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 deterministic, 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. Good when no efficient algorithm exists.

Binary Planar Partitions

Binary Planar Partition is the problem of dividing disjoint line segments into regions, so that each region contains only one line segment. This can be represented as binary tree. This has a number of applications in computer graphics. We will look at `RandAuto`, which works by picking a random permutation of the line segments at extending them infinitely in both directions, until no regions contain more than one segment. The expected size of the autopartition is $\mathcal{O}(n \lg n)$.
Proof: For vertices u and v, let $index(u, v) = i$ be the number of line segments $l(u)$ cuts before cutting v (including) and let it be ∞ if it does not cut v. Let $u \dashv v$ be the events that u cuts v which happens only if none of $u_1, \ldots, u_{i-1}, v$ is picked first. This happens with probability $1/(i+1)$. Let $C_{u,v}$ be the

indicator variable for this. We get that

$$\begin{aligned} E[P_\pi] &= n + \sum_u \sum_{v \neq u} E[C_{u,v}] \\ &= n + \sum_u \sum_{v \neq u} P[u \dashv v] \\ &\leq n + \sum_u \sum_{v \neq u} \frac{1}{index(u,v)+1} \end{aligned}$$

We remember line segments are extended in both directions, meaning it will meet other line segments at most once, implying:

$$\begin{aligned} n + \sum_u \sum_{v \neq u} \frac{1}{index(u,v)+1} &\leq n + 2\sum_u \sum_{i=1}^{n-1} \frac{1}{i+1} \\ &\leq n + 2nH_n \\ &= \mathcal{O}(n \lg n) \end{aligned}$$

Pick two