

## Class 2: Agenda and Questions

**1 Warm-up**

There are  $n$  pigeons and  $n$  pigeon-holes; each pigeon has its own pigeon-hole. However, after a wild night the  $n$  pigeons return to a uniformly random pigeon-hole (so it could be that some holes are empty, and some have more than one pigeon). What's the expected number of empty pigeon-holes?

**Group Work: Solutions**

The expected number of empty holes, by linearity of expectation, is

$$\mathbb{E} \sum_{i=1}^n \mathbf{1}[\text{hole } i \text{ is empty}] = \sum_i \Pr[\text{hole } i \text{ is empty}].$$

The probability that any one hole is empty is  $(1 - 1/n)^n$ . (that is, with probability  $1 - 1/n$  independently for each of  $n$  pigeons, a pigeon does not pick that hole). Thus, the expected number of empty holes is

$$n(1 - 1/n)^n.$$

As  $n$  gets large, this tends to  $n/e$ . (This is because  $1 - 1/n \approx e^{-1/n}$  when  $n$  is large; you can see this by writing out the Taylor series for  $e^{-1/n} = 1 - 1/n + 1/(2n^2) - 1/(6n^3) + \dots$ ).

**2 Questions?**

Any questions from the short lecture videos or the quiz?

**3 Coupon Collecting**

This will be a good chance to practice the power of linearity of expectation.

Let  $W$  be a collection of  $n$  words. For example,  $W = \{\text{hello, randomized, algorithms, penguin, spaceship, \dots, mushroom}\}$ . There's a button in front of you. Each time you push the button, it will say a word uniformly at random from  $W$ . If you push the button multiple times, it answers independently each time. (In particular, it's possible to get the same word twice). The question is:

What is the expected number of times you push the button before you see all  $n$  words?

### 3.1 Group work

You'll answer this question by answering the following questions in your groups.

Let  $X_i$  be the time at which you see the  $i$ 'th new word. For example, if you push the button six times and see

penguin, penguin, randomized, hello, randomized, spaceship

then  $X_1 = 1$ , since you saw your first new word (“penguin”) on push 1.  $X_2 = 3$ , since you saw the second new word (“randomized”) on push 3. And  $X_3 = 4, X_4 = 6$ .

#### Group Work

1. What is  $\mathbb{E}X_1$ ? (This is not a trick question).
2. What is  $\mathbb{E}(X_2 - X_1)$ ? That is, in expectation, how many times do you press the button, after you have seen the first word, before you see a new, second word?
3. What is  $\mathbb{E}(X_3 - X_2)$ ?
4. For any  $i = 2, 3, \dots, n$ , what is  $\mathbb{E}(X_i - X_{i-1})$ ?
5. Use your answers to the above, plus linearity of expectation, to answer our question: what is the expected number of times you push the button before you see all  $n$  words? It's okay if your answer is a summation, but if you have time try to simplify it to get a big-Theta expression.

#### Group Work: Solutions

1.  $\mathbb{E}X_1 = 1$ .
2.  $\mathbb{E}(X_2 - X_1) = \frac{1}{1-1/n}$ . This is because we have a  $1 - 1/n$  chance of getting a word other than the first one. We saw in the lecture video on linearity of expectation (or you know anyway) that the expected number of times you have to flip a  $p$ -biased coin before seeing heads is  $1/p$ , so the answer is  $1/(1 - 1/n)$ .
3.  $\mathbb{E}(X_3 - X_2) = \frac{1}{1-2/n}$ .
4.  $\mathbb{E}(X_i - X_{i-1}) = \frac{1}{1-(i-1)/n}$

5. We have

$$\begin{aligned}
 \mathbb{E}X_n &= \mathbb{E}X_1 + (X_2 - X_1) + (X_3 - X_2) + \cdots + (X_n - X_{n-1}) = \mathbb{E}X_1 + \sum_{i=1}^{n-1} \mathbb{E}(X_{i+1} - X_i) \\
 &= 1 + \sum_{i=1}^{n-1} \frac{1}{1 - i/n} \\
 &= 1 + \sum_{i=1}^{n-1} \frac{n}{i} \\
 &= 1 + n \sum_{i=1}^{n-1} \frac{1}{i} \\
 &= \Theta(n \log n).
 \end{aligned}$$

