COMPSCI 330: Design and Analysis of Algorithms 3/22/2016

Las Vegas and Monte Carlo Algorithms

Lecturer: Debmalya Panigrahi *Scribe: Tianqi Song*

1 Overview

In this lecture, first we cover the *birthday paradox*. Then, we will introduce *Monte Carlo* and *Las Vegas* algorithms. [1]

2 The Birthday Paradox

2.1 Problem Statement

There are k people and each person has a random birthday from n days. How large should k be such that the chance that two people have the same birthday is at least α?

2.2 Analysis

Let X be the event that every person has a distinct birthday. The ith person has $(n-i)$ options to have a distinct birthday and then the probability to have a distinct birthday is $\frac{n-i}{n}$, where $i \in \{0,1,2,...,k-1\}$. We have:

$$Pr[X] = \frac{n}{n}(\frac{n-1}{n})(\frac{n-2}{n})\dots(\frac{n-(k-1)}{n}) \tag{1}$$

$$= 1(1-\frac{1}{n})(1-\frac{2}{n})\dots(1-\frac{k-1}{n}) \tag{2}$$

Because $e^{-x} \approx 1-x$ when x is small:

$$Pr[X] \approx 1 \cdot e^{-\frac{1}{n}} \cdot e^{-\frac{2}{n}} \dots e^{-\frac{k-1}{n}} \tag{3}$$

$$\approx e^{-\frac{(1+(k-1))(k-1)}{\frac{2}{n}}} \tag{4}$$

$$= e^{-\frac{k(k-1)}{2n}} \tag{5}$$

We just need to solve $Pr[X] \approx e^{-\frac{k(k-1)}{2n}} \leq 1-\alpha$. For example, when $\alpha = 0.5$ and $n = 365$, we have $k \geq 23$.

3 Monte Carlo Algorithms

Definition 1. *A randomized algorithm is called a **Monte Carlo** algorithm if it may fail or return incorrect answers, but has runtime independent of the randomness.*

[1] Some materials are from a note by Samuel Haney for this class in Fall 2014 and a note by Allen Xiao for COMPSCI 532 in Fall 2015.

3.1 Global Minimum Cut

Definition 2. *Let $G = (V, E)$ be an undirected, unit-capacity graph, where $|E| = m$ and $|V| = n$. The **global minimum cut** problem is to find the smallest non-trivial cut in G. In other words, a complete partitioning of V into (S, T) which minimizes:*

$$|\{(v, w) \mid (v, w) \in E \cap (S \times T)\}|$$

where both S and T are nonempty.

3.1.1 Karger 1993

A Monte Carlo algorithm by Karger in 1993 gives the global min-cut, with high probability, in $O(m)$ time. This algorithm is based around *edge contractions*:

Definition 3. *Let $G = (V, E)$ be a (multi)graph, and $(u, v) \in E$. **Contracting** edge (u, v) produces a new multigraph where:*

1. *Both u and v are replaced by a single new supervertex $w = \{u, v\}$.*
2. *Each edge between u and v is replaced by a self-loop (w, w). Edges which have u or v as an endpoint now have w as that endpoint.*

The resulting graph may include multi-edges and self-loops.

Edge contractions can simplify the graph while preserving a cut (S, T).

Fact 1. *Let (S, T) be a cut in G. Let (v, w) be an edge where both $v, w \in S$ or both $v, w \in T$. Then if G' is the graph after contracting (v, w), then $|(S, T)|$ is the same in G' as in G.*

Lemma 2. *Let G' be the graph after contracting e in G. Every cut in G' corresponds a cut of the same size in G, on the same supervertices.*

Proof. This follows directly from Fact 1. Note, however, that the opposite direction does not hold: not every cut in G' reappears as a cut in G. In particular, the cuts in G which included e disappear in G'. □

Contraction preserves the size for cuts which don't use the contracted edge. Intuitively, contraction doesn't *produce* smaller cuts in G', which means the minimum cut remains the minimum cut.

Corollary 3. *Let (S, T) be the global minimum cut of G, and let G' be the graph after contracting $e \notin (S, T)$. Then (S, T) is the global minimum cut of G', and has the same size.*

The algorithm is as follows:

Algorithm 1 (Karger 1993)

```
G_n ← G
for i = (n − 1) to 2 do
    G_i ← G_{i+1}
    Contract e ∈ E_i, picked uniformly at random, and remove all self-loops.
return Cut represented by two remaining supervertices of G_2.
```

3.1.2 Success Probability

First, we relate the size minimum cut to the number of edges:

Lemma 4. *Let λ be the size of the minimum cut. On a graph with i vertices, $|E| \geq (i\lambda)/2$.*

Proof. Let $\deg(v)$ be the degree of v. Summing over all vertices double counts every edge.

$$\sum_{v \in V} \deg(v) = 2|E|$$

The degree cut of any vertex gives an upper bound on the size of the minimum cut.

$$\sum_{v \in V} \deg(v) \geq i\lambda$$

Rearranging:

$$|E| \geq \frac{i\lambda}{2}$$

□

Theorem 5. *For any given global mincut $(S, T = V \setminus S)$, Karger's algorithm outputs (S,T) with probability $\Omega\left(1/n^2\right)$.*

Proof. For any cut to be output by the algorithm, none of its edges must be contracted before the end (G_2). Let λ be the number of edges in global min-cut (S,T). Let $G_i = (V_i, E_i)$ be the graph at i vertices.

