CS5330: Randomized Algorithms

Problem Set 1—Solutions

Due: January 22, 6:30pm

Instructions. The *exercises* at the beginning of the problem set do not have to be submitted—though you may. The first three exercises should be easy review of basic probability (i.e., material you already know prior to this class). The remaining two exercises are more interesting and involve the minimum cut algorithm from class. There are three problems to submit. The first involves a basic probability calculation, while the second is a simple algorithm. The last question is a bit more involved and requires generalizing the algorithm and analysis from class.

- Please submit the problem set on IVLE in the appropriate folder. (Typing the solution using latex is recommended.) If you want to do the problem set by hand, please submit it at the beginning of class.
- Start each problem on a separate page.
- If you submit the problem set on paper, make sure your name is on each sheet of paper (and legible).
- If you submit the problem set on paper, staple the pages together.

Remember, that when a question asks for an algorithm, you should:

- First, give an overview of your answer. Think of this as the executive summary.
- Second, describe your algorithm in English, giving pseudocode if helpful.
- Third, give an example showing how your algorithm works. Draw a picture.

You may then give a proof of correctness, or explanation, of why your algorithm is correct, an analysis of the running time, and/or an analysis of the approximation ratio, depending on what the question is asking for.

Advice. Start the problem set early—questions may take time to think about. Come talk to me about the questions. Talk to other students about the problems.

Collaboration Policy. The submitted solution must be your own unique work. You may discuss your high-level approach and strategy with others, but you must then: (i) destroy any notes; (ii) spend 30 minutes on facebook or some other non-technical activity; (iii) write up the solution on your own; (iv) list all your collaborators. Similarly, you may use the internet to learn basic material, but do not search for answers to the problem set questions. You may not use any solutions that you find elsewhere, e.g. on the internet. Any similarity to other students' submissions will be treated as cheating.

Easy Exercises and Review (*May not be submitted.*)

Exercise 1. A fair coin is one that comes up heads with probability $p = 1/2$ and tails with probability $(1 - p) = 1/2$. Flip a fair coin ten times. What is the probability that:

a. The number of heads is equal to the number of tails?

b. There are more heads than tails?

Exercise 2. For each of the following, explain your answer:

a. Suppose that you roll a fair die that has six sides numbered $1, 2, \ldots, 6$. Is the event that the number on top is a multiple of 2 independent of the event that the number of top is a multiple of 3?

b. Suppose that you roll a fair die that has four sides numbered $1, 2, 3, 4$. Is the event that the number on top is a multiple of 2 independent of the event that the number of top is a multiple of 3?

Exercise 3. The FastCut algorithm reduces the problem from size n to $n/\sqrt{2} + 1$ and recurses twice on the remaining graph. Consider a version that reduces the problem from size $n/\sqrt{2}$ and recurses three times on the remaining graph. This would yield the following recurrence:

$$T(n) = 3T(n/\sqrt{2}) + n^2$$

Solve this recurrence.

More Interesting Exercises (*May not be submitted.*)

Exercise 4. Recall in class we described the following algorithm for finding a min-cut in a graph $G = (V, E)$:

- Assign each edge in the graph G a weight randomly chosen from $[0, 1]$. (If the weights are not unique, repeat.)
- Find an MST T of the weighted graph G.
- Let e be the maximum weight edge in T. By deleting edge e from the MST, it divides the tree into two sets of nodes: V_1 and V_2. Return these two sets as your cut.

Answer the following two questions:

a. Explain why this algorithm finds a min-cut with probability at least $2/(n \cdot (n-1))$.

b. Modify this algorithm using the ideas from the FastCut algorithm so that it finds a min-cut with probability at least $1/\log n$. (Hint: you will not want to simply run the MST algorithm to completion, but instead stop early and *do something*.)

Exercise 5. Assume that graph $G = (V, E)$ is a weighted graph. Give an algorithm for finding a min-cut in graph G with probability at least $1/(n \cdot (n-1))$. (Hint: Think about choosing each edge with proportion to its weight.) Prove that your algorithm is correct.