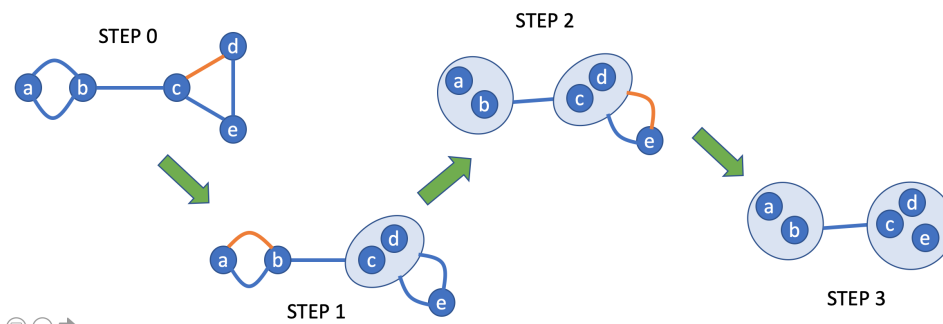
To see this last thing, you could approximate  $\sum_{i=1}^n \frac{1}{i} \approx \int_{x=1}^n \frac{1}{x} dx = \log(n)$ . (Note: if we wanted a tighter bound than a  $\Theta(\cdot)$  expression, we could get that too by being more careful about the  $\approx$ ...you'll do this on your homework!)

## 4 Karger-Stein Algorithm

Next, we'll take a look at a way to speed up Karger's algorithm.

### 4.1 Motivation for Karger-Stein

Here is one small run of Karger's algorithm:



#### Group Work

What is the probability of failure at each point in this run? That is, what is the probability that we choose an edge crossing the minimum cut?

Based on your answers, think of ways to improve Karger's algorithm. (In particular, might there be a smarter way to boost the success probability than just to repeat it a

bunch of times?)

### Group Work: Solutions

The answer is  $1/6, 1/5, 1/3$ . We see that repeating Karger's algorithm is wasteful since earlier steps are more likely to be successful than later steps; we should repeat later steps more!

Your answer perhaps motivates the following algorithm:

MODIFIED-KARGER:

1. Start with a graph  $G$  on  $n$  vertices.
2. Run Karger's algorithm (once) until there are  $m$  vertices remaining. Call the graph you end up with  $G'$ . (Note that  $G'$  will have some "mega-vertices" comprised of merged vertices).
3. Repeat Karger's algorithm  $k$  times independently on  $G'$  until we end up with only two mega-vertices. Return the smallest cut we find.

## 4.2 Group Work

In this group work, you will analyze the Modified-Karger algorithm above.

### Group Work

1. Give a bound on the probability, in terms of  $n$  and  $m$  and  $k$ , that MODIFIED-KARGER is successful. (You may not be able to find the probability exactly, but give a decent lower bound, like we did for the original Karger's algorithm; it's okay if your expression is a bit complicated).

You may want to consult the lecture notes on Karger's algorithm. They are available on the course website: [cs265.stanford.edu](http://cs265.stanford.edu)

*Hint:* You can break up the failure probability into two parts: the probability that we choose an edge crossing the mincut when reducing  $G$  to  $G'$ , and the probability that we choose an edge crossing the cut in all of the  $k$  runs of Karger of  $G'$ .

2. Choose  $m = \sqrt{n}$  and  $k = n \log n$ . Use part 1 to show that the success probability of MODIFIED-KARGER is  $\Omega(1/n)$ .

*Hint:* The fact that we're looking for a  $\Omega(\cdot)$  answer means that it's okay to ignore pesky constants in your analysis.

*Hint:* You might want to use the useful fact that  $1 - x \leq e^{-x}$  for all  $x$ .

3. Show that if you repeat MODIFIED-KARGER, with the parameters above,  $\Theta(n)$  times, you can obtain a success probability of 0.99.

4. How does this compare to the original Karger's algorithm? More precisely:

We saw in the mini-lecture that repeating Karger's algorithm  $O(n^2)$  times leads to a success probability of 0.99. This would involve  $O(n^3)$  edge contractions ( $n$  edge contractions per run of Karger's algorithm).

You've just shown that "repeating MODIFIED-KARGER  $\Theta(n)$  times" also has a success probability of 0.99. How many edge contractions does this involve?

5. Think about the following two challenge problems:

- Above, we saw how to improve Karger's algorithm by breaking one step down into two steps. What if we were to iterate on this idea? (That is, instead of just running Karger on  $G'$ , recursively run MODIFIED-KARGER on  $G'$ ). How would you pick the parameters here? How few edge contractions can you get, if you want success probability 0.999?
- Can you derandomize Karger's algorithm?