(S,T) disappears in G_i if one of its edges are contracted. Let $\mathscr{E}_i$ be the event that no (S,T) edge is contracted in the contraction on G_i to form G_{i-1}.

$$\begin{aligned}
\Pr\left((S,T) \text{ successfully output}\right) &= \Pr(\mathscr{E}_n \cap \mathscr{E}_{n-1} \cap \cdots \cap \mathscr{E}_2) \\
&= \Pr(\mathscr{E}_n) \Pr(\mathscr{E}_{n-1} \mid \mathscr{E}_n) \Pr(\mathscr{E}_{n-2} \mid \mathscr{E}_n, \mathscr{E}_{n-1}) \cdots
\end{aligned}$$

Lemma 4 tells us that:

$$|E_i| \geq \frac{i\lambda}{2}$$

Since the edges are chosen uniformly from E_i:

$$\begin{aligned}
\Pr\left(\mathscr{E}_i \mid \cap_{k=n}^{i+1} \mathscr{E}_k\right) &= 1 - \frac{\lambda}{|E_i|} \\
&\geq 1 - \frac{\lambda}{(i\lambda)/2} \\
&= 1 - \frac{2}{i}
\end{aligned}$$

Replacing:

$$\begin{aligned}
\Pr\left((S,T) \text{ successfully output}\right) &= \Pr(\mathscr{E}_n) \Pr(\mathscr{E}_{n-1} \mid \mathscr{E}_n) \Pr(\mathscr{E}_{n-1} \cap \mathscr{E}_n \mathscr{E}_{n-1}) \cdots \\
&\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{3}\right) \\
&= \frac{n-2}{n} \times \frac{n-3}{n-1} \times \frac{n-4}{n-2} \times \cdots \times \frac{3}{5} \times \frac{2}{4} \times \frac{1}{3} \\
&= \frac{2 \cdot 1}{n(n-1)} \\
&= \binom{n}{2}^{-1} = \Theta(1/n^2)
\end{aligned}$$

□

Corollary 6. *There are at most* $\binom{n}{2}$ *unique minimum cuts in a graph.*

How can we obtain a high probability bound? Repeat the algorithm $O(n^2 \log n)$ times and output the smallest cut.

$$\begin{aligned}
\Pr(\text{None of the outputs were global minimum cut}) &\leq \prod_{t=1}^{n^2 c \log n} \left(1 - \frac{1}{n^2}\right) \\
&\leq \left(e^{-1/n^2}\right)^{n^2 c \log n} \\
&= O(1/n^c) \\
\Pr(\text{At least one output was global minimum cut}) &= \Omega(1 - 1/poly(n))
\end{aligned}$$

3.1.3 Running Time

The contraction algorithm we described can be implemented in $O(mn^2 \log n)$ time.

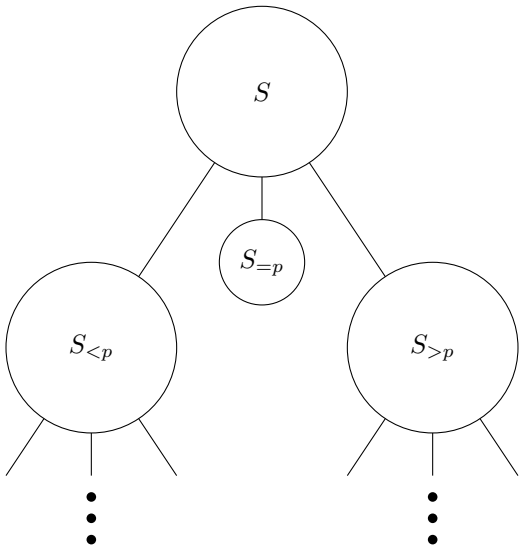
4 Las Vegas Algorithms

Definition 4. *A randomized algorithm is called a* ***Las Vegas*** *algorithm if it always returns the correct answer, but its runtime bounds hold only in expectation.*

In Las Vegas algorithms, runtime is at the mercy of randomness, but the algorithm always succeeds in giving a correct answer.

