node covers $p\ell/n$ paths
- There must be one node that covers at least $p\ell/n$ paths.

Proof by induction

- Let u_k the number of uncovered paths after k steps.
- Let $i = 1$, i.e., we selected the first path
- The sum of all nodes of all paths is $p\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only n nodes in total, on average a node covers $p\ell/n$ paths
- There must be one node that covers at least $p\ell/n$ paths.
- The greedy algorithm selects such a node that covers most paths, we "loose" at least $p\ell/n$ many paths in that step, hence we have

$$u_1 \leq p - \frac{p\ell}{n} = p\left(1 - \frac{\ell}{n}\right)$$

many paths left.

Induction step

- After the i-th iteration we have $u_i \leq p\left(1 - \frac{\ell}{n}\right)^i$ paths left.

Induction step

- After the i-th iteration we have $u_i \leq p\left(1-\frac{\ell}{n}\right)^i$ paths left.
- The sum of all nodes of all uncovered paths is $u_i\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path

Induction step

- After the i-th iteration we have $u_i \leq p\left(1-\frac{\ell}{n}\right)^i$ paths left.
- The sum of all nodes of all uncovered paths is $u_i\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only $n-i$ nodes left, on average a node covers $u_i\ell/n-i \geq u_i\ell/n$ paths

Induction step

- After the i-th iteration we have $u_i \leq p\left(1-\frac{\ell}{n}\right)^i$ paths left.
- The sum of all nodes of all uncovered paths is $u_i \ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only $n - i$ nodes left, on average a node covers $u_i \ell / n - i \geq u_i \ell / n$ paths
- There must be one node that covers at least $u_i \ell / n$ paths.

Induction step

- After the i-th iteration we have $u_i \leq p\left(1-\frac{\ell}{n}\right)^i$ paths left.
- The sum of all nodes of all uncovered paths is $u_i\ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only $n - i$ nodes left, on average a node covers $u_i\ell/n - i \geq u_i\ell/n$ paths
- There must be one node that covers at least $u_i\ell/n$ paths.
- Together we get

$$u_{i+1} \leq u_i - u_i\ell/n = u_i \cdot (1 - \ell/n) \leq p\left(1-\frac{\ell}{n}\right)^i\left(1-\frac{\ell}{n}\right) = p\left(1-\frac{\ell}{n}\right)^{i+1}$$

Induction step

- After the i-th iteration we have $u_i \leq p\left(1-\frac{\ell}{n}\right)^i$ paths left.
- The sum of all nodes of all uncovered paths is $u_i \ell$ if we count nodes multiple time, i.e., as many times as a node appears on a path
- We have only $n - i$ nodes left, on average a node covers $u_i \ell / n - i \geq u_i \ell / n$ paths
- There must be one node that covers at least $u_i \ell / n$ paths.
- Together we get

$$u_{i+1} \leq u_i - u_i \ell / n = u_i \cdot (1 - \ell/n) \leq p\left(1-\frac{\ell}{n}\right)^i \left(1-\frac{\ell}{n}\right) = p\left(1-\frac{\ell}{n}\right)^{i+1}$$

- This proofs our claim

3c

- Show that for $i > \frac{n}{\ell} \ln p$ all paths are covered. *Hint:* $1 - x < e^{-x}$ for all $x > 0$

3c

- Show that for $i > \frac{n}{\ell} \ln p$ all paths are covered. *Hint:* $1 - x < e^{-x}$ for all $x > 0$
-

3c

- Show that for $i > \frac{n}{\ell} \ln p$ all paths are covered. *Hint:* $1 - x < e^{-x}$ for all $x > 0$
-
- We show that $i > \frac{n}{\ell} \ln p$ implies $p(1 - \frac{\ell}{n})^i < 1$ which means that all paths are covered by greedily selecting i nodes (realtime calculation)

$$
\begin{aligned}
& i > \tfrac{n}{\ell} \ln p \\
\iff \quad & -\tfrac{i\ell}{n} < -\ln p \\
\iff \quad & e^{-\frac{i\ell}{n}} < \tfrac{1}{p} \\
\iff \quad & \left(e^{-\frac{\ell}{n}}\right)^i < \tfrac{1}{p} \\
\overset{\text{Hint}}{\implies} \quad & \left(1 - \tfrac{\ell}{n}\right)^i < \tfrac{1}{p} \\
\iff \quad & p\left(1 - \tfrac{\ell}{n}\right)^i < 1
\end{aligned}
$$

Exercise 4

In the lecture, we showed that every (undirected) graph with edge connectivity λ has at most $n^{2\alpha}$ cuts of size at most $\alpha \cdot \lambda$. Use this fact to prove the following statement:

Let $G = (V, E)$ be an undirected graph with constant edge connectivity $\lambda \geq 1$ and let $p := \min\left\{1, \frac{c \ln n}{\lambda}\right\}$, where $c > 0$ is a constant. Assume that edge $e \in E$ is sampled independently with probability p. Let E_p be the set of sampled edges and let $G_p = (V, E_p)$ be the graph induced by the sampled edges. Show that if the constant c is chosen sufficiently large, the graph G_p is connected with high probability.