### Group Work: Solutions

1. The probability that the Modified Karger's algorithm fails is the probability that the first step fails or the second step fails. By the union bound, this is bounded above by

$$\Pr[G \rightarrow G' \text{ fails}] + (\Pr[\text{fail on } G'])^k.$$

The probability that Karger's algorithm chooses an edge crossing the min cut when reducing  $G$  to  $G'$  is at most

$$1 - \left(\frac{n-2}{n}\right) \left(\frac{n-3}{n-1}\right) \cdots \left(\frac{m}{m+2}\right) \left(\frac{m-1}{m+1}\right) = 1 - \frac{m(m-1)}{n(n-1)},$$

using exactly the same reasoning we used when we analyzed Karger's algorithm in the mini-lecture.

The probability that Karger's algorithm fails on  $G'$  is  $1 - \frac{1}{m(m-1)}$ , since this is just plugging in the result from the mini-lecture into a graph with  $m$  vertices instead of  $n$  vertices.

Adding them together, we get a bound of:

$$1 - \frac{m(m-1)}{n(n-1)} + \left(1 - \frac{1}{m(m-1)}\right)^k$$

2. Plugging in  $m = \sqrt{n}$  and  $k = n \log n$ , and (following the hint), ignoring pesky

constants, the failure probability from part 1 is at most

$$\begin{aligned}
 1 - \frac{m(m+1)}{n(n-1)} + \left(1 - \frac{2}{m(m-1)}\right)^k &\approx 1 - \frac{m^2}{n^2} + (1 - 2/m^2)^k \\
 &\leq 1 - \frac{n}{n^2} + (e^{-2/n})^{n \log n} \\
 &= 1 - \frac{1}{n} + e^{-2 \log n} \\
 &= 1 - \frac{1}{n} + \frac{1}{n^2} \\
 &\leq 1 - 1/2n.
 \end{aligned}$$

**Note!** In fact you can get the same thing taking  $k = n$  instead of  $k = n \log n$ , with a slightly more slick analysis. (Thanks to Shurui Liu for pointing this out!) Instead of using the union bound on the two failure probabilities, let's multiply the two probabilities of success together. In more detail:

Let  $A$  be the event that the min cut in  $G$  is still present in  $G'$  (that is, Karger's algorithm didn't fail in the first step). Let  $B$  be the event that the min cut in  $G'$  is found in at least one of the  $k$  runs of the "parallel" Karger's algorithm. As above, we have

$$\Pr[A] \geq \frac{m(m-1)}{n(n-1)}$$

and

$$\Pr[B] \geq 1 - \left(1 - \frac{2}{m(m-1)}\right)^k.$$

Now, observe that if both  $A$  and  $B$  occur, then Modified-Karger is successful. So we write

$$\begin{aligned}
 \Pr[\text{Modified-Karger wins}] &= \Pr[A \wedge B] \\
 &= \Pr[A] \Pr[B|A] \\
 &\geq \Pr[A] \left(1 - \left(1 - \frac{2}{m(m-1)}\right)^k\right) \\
 &\geq \frac{m(m-1)}{n(n-1)} \cdot \left(1 - \left(1 - \frac{2}{m(m-1)}\right)^k\right),
 \end{aligned}$$

Above, you might be concerned, since we wrote that  $\Pr[B|A] \geq [\text{stuff}]$ , when above we only showed that  $\Pr[B] \geq [\text{stuff}]$ . As written, the events  $A$  and  $B$  actually aren't independent; that's because the event  $B$  depends on  $G'$ , which depends on the event  $A$ . (To see an example where these aren't independent, see the example below). However, what we actually showed above was that for *any particular* graph

$G'$ , the probability that Karger's algorithm succeeds on that graph is at least  $p := 1 - \left(1 - \frac{2}{m(m-1)}\right)^k$ . So we can write

$$\begin{aligned} \Pr[B|A] &= \sum_{\text{all graphs } G'} \Pr[G'|A] \Pr[B|G', A] \\ &= \sum_{\text{all graphs } G'} \Pr[G'|A] \Pr[B|G'] \\ &\geq \sum_{\text{all graphs } G'} \Pr[G'|A] p \\ &= p, \end{aligned}$$

where in the last line we used that  $\sum_{\text{all graphs } G'} \Pr[G'|A] = 1$ . So in fact it is okay to do this.