Problem 1. [*ESP*]

Dr. P. T. Fogg[1] is a noted research in telepathy. In order to prove the existence of telepathy, he runs the following experiment: Two psychics are positioned on opposite sides of an opaque, sound-proof barrier. Each has a fair six-sided die. Each time the test is run, the two psychics each roll their own die, and then attempt to guess the value of the *other* psychic's die. If telepathy does not exist, each psychic has a 1/6 chance of guessing the correct value.

Dr. Fogg ran his experiment using the following policy for determining how many times the test is to be run: the psychics rolls their dice repeatedly until both roll a six. At that point, the experiment is immediately halted, before the psychics can make a guess.

Problem 1.a. Surprisingly, Dr. Fogg finds that the psychics are guessing correctly with a probability slightly better than 1/6! Explain, in words, the flaw in this experimental design.

Solution: The critical flaw is that if a psychic rolls a six, and if the game does not end, then the psychic immediately knows that the other psychic did not roll a six. Hence, the psychic who rolled a six can guess a value from the range $1, 2, 3, 4, 5$, i.e., should never guess six.

Problem 1.b. If a psychic exploits this flow optimally, with what probability can she guess the number on the opposite die?

Solution: First, we know that if a psychic rolls a $\{1, 2, 3, 4, 5\}$, then that psychic has a 1/6 chance of guessing correctly. Otherwise, if a psychic rolls a 6, then that psychic has a 1/5 chance of guessing correctly. It remains to calculate the probability that a psychic does or does not roll a 6.

In particular, notice that all events in question are conditioned on the fact that the experiment does not end. We are really asking for the probability that a psychic guesses correctly *given that the game does not end on that roll.* Let G be the event that the game does not end. Let X be the roll of the psychic making the guess, and let Y be the roll of the other psychic.

$$\begin{aligned}
\Pr[X = 6|G] &= \Pr[X = 6 \cap G]/\Pr[G] \\
&= \Pr[X = 6 \cap Y \neq 6]/\Pr[X \neq 6 \cup Y \neq 6] \\
&= \frac{\Pr[X = 6]\Pr[Y \neq 6]}{1 - \Pr[X = 6 \cap Y = 6]} \\
&= \frac{(1/6)(5/6)}{35/36} \\
&= \frac{5}{36}\frac{36}{35} \\
&= 1/7
\end{aligned}$$

Notice that this relies on the independence of the coin flips by the two psychics. Similarly, we can calculate the $\Pr[X \neq 6|G] = 1 - \Pr[X = 6|g] = 6/7$.

Finally, we conclude that the probability of success for a psychic is:

$$\begin{aligned}
&= \Pr[X \neq 6|G](1/6) + \Pr[X = 6|G](1/5) \\
&= (6/7)(1/6) + (1/7)(1/5) \\
&= 6/35
\end{aligned}$$

[1]The "T" stands for "telepath."

Problem 2. [*All minimum cuts.*]

Give a (simple) algorithm for finding *all* the min-cuts in a graph, with high probability. Prove that your algorithm is correct. (Hint: if your proof involves a long analysis of graph structure, then there is a simpler solution.)

Solution:

Overview. The basic idea is to repeat the FastCut algorithm $2(c+2)\log^2(n)$, which will ensure that with high probability, each min-cut is discovered at least once. The FastCut algorithm is modified to return *all* the minimum cuts discovered, not just one.