Hint: A graph is connected if and only if there is an edge across each of the $2^{n-1} - 2$ *possible cuts. Analyze the probability that for a cut of a given size* k *in* G*, at least one edge is sampled in* E_p*. Then, use the upper bound from the lecture on the number of cuts of a given size and a union bound over all cuts of a given size in* G*. Finally, one can do a union bound over all possible cut sizes.*

Ex 4 Solution

- We assume $p < 1$, otherwise $G_p = G$ is obviously connected.

$$
\begin{aligned}
(1-p)^k &= \Big(1 - \frac{c \ln n}{\lambda}\Big)^k \\
&\overset{(*)}{<} \exp\Big(-\frac{kc \ln n}{\lambda}\Big) \\
&= \exp\Big(-\alpha c \ln n\Big) = n^{-c\alpha}
\end{aligned}
$$

$(*) : 1 - x < e^{-x}$

Ex 4 Solution

- We assume $p < 1$, otherwise $G_p = G$ is obviously connected.
- The graph G_p becomes disconnected if all edges of any cut are removed.

$$
\begin{aligned}
(1-p)^k &= \left(1 - \frac{c \ln n}{\lambda}\right)^k \\
&\overset{(*)}{<} \exp\left(-\frac{kc \ln n}{\lambda}\right) \\
&= \exp\left(-\alpha c \ln n\right) = n^{-c\alpha}
\end{aligned}
$$

$(*) : 1 - x < e^{-x}$

Ex 4 Solution

- We assume $p < 1$, otherwise $G_p = G$ is obviously connected.
- The graph G_p becomes disconnected if all edges of any cut are removed.
- We first consider one fixed cut of size $k := \alpha\lambda$

$$
\begin{aligned}
(1-p)^k &= \left(1 - \frac{c \ln n}{\lambda}\right)^k \\
&\overset{(*)}{<} \exp\left(-\frac{kc \ln n}{\lambda}\right) \\
&= \exp\left(-\alpha c \ln n\right) = n^{-c\alpha}
\end{aligned}
$$

$(*) : 1 - x < e^{-x}$

Ex 4 Solution

- We assume $p < 1$, otherwise $G_p = G$ is obviously connected.
- The graph G_p becomes disconnected if all edges of any cut are removed.
- We first consider one fixed cut of size $k := \alpha\lambda$
- The probability that we remove all edges of a cut of size $k := \alpha\lambda$ is $(1-p)^k$. We obtain

$$
\begin{aligned}
(1-p)^k &= \left(1 - \frac{c \ln n}{\lambda}\right)^k \\
&\overset{(*)}{<} \exp\left(-\frac{kc \ln n}{\lambda}\right) \\
&= \exp\left(-\alpha c \ln n\right) = n^{-c\alpha}
\end{aligned}
\qquad (*) : 1 - x < e^{-x}
$$

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$
- In the lecture we saw that we have at most $n^{2\alpha}$ cuts of size $k = \alpha\lambda$.

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$
- In the lecture we saw that we have at most $n^{2\alpha}$ cuts of size $k = \alpha\lambda$.
- Let us enumerate these cuts with $C_1, \ldots, C_\ell$ with $\ell \leq n^{2\alpha}$

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$
- In the lecture we saw that we have at most $n^{2\alpha}$ cuts of size $k = \alpha\lambda$.
- Let us enumerate these cuts with $C_1, \ldots, C_\ell$ with $\ell \leq n^{2\alpha}$
- Let $\mathcal{E}_i^k$ be the event that all edges of a specific cut C_i are removed.

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$
- In the lecture we saw that we have at most $n^{2\alpha}$ cuts of size $k = \alpha\lambda$.
- Let us enumerate these cuts with $C_1, \ldots, C_\ell$ with $\ell \leq n^{2\alpha}$
- Let $\mathcal{E}_i^k$ be the event that all edges of a specific cut C_i are removed.
- Let $\mathcal{E}^k := \bigcup_{i=1}^{\ell} \mathcal{E}_i^k$ be the event that all edges of at least one of the cuts C_i out of the cuts $C_1, \ldots, C_\ell$ of size k are removed.

