CS 388R: Randomized Algorithms, Fall 2019 August 29th, 2019

Lecture 1: Introduction to randomized algorithms; min-cut

Prof. Eric Price *Scribe: Tongzheng Ren, Garrett Bingham*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Randomized Algorithms

Examples of Randomized Algorithms:

- Primality Testing
- Quick Sort
- Factoring
- Hash tables

Benefits of randomized algorithms:

- Speed
- Simplicity
- Some things only possible with randomization

Keep in mind that randomness is over the choices of algorithms, not the choices of input.

Key techniques of randomized algorithms:

- Avoiding adversarial inputs
  - For example, how should one choose the pivot in quicksort? One way is to always choose the first element, but in the adversarial case, this results in $O(n^2)$ time.
  - In the case of hashing, we might use some modulo function. While it may work well in some cases, for structured input there will likely be many collisions.
- Fingerprinting: compare short, random description of items
- Random sampling
- Load balancing
- Symmetry breaking
- Probabilistic existence proofs

Types of randomized algorithms:

- **Las Vegas**: always correct, but the running time is random
- **Monte Carlo**: running time is fixed, but the algorithm is only correct with high probability

Las Vegas style algorithms can be converted to Monte Carlo algorithms by designating a fixed stopping time T. Monte Carlo algorithms cannot in general be made into Las Vegas algorithms.

2 Quick Sort

Algorithm 1 `QuickSort`(X)

Input: List X
Choose random pivot $t \in$ `range(len(`X`))`
return `QuickSort`$([X_i | X_i < X_t]) + [X_t] +$ `QuickSort`$([X_i | X_i > X_t])$

Expected running time

Define Z_{ij} := number of times the ith smallest element and jth smallest element are compared $\in \{0,1\}$.

$$\text{Time} = O(\text{total comparisons}) = O\left(\sum_{i<j} Z_{i,j}\right)$$

Notice that:

$$\mathbb{P}[Z_{i,j} = 1] = \frac{2}{j-i+1}$$

This is because the probability the ith and jth elements are compared is equal to the probability that either the ith or jth element is chosen as a pivot before any of the $i+1, \ldots, j-1$ elements are.

Next, we have

$$\mathbb{E}[\text{Time}] \lesssim \mathbb{E}\left[\sum_{i<j} Z_{i,j}\right] = \sum_{i<j} \frac{2}{j-i+1} = 2\sum_{i<j} \frac{1}{j-i+1} = 2\sum_{i} \frac{1}{n+1-i} \leq 2n \sum_{i=2}^{n} \frac{1}{i} \leq 2n \log n$$

where $f \lesssim g$ means $\exists C$ constant that $f \leq Cg$. Notice that $\sum_{i=2}^{n} \frac{1}{i}$ is the harmonic series.

3 Karger's min-cut algorithm [Kar93]

Min-cut definition: Given some graph $G = (V, E)$ with n vertices and m edges, a global min-cut is a set $S \subset V : 1 \leq |S| \leq n-1$ that minimizes the number of edges going from S to $\overline{S}$ (the vertices not in S). We define the cut-value of S as the number of edges from S to $\overline{S}$, denoted $\mathbb{E}(S, \overline{S})$

Possible approaches include some traditional deterministic algorithms like the Ford–Fulkerson method with the max-flow min-cut theorem, etc. We will discuss faster algorithms.

Algorithm 2 Karger's min-cut algorithm

Input: Graph $G = (V, E)$ with n vertices and m edges
while $n > 2$ **do**
  Contract a random edge $e(u, v)$: merge the vertices and remove self-loops
end while
return Preimage of the two remaining vertices

Here we allow for multiplicity (there can be multiple edges between one pair of vertices). See here for a single run of Karger's min-cut algorithm.

Lemma 1. *Algorithm 2 succeeds with probability larger than* $\frac{2}{n^2}$.

Proof. Let $d(u)$ denote the degree of vertex u.

$$\mathbb{P}[\text{fail in the first step}] = \frac{\mathbb{E}(S, \overline{S})}{n} \leq \frac{\min d(u)}{m} \leq \frac{\frac{1}{n}\sum d(u)}{m} = \frac{2}{n}$$

Similarly,

$$\mathbb{P}[\text{fail in the } i\text{-th step}|\text{succeed in the } i-1\text{-th step}] \leq \frac{2}{n-i}$$

Thus:

$$\mathbb{P}[\text{succeed in the all of steps}] \geq \prod_{i=1}^{n-2}\left(1 - \frac{2}{n+1-i}\right) = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdots \frac{2}{4} \cdot \frac{1}{3} = \frac{2}{n(n-1)} \geq \frac{2}{n^2}$$

□

When n is large, this guarantee is poor. However, if we repeat n^2 times and return the best result, then the failure probability becomes

$$\left(1 - \frac{2}{n^2}\right)^{n^2} \approx \frac{1}{e^2} > \frac{2}{3}$$

The time complexity is $n^2 m\alpha(n) = n^2(m \log_{m/n} n)$ by Union-Find/Disjoint-set data structure whose time complexity is $O(\alpha(n))$.

4 Karger-Stein faster min-cut algorithm [KS96]

Intuition Most of the work is done at the beginning when there is a low chance of failure.

The running time is:

$$T(n) = 2\left(T\left(\frac{n}{\sqrt{2}}\right) + O\left(n^2\right)\right) = O(n^2 \log n)$$

Algorithm 3 Karger-Stein min-cut algorithm

Input: Graph $G = (V, E)$ with n vertices and m edges
for i=1, 2 **do**
  Run Algorithm 2 for $\frac{n}{\sqrt{2}}$ steps
  Recursively run Algorithm 3
end for
return Better of the two results

since the depth of the search is $O(\log n)$ and each step takes $O(n^2)$ time.

Let $\mathbb{P}(n)$ denote the success probability, then

$$\begin{aligned}\mathbb{P}(n) =& 1 - (1 - \text{chance one branch succeeds})^2 \qquad \text{i.e. } \mathbb{P}\left(\frac{n}{\sqrt{2}}\right) \text{ by definition} \\ =& 1 - \left(1 - \frac{1}{2}\mathbb{P}\left(\frac{n}{\sqrt{2}}\right)\right)^2 \\ =& \mathbb{P}\left(\frac{n}{\sqrt{2}}\right) - \frac{1}{4}\mathbb{P}\left(\frac{n}{\sqrt{2}}\right)^2\end{aligned}$$

We can find that $\mathbb{P}(n) = \Theta(\frac{1}{\log n})$. To show this, let $x = \log_{\sqrt{2}} n$ and $f(x) = \mathbb{P}(2^{\frac{x}{2}})$. Then

$$f(x) = f(x-1) - \frac{1}{4}f(x-1)^2$$

We can find the solution $f(x) = \frac{4}{x}$, thus $\mathbb{P}(n) = \Theta(\frac{1}{\log n})$. Also see [KS96] for another approach.

If we repeat Algorithm 3 $O(\log n)$ times, we get $O(n^2 \log^2 n)$ time with constant probability of success. To see this, we consider the success probability:

$$1 - (1 - \mathbb{P}(n))^{\log n} = \Theta(1) + O\left(\frac{1}{\log n}\right)$$

is some constant. This method outperforms the $O(mn^2 \log n)$ time complexity approach mentioned earlier, as in practice m can be on the order of $O(n^2)$.

References

[Kar93] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, pages 21–30, 1993.

[KS96] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.