Randomized Algorithms

Quicksort

Describe quicksort - if we pick a correct pivot elements y (that splits one half into at most $3n/4$ elements), this yields a running time of $\mathcal{O}(n \lg n)$.
It isnt worth it to find such a y, instead pick y uniformly at random.
Number of comparisons is the dominant cost. Let S_1 be the smallest element and S_n the largest. Define X_{ij}. Expected cost is then:

$$E[\sum_{i=1}^{n}\sum_{j>i} X_{ij}] = \sum_{i=1}^{n}\sum_{j>i} E[X_{ij}]$$

Now $E[X_{ij}] = p_{ij} \cdot 1 + (1 - p_{ij}) \cdot 0$.
Two elements i and j are only compared if an element $i < l < j$ is picked before either i or j as pivot, so

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j>i} p_{ij} &= \sum_{i=1}^{n}\sum_{j>i} \frac{2}{j-i+1} \\ &\leq \sum_{i=1}^{n}\sum_{k}^{n-i+1} \frac{2}{k} \\ &\leq 2\sum_{i=1}^{n}\sum_{k}^{n} \frac{1}{k}\end{aligned}$$

So we have an upper bound on our expected number of comparisons - $2nH_n$.
Furthermore, $H_n \approx \ln n + \Theta(1)$, which gives us an expected running time of $\mathcal{O}(n \lg n)$. This is regardless of input - compare with normal quicksort.

This was a *Las Vegas* algorithm as it is deterministic. Now to a *Monte Carlo* algorithm, which is not determenistic, but where we can bound the probability it is incorrect.

Min-Cut algorithm

An algorithm to find a min-cut in an undirected graph G is to pick an edge and contract it. We keep doing this until only two vertices remain and the number of edges between should supposedly be a min-cut. It is not always true, but we can bound the probability it is incorrect.

Let C be a min-cut of size k. The graph has at least $kn/2$ edges. Thus the probability of picking an edge belonging to the min-cut C is $k/(nk/2) = 2/n$ or $1 - 2/n$ for not picking an edge belonging to the min-cut. When we pick the next edge, it has probability of $1 - 2/(n-1)$ for not belonging to C. Generally for all $n-2$ iteration, there is a probability of $1-2/(n-i+1)$ for not picking an edge belonging to C.
We write this probability as $P[\cap_{i=1}^{n-2}\xi_i] \geq \prod_{i}^{n-2}(1 - \frac{2}{n-i+1}) = \frac{2}{n(n-1)}$.

This is not very big, but we can run the algortihm many times (picking random edges each time) to minimize the probability that the algorithms fails smaller. Good when no efficient algorithm exists.