3.6.2 Min-cut in improved time $O(n^2 \log^2 n)$

Key to speeding up the min-cut procedure is the observation that, fixing a min-cut $(S, \bar{S})$, since its edges form only a small fraction of the overall weight, we are unlikely to choose any of its edges until late in the contraction process. So instead of running $O(n^2)$ *independent* trials, we'll use a branching procedure, so that we duplicate our search only when needed before the probability of success goes down too much.

Algorithm 5: KargerStein(G): Karger-Stein Min Cut Algorithm [28]

Input: Undirected, simple graph $G = (V, w)$ with edge-weights w
Output: A cut
if $|V| \leq 6$ **then**
  Find a min-cut by examining all cuts
else
  $n' := \lceil \frac{|V|}{\sqrt{2}} + 1 \rceil$
  Cut_1 :=KargerStein(Karger(G, n'))
  Cut_2 :=KargerStein(Karger(G, n'))
  Return min$\{\text{Cut}_1, \text{Cut}_2\}$

The first order of business is, since we are sprouting so many processes, what is the space complexity? Actually this is easy—the tree is examined DFS, has levels indexed by $i = \log_{\sqrt{2}} n$ from 3 (the bottom) through $2 \lg n$; at any given time the program is holding, for each level $3 \leq i \leq 2 \lg n$, (a) a solution (a cut) to one problem of this size, and (b) another graph of this size. Put together these occupy space $n^2/2^i$. Overall this is a geometric series so the total space complexity is $O(n^2)$.

Next, what is the runtime? We already saw that each call to the Karger routine takes time $O(n^2)$. So we have the recurrence

$$T_{KS}(n) \in \Theta(n^2) + 2T_{KS}(n/\sqrt{2})$$

This is one of those "critical" cases in the constant-coefficients recurrence theorem, so we have $T_{KS}(n) \in \Theta(n^2 \log n)$.

The third order of business is, what is the success probability? This is easier solved in terms of i, so write it as p_i (as a lower bound for all $2^{i/2} \leq n < 2^{(i+1)/2}$), and now notice that success precisely corresponds to percolation out to depth $2 \lg n$ in the process where a parent-child edge exists if the min-cut of the child is the same as (rather than larger than) the min-cut of its parent.

In this particular problem, we haven't tried to make p_i tend to a fixed limit—that would require more branching, and make the algorithm sub-optimal in runtime. Instead we settled for a very slowly decreasing probability of success.

Claim: $p_i \geq \frac{7/2}{i+4}$. Proof: First note this holds for $i = 3$ which is the largest base-case we handle deterministically. Now, applying the branching-process interpretation and plugging in (3.23) gives (where $n_i = 2^{i/2}$)

$$p_{i+1} \geq 1 - \left(1 - \frac{n_{i+1}(n_{i+1} - 1)}{n_i(n_i - 1)} p_i\right)^2 \geq 1 - (1 - p_i/2)^2 = p_i - p_i^2/4$$

Applying the inductive hypothesis (and that $p_i - p_i^2/4$ is monotone increasing in $[0,1]$),

$$\begin{aligned} p_{i+1} &\geq \frac{7/2}{i+4} - \frac{49/16}{(i+4)^2} \\ &= \frac{7/2}{i+5}\frac{i+5}{i+4}\left(1 - \frac{7/8}{i+4}\right) \\ &= \frac{7/2}{i+5}\left(1 + \frac{i/8 - 3/8}{(i+4)^2}\right) \\ &\geq \frac{7/2}{i+5} \quad \text{for } i \geq 3 \end{aligned}$$

Corollary 34. *By running Karger-Stein $O(\log n)$ times, we can output a min-cut with probability at least $1/2$, in total runtime $O(n^2 \log^2 n)$.*

3.7 Network reliability problem

Given a simple (no loops or multiple edges) undirected, unweighted, connected graph G. (Actually, it's fine if the graph is not connected, but the problem is trivial then.) Each edge fails independently with probability p. Define

$$\text{FAIL}(p) = \Pr[G \text{ is disconnected}].$$

Our problem is to compute FAIL(p). This is a #P-complete problem. So we aim for a multiplicative approximation.

In the regime that the network G is fairly unreliable, specifically, $\text{FAIL}(p) \in n^{-O(1)}$, then this problem is easy: just simulate and test. (That will indeed be the "easy case" of the algorithm we present below.) The challenging case is when the network is actually pretty reliable. Then we are trying to estimate the probability of a rare failure event. This is, throughout engineering, a difficult problem.

For network reliability, there is an elegant FPRAS (recall definition 29) due to Karger, [2] which we'll now see.

[2] And see [27] for an in some ways improved algorithm.

Also, recently, Guo and Jerrum [22] obtained a FPRAS for the quantity $1 - \text{FAIL}(p)$. That is, they can multiplicatively approximate the probability that the network is connected, even when that probability is very small.