4.1 Randomized Quicksort Analysis

Recall that the randomized quicksort algorithm picks a pivot at random, and then partitions the elements into three sets: all the elements less than the pivot, all elements equal to the pivot, and all elements greater than the pivot.



We analyze the runtime using charging. In the computation tree shown above, we count the number times each element appears. The sum over all elements is equal to the sum of the sizes of all the sets in the computation tree, which is proportional to the runtime.

Claim 1. *The number of times an element appears in sets in the computation tree is $O(\log n)$ in expectation.*

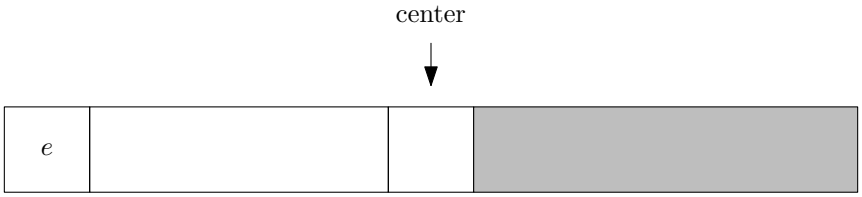
Each set S in the computation tree has three children. Let $S_{=p}$ be child set with elements equal to the pivot. Let $S_{>p}$ and $S_{<p}$ be defined similarly. We color the tree edges to these children as follows:

- Color the edge to $S_{=p}$ red.
- Color the edge to the larger of $S_{>p}$ and $S_{<p}$ black.
- Color the edge to the smaller of $S_{>p}$ and $S_{<p}$ blue.

Now, we trace the path of some element e through the computation tree, and count the number of edges of each color along the path.

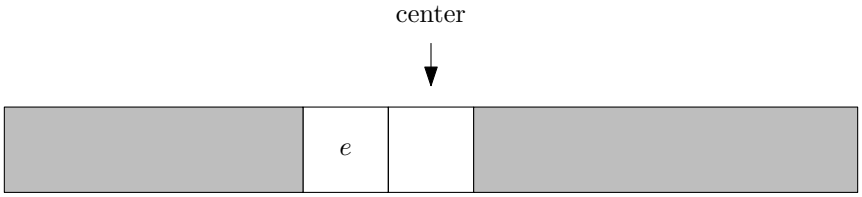
- The number of red edges is 1. Only the last edge can be red.
- The number of blue edges is at most $\log_2 n$, since the smaller of $S_{>p}$ and $S_{<p}$ has at most $\frac{1}{2} \cdot |S|$ elements.
- The number of black edges, using the definitions we have given, could be large.

What is the probability that an edge is black? That is, given that $e \in S$, what is the probability that the next edge in e's path is black? Suppose e is the smallest element in the array.



For this element, any pivot in the right half of the array, illustrated with the shaded region in the image above, will cause e's next edge to black. Therefore, the probability that the next edge is black in this case is roughly $\frac{1}{2}$.

However, consider some element e near the center of the array.



The shaded region shows which pivots will cause the next edge in e's path to be black. For this element, the probability that the next edge is black is nearly 1. The probability will be high for all elements near the center of the array.

We need to modify our definition of black edges and blue edges to fix the problem. If $|S_{>p}| > \frac{3}{4}|S|$, color the edge to $S_{>p}$ black. Otherwise, color the edge to $S_{>p}$ blue. We color the edge to $S_{<p}$ using identical rules. Note that now it is possible for both edges to be blue (but not both to be black). We have the following:

- The number of red edges is still 1.
- The number of blue edges is at most $\log_{4/3} n$.

To calculate the probability of an edge being black, we split the array into four equal parts:

If the pivot is selected from the shaded quadrants, one of the edges will be blue and the other will be black. If the pivot is selected from the non-shaded quadrants, both edges will be blue. Therefore, regardless of which element $e \in S$ we pick, the probability that the next edge on e's path is black is at most $\frac{1}{2}$.

Now, we can bound the number of black edges in a path. Let X_i a random variable, equal to the number of black edges between the ith and the $(i+1)$st blue edge. As we have seen,

$$\mathbb{E}[X_i] \leq \frac{1}{p_{success}} \leq \frac{1}{1/2} = 2.$$

By linearity of expectation, we have

$$\mathbb{E}[\text{total number of black edges}] = \mathbb{E}\left[\sum_i X_i\right] = \sum_i \mathbb{E}[X_i] \leq 2\left(\log_{4/3} n\right) = O(\log n).$$

We now need to bound the total running time, T. Let T_e be the amount of running time charged to e (which is proportional to the number of sets e appears in).

$$\mathbb{E}[T] = \mathbb{E}\left[\sum_e T_e\right] = \sum_e \mathbb{E}[T_e] = \sum_e O(\log n) = O(n \log n).$$