Now we can plug in  $m = \sqrt{n}$  and  $k = n$  to see that

$$\begin{aligned} \Pr[\text{Modified-Karger wins}] &\geq \frac{m(m-1)}{n(n-1)} \cdot \left(1 - \left(1 - \frac{2}{m(m-1)}\right)^k\right) \\ &\approx \frac{n}{n^2} \cdot (1 - e^{-2k/n}) \\ &= \frac{1}{n} (1 - 1/e^2) \\ &= \Omega(1/n), \end{aligned}$$

the same asymptotic result as we got before! Above, we have ignored some  $\pm 1$ 's when simplifying (this is okay eventually because of the  $\Omega(\cdot)$  at the end); and we used the fact that  $1 - x \approx e^{-x}$  for small  $x$ .

This is an example where the union bound was lossy. More generally, let  $A$  and  $B$  be two (good) events, and suppose that they are independent. Then the probability that they both occur is exactly

$$\Pr[A \wedge B] = (1 - \Pr[\bar{A}])(1 - \Pr[\bar{B}]) = 1 - (\Pr[\bar{A}] + \Pr[\bar{B}]) + \Pr[\bar{A}] \Pr[\bar{B}].$$

The bound that we get with the union bound is

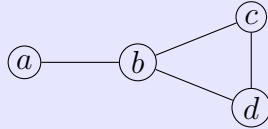
$$\Pr[A \wedge B] = 1 - \Pr[\bar{A} \vee \bar{B}] \geq 1 - (\Pr[\bar{A}] + \Pr[\bar{B}]).$$

So we can see that the union bound is off by the term  $\Pr[\bar{A}] \Pr[\bar{B}]$ .

In some cases, this last term is small relative to the other two, so it doesn't matter. But in other cases (like this one!) it actually makes a difference. Of course, the advantage of the union bound is that it works for *any* two events,  $A$  and  $B$ , they

don't have to be independent. So it can often be easier to apply the union bound if you are worried about how your events depend on each other.

To see that  $A$  and  $B$  aren't necessarily independent, consider the following graph.



Say that  $m = 3$  and  $k = 1$ , and let's compute both  $\Pr[B]$  and  $\Pr[B|A]$ . Since  $n = 4$ , that means that our first run of Karger's algorithm will only run for one step, and so will our second.

With probability  $1/4$ , the first run of Karger's algorithm accidentally contracts the min cut (the edge  $\{a, b\}$ ), and we end up with  $G'$  being a triangle. From there the second run of Karger's algorithm will always find the correct min-cut of  $G'$  (since any cut is a correct min-cut of a triangle).

With probability  $3/4$ , the first run of Karger's algorithm contracts one of the edges in the triangle on  $b, c, d$ . Then  $G'$  is a graph that looks like this:



Now, the probability that the second run of Karger's algorithm correctly finds the min-cut of  $G'$  is  $2/3$ , and the probability that it doesn't is  $1/3$ .

Having worked all this out, we see that

$$\Pr[B] = \frac{1}{4} \cdot 1 + \frac{3}{4} \cdot \frac{2}{3} = \frac{3}{4}.$$

On the other hand,

$$\Pr[B|A] = \frac{2}{3},$$

since it is the probability of winning on the  $G'$  pictured above. (We could also work this out from the definition:  $\Pr[B|A] = \Pr[B \wedge A] / \Pr[A] = \frac{(3/4) \cdot (2/3)}{3/4} = 2/3$ ). In particular, the two are not the same, so  $A$  and  $B$  are not independent.

3. If we repeat  $100n$  times, the failure probability is

$$(1 - 1/(2n))^{100n} \leq e^{-50},$$

which is tiny.

4. The total number of edge contractions is  $O(n^{5/2} \log n)$ . This is because there are  $O(n)$  contractions to reduce  $G$  to  $G'$ , and  $O(km) = O(n^{3/2} \log n)$  to repeat Karger  $k$  times on  $G'$ . The second part dominates, and then we repeat all of this  $\Theta(n)$  times, to get  $O(n^{5/2} \log n)$ . This is better than  $O(n^3)$ !
5. Check out [Karger, Stein 1996] (linked from the course website) to see one solution! Check out [Karger 1998] (also linked from website) for another algorithm, also based on random contractions, that gets near-linear time!

The state-of-the-art deterministic algorithm is (I believe) [Kawarabayashi, Thorup, 2015]. They get a near-linear time deterministic algorithm!