- Last slide: cut of size $\alpha\lambda$ has no edges with probability $< n^{-c\alpha}$
- In the lecture we saw that we have at most $n^{2\alpha}$ cuts of size $k = \alpha\lambda$.
- Let us enumerate these cuts with $C_1, \dots, C_\ell$ with $\ell \leq n^{2\alpha}$
- Let $\mathcal{E}_i^k$ be the event that all edges of a specific cut C_i are removed.
- Let $\mathcal{E}^k := \bigcup_{i=1}^{\ell} \mathcal{E}_i^k$ be the event that all edges of at least one of the cuts C_i out of the cuts $C_1, \dots, C_\ell$ of size k are removed.
- We do a union bound and get

$$
\begin{aligned}
\mathbb{P}(\mathcal{E}^k) = \mathbb{P}\Big(\bigcup_{i=1}^{\ell} \mathcal{E}_i^k\Big) &\leq \sum_{i=1}^{\ell} \mathbb{P}(\mathcal{E}_i^k) \\
&\leq \ell n^{-c\alpha} \leq n^{2\alpha} \cdot n^{-c\alpha} = n^{-(c-2)\alpha} = n^{-(c-2)k/\lambda}
\end{aligned}
$$

- Result from last slide: The probability that **any** cut with size **exactly** k has no edges is $\leq n^{-(c-2)k/\lambda}$.

- Result from last slide: The probability that **any** cut with size **exactly** k has no edges is $\leq n^{-(c-2)k/\lambda}$.
- Finally we bound the probability that one cut of *any* cut of *any* size k gets all its edges removed.

- Result from last slide: The probability that **any** cut with size **exactly** k has no edges is $\leq n^{-(c-2)k/\lambda}$.
- Finally we bound the probability that one cut of *any* cut of *any* size k gets all its edges removed.
- Clearly we have $k \leq n$.

- Result from last slide: The probability that **any** cut with size **exactly** k has no edges is $\leq n^{-(c-2)k/\lambda}$.
- Finally we bound the probability that one cut of *any* cut of *any* size k gets all its edges removed.
- Clearly we have $k \leq n$.
- Let $\mathcal{E} := \bigcup_{k=1}^{n} \mathcal{E}^k$. We obtain

$$\begin{aligned}
\mathbb{P}(\mathcal{E}) = \mathbb{P}\Big(\bigcup_{k=1}^{n} \mathcal{E}^k\Big) &\leq \sum_{k=1}^{n} \mathbb{P}(\mathcal{E}^k) \\
&\leq \sum_{k=1}^{n} n^{-(c-2)k/\lambda} \leq \sum_{k=1}^{n} n^{-(c-2)/\lambda} \\
&= n \cdot n^{-(c-2)/\lambda} = n^{-(c-2-\lambda)/\lambda}
\end{aligned}$$

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$
- We desire that this occurs with probability at most $n^{-c'}$ for some constant $c' > 0$.

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$
- We desire that this occurs with probability at most $n^{-c'}$ for some constant $c' > 0$.
- Then the graph is connected with high probability $(1 - n^{-c'})$

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$
- We desire that this occurs with probability at most $n^{-c'}$ for some constant $c' > 0$.
- Then the graph is connected with high probability $(1 - n^{-c'})$
- For this we just need to set our constant c such that

$$(c - 2 - \lambda)/\lambda > c'$$

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$
- We desire that this occurs with probability at most $n^{-c'}$ for some constant $c' > 0$.
- Then the graph is connected with high probability $(1 - n^{-c'})$
- For this we just need to set our constant c such that

$$(c - 2 - \lambda)/\lambda > c'$$

- This is the case for

$$c > c'\lambda + \lambda + 2$$

- Result from last slide: The probability that the graph is not connected (that any cut of any size has no edge) is $\leq n^{-(c-2-\lambda)/\lambda}$
- We desire that this occurs with probability at most $n^{-c'}$ for some constant $c' > 0$.
- Then the graph is connected with high probability $(1 - n^{-c'})$
- For this we just need to set our constant c such that

$$(c - 2 - \lambda)/\lambda > c'$$

- This is the case for

$$c > c'\lambda + \lambda + 2$$

- This term is independent from the graph size n and thus indeed a constand as required.