Details. Fix some min-cut C. We know that if you run the FastCut algorithm, it will find cut C at *some* leaf of the recursion tree with probability at least $1/2\log(n)$. If we run the FastCut algorithm $2(c+2)\log^2(n)$ times, then the probability that it does not find cut C is:

$$\begin{aligned}\left(1-\frac{1}{2\log n}\right)^{2(c+2)\log^2(n)} &\leq e^{-(c+2)\log n}\\ &\leq \frac{1}{n^{(c+2)}}\end{aligned}$$

We know that there are at most n^2 possible different min-cuts $C_1, C_2, \ldots$. Thus during our $(c+2)\log^2(n)$ repetitions of FastCut, the probability that we do not see *any one* of these is $\sum(1/n^{c+2}) \leq 1/n^c$ (since there are at most n^2 min-cuts). Thus, with high probability, we have a set containing all the minimum cuts.

Looking at the set of cuts discovered, we can identify a cut with minimal value, and discard any larger cut. The result should be a set containing exactly the min-cuts.

Discussion. Notice that the FastCut algorithm does not guarantee that it will find a specific cut C; it only guarantees it will find a cut with value $\leq C$, as it returns the minimum discovered in the recursion tree. Thus you cannot claim that FastCut returns your specific cut with probability at least $1/2\log n$.
In fact, the FastCut algorithm actually finds more than one cut per invocation—otherwise this claim would not make sense. In particular, each invocation of FastCut finds $\Theta(n^2)$ different cuts, more than one of which may be minimal. To see this, count the number of leaves in the branching process; since the tree is of height $2\log(n)$, it has n^2 leaves. Each leaf represents a cut.

Luckily, it turns out that the FastCut algorithm does guarantee that for any minimum cut C, with probability $1/2\log n$, one of the leaves will be your cut C. Thus you need to modify the FastCut algorithm to return *all* cuts discovered, rather than the minimum cut discovered.

A simpler alternate solution would be to repeatedly run the slower KargerCut algorithm that collapses the graph to 2 nodes each time. In this case, we do not need to modify the min-cut algorithm at all (but more repetitions are needed to find all the min-cuts).

Problem 3. [*Almost minimum cuts.*]

Given a graph G with a min-cut of size k, let C be a cut of size (exactly) $\alpha \cdot k$. (For simplicity, you may assume that 2α is an integer, though it is not necessary for the final conclusion.)

Problem 3.a. Assume you run *Collapse*$(G, n, 2\alpha)$, i.e., you repeatedly contract edges until there are only 2α nodes left. Show that the probability that no edge in C was contracted is at least $\Omega(1/\binom{n}{2\alpha})$.

Solution:

Overview. The analysis mirrors that done for min-cuts. First we determine the probability that a given contraction contracts an edge in C, and then we multiple the probabilities together to get the probability that no edge in C is chosen throughout the contraction process.

Details. First, we compute the probability of contracting an edge in cut C. As before, we know that the total number of edges in the graph $m \geq kn/2$. Since there are αk edges in the cut, the probability of choosing an edge in the cut is at most $2\alpha/n$; thus the probability of not choosing an edge in the cut is at least $(n - 2\alpha)/n$.

More generally, after i steps of edge contraction, there remain at least $m \geq k(n-i)/2$ edges in the graph, so the probability of choosing any of the αk edges in the cut is at most $2\alpha/(n-i)$; thus the probability of not choosing an edge in the cut is at least $(n - 2\alpha - i)/(n - i)$.

We repeatedly contract edges until there are only 2α edges left, so in the last invocation $(n - i) = 2\alpha + 1$, i.e., $i = n - 2\alpha - 1$.Therefore, the probability of *not* contracting an edge in the cut is at least:

$$\prod_{i=0}^{n-2\alpha-1} \frac{n - 2\alpha - i}{n - i}$$

The numerator here is equal to $(n - 2\alpha)!$, and the denominator here is equal to $n!/(2\alpha)!$. Since $\binom{n}{2\alpha} = n!/(n-2\alpha)!(2\alpha)!$, we get that the probability of not contracting an edge in cut C is at least $\Omega(1/\binom{n}{2\alpha})$.

Problem 3.b. Assume you have a graph with 2α nodes containing cut C. Choose a cut at random from the graph. What is the probability that you return graph cut C?

Solution: Since there are $2^{2\alpha}$ cuts in a graph with 2α nodes, the probability of choosing any one of those cuts is $2^{-2\alpha}$.

Problem 3.c. Prove that there are at most $O((2n)^{2\alpha})$ cuts of size (exactly) $\alpha \cdot k$.

Solution:

Overview. First, we summarize an algorithm for finding a cut of size at most $\alpha \cdot k$. We calculate the probability that the algorithm finds such a cut, and use that to bound the number of possible such cuts. If $\alpha = 1$, we already know that there are at most n^2 min-cuts. Here, we assume that $\alpha \geq 1.5$. (Recall we are assuming that 2α is integral.)

Details. First, we review the algorithm for finding such a cut C: First collapse the graph to size 2α. Then choose a cut at random.

The probability of not collapsing any edge in cut C is at least $\binom{n}{2\alpha}^{-1}$, and the probability of then selecting cut C is at least $2^{-2\alpha}$. We observe that:

$$\begin{aligned} 2^{2\alpha}\binom{n}{2\alpha} &\leq 2^{2\alpha}\left(\frac{en}{2\alpha}\right)^{2\alpha} \\ &\leq \left(\frac{en}{\alpha}\right)^{2\alpha} \\ &\leq (2n)^{2\alpha} \end{aligned}$$

Thus the probability of choosing cut C is at least $(2n)^{-2\alpha}$.

Finally, assume for the sake of contradiction that there are $> (2n)^{2\alpha}$ cuts of size exactly αk. The algorithm returns each of these cuts with probability $(2n)^{-2\alpha}$. Since these are all disjoint events in our probability space, their probabilities must sum to at most 1. This contradicts the assumption that there are $> (2n)^{2\alpha}$ such cuts.

Problem 3.d. (*Challenge:*) Assume that for any (real) value of α, there are at most $O((2n)^{2\alpha})$ cuts of size $\alpha \cdot k$. Let G be a connected graph with a minimum cut size of k.

Mark each edge in G with probability $c\ln(2n)/k$, for some constant c. (Note that this is only meaninful if $k > c\ln(2n)$.) Delete every unmarked edges from the graph. Prove that with high probability graph G is still connected.

(The claim is true with high probability of the probability is at least $1 - 1/n^{c'}$, where given c' we can choose c above to yield the proper result. For example, we might choose $c = c' + 8$.)

Hint: a graph is connected if, for every cut, there is at least one edge across the cut.

Solution: To show that the graph is still connected, we have to show that for every cut in the graph, there is at least one marked edge. Notice that there are at most n^2 different cut sizes in graph G, since every cut has an (integer) size from 1 to $\binom{n}{2}$ (since every cut contains a subset of the edges). There are therefore at most n^2 values $\alpha_1, \alpha_2, \ldots, \alpha_{n^2}$ where every cut has size $\alpha_j k$ for some j.
Let C be a cut of size $\alpha_j k$. The probability that no edge in C is marked is at most:

$$\begin{aligned} (1 - c\ln(2n)/k)^{\alpha_j k} &\leq e^{-c\ln(2n)\alpha_j} \\ &\leq (2n)^{-c\alpha_j} \end{aligned}$$

We know that there are at most $(2n)^{2\alpha_j}$ cuts of size $\alpha_j \cdot k$, so by a union bound, we conclude that the probability that *any* cut of size $\alpha_j k$ has no edge marked is at most:

$$\begin{aligned} (2n)^{2\alpha_j}(2n)^{-c\alpha_j} &\leq (2n)^{-(c-2)\alpha_j} \\ &\leq n^{-(c-2)} \end{aligned}$$

(since $\alpha_j \geq 1$). Now taking a union bound over all n^2 values of α, we see that the probability that any cut has no marked edge is at most $n^2/n^{c-2} \leq 1/n^{c-4}$. This, with high probability, every cut has at least one marked edge and hence the graph is connected.