CS 388R: Randomized Algorithms, Fall 2021 September 2nd, 2021

Lecture 3: Quick Sort Analysis

Prof. Eric Price *Scribe: Rochan Avlur, Alex Witt*

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

1 Overview

In the last lecture, we discussed the coin flip problem and through examples, we covered Additive and Multiplicative Chernoff bounds, Benett's Inequality and Gaussian approximations for the same.

In this lecture, we analyze Quick-Sort algorithm for both cases, when the pivot is chosen deterministically and at random. We discuss the implications of using Chernoff bounds on non-independent distributions and show how Chernoff bounds can still be applied when certain conditions are met by the distributions.

2 Quick-Sort

Quick-sort is a divide-and-conquer technique for sorting elements in a list. Quick-sort is widely used as it is generally fast and in-place. However, it is an unstable sorting algorithm, meaning the relative order of two equal values in the input list is not preserved after the algorithm terminates. Given a list $\mathbf{L}$ with n elements indexed as $\mathbf{L}[i]$ for $i \in \{1, \dots, n\}$, Algorithm 1 and 2 illustrate the pseudocode for quick-sort based on how the pivot is chosen.

Time Complexity. Swaps in Quick-Sort have $\mathcal{O}(n)$ complexity and the number of nested calls (in *best* and *average* case) is $\mathcal{O}(\log n)$ giving us a complexity of $\mathcal{O}(n \log n)$. If the pivot is always chosen as the first element and the input array is already sorted, this is the *worst* case for the deterministic algorithm yielding $\mathcal{O}(n^2)$.

Algorithm 1 Quick-sort w/ deterministic pivot choice

```
procedure detQuicksort($\mathbf{L}$)
    $s$ = sizeof($\mathbf{L}$)
    if $s = 0$ then return []
    $x = \mathbf{L}[0]$
    return Quicksort([$y$ for $y \in \mathbf{L}[1:]$ if $y \leq x$])
          +$\mathbf{L}[x]$
          +Quicksort([$y$ for $y \in \mathbf{L}[1:]$ if $y > x$])
```

Note: While comparing tuples, we compare the first element and then the second element.

Algorithm 2 Quick-sort w/ random pivot choice

```
procedure randQuicksort($\mathbf{L}$)
    $s = \text{sizeof}(\mathbf{L})$
    if $s = 0$ then return []
    $x \in \{1, \ldots, s\}$ at random
    return Quicksort([$y$ for $j, y \in$ enumerate($\mathbf{L}$) if$(y, j) < (\mathbf{L}[x], x)$])
            +$\mathbf{L}[x]$
            +Quicksort([$y$ for $j, y \in$ enumerate($\mathbf{L}$) if$(y, j) > (\mathbf{L}[x], x)$])
```

3 Expected Running Time

Let T represent the total running time of randquicksort. How do we arrive at the expected running time $\mathbb{E}(T)$?

Definition. To simplify notation, we introduce two symbols, $\simeq$ and $\lesssim$. We define them as,

$$F \lesssim g \text{ if } \exists\, c > 0 \text{ s.t } F \leq c \cdot g \tag{1}$$

and,

$$F \simeq g \text{ if } F \lesssim g \text{ and } g \lesssim F \tag{2}$$

Solution. We begin with analyzing the *swap* procedure in randquicksort. Assume the values of a list $\mathbf{L}$ of length n are $\{1, \ldots, n\}$. Let $E_{i,j}$ be the event when $\mathbf{L}[i]$ and $\mathbf{L}[j]$ (elements at indices i & j) are compared.

Our total running time can be approximated as,

$$T \simeq \sum_{i<j} E_{i,j} \tag{3}$$

where,

$$E_{i,j} = \begin{cases} 1, \mathbf{L}[i] \text{ \& } \mathbf{L}[j] \text{ are compared} \\ 0, \mathbf{L}[i] \text{ \& } \mathbf{L}[j] \text{ are } \textbf{not} \text{ compared} \end{cases} \tag{4}$$

Let us analyze the scenarios that lead to $E_{i,j} = 1$. One possibility is when either $\mathbf{L}[i]$ or $\mathbf{L}[j]$ is a pivot. Since the pivot is compared with all values in the partition it belongs to, $E_{i,j} = 1$. The other case is when both $\mathbf{L}[i]$ and $\mathbf{L}[j]$ are on the same side of the pivot [1]. Together, the condition for $E_{i,j} = 1$ is when $\mathbf{L}[i]$ and $\mathbf{L}[i]$ are pivots before the first pivot at index $i + 1, i + 2, \ldots, j - 2, j - 1$.

Since we are equally likely to choose index i or j as the pivot, the probability of either being chosen is,

$$Pr[E_{i,j}] = \frac{2}{j - i + 1} \tag{5}$$

[1] Revisit. Why?

Expectation of running time is,

$$\begin{aligned}
\mathbb{E}[T] &\lesssim \sum \mathbb{E}[E_{i,j}] && (6)\\
&= \sum_{i<j} \frac{2}{j-i+1} && (7)\\
&= \sum_{i=1}^{n} 2 \cdot \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+1} \right) && (8)\\
&\leq 2n \log n && (9)
\end{aligned}$$

with the help of harmonic progression equality,

$$H_{n-i+1} - 1 = \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i+1} \right)$$

and inequality,

$$H_{n-i+1} - 1 \leq H_n \leq \log n$$

3.1 How do we calculate the probability that quick-sort algorithm works in $\mathcal{O}(n \log n)$ time?

Say for instance, we want to calculate,

$$Pr\left[\sum E_{i,j} > 10 \cdot n \log n \right] \tag{10}$$

Here, we cannot use Chernoff bounds since the random variable's events are not independent. In quick-sort, $E_{i,j}$ are not independent since comparing values at index i and j depends on the pivot used to partition the .

Let us use Markov's inequality instead to find an upper bound on the probability.

3.2 Markov's Inequality

Let $X \geq 0$. For some $t > 0$,

$$\mathbb{E}[X] \geq Pr[X \geq t] \cdot t \tag{11}$$

More commonly, the above equation is written as,

$$Pr[x \geq t] \leq \frac{\mathbb{E}[X]}{t} \tag{12}$$

Applying Markov's inequality to compute the probability of Equation 10, we get,

$$Pr\left[\sum E_{i,j} > 10 \cdot n \log n \right] \leq \frac{2n \cdot \log n}{10n \cdot \log n} = \frac{1}{5} \tag{13}$$

How do we show $T = \mathcal{O}(n \log n)$ with high probability (w.h.p)? Or, in other words, how do we show that $\forall c > 0, T = \mathcal{O}_c(n \log n)$ with probability $1 - n^{-c}$?

In QuickSort, the work done per layers is $\mathcal{O}(n)$. Hence, we want to show w.h.p that the number of layers is $\leq \log n$.

Lemma 1. *For any fixed x, number of layers till x is a pivot is $\mathcal{O}(\log n)$ with high probability.*

Let S_i be the set of elements in the ith layer in the list containing x. $S_i = \{1, \dots, n\}$ and we want $|S_{\mathcal{O}(\log n)}| = 1$.

We define an iteration of QuickSort as good if the pivot is chosen somewhere between 25% and 75% of the length of S_i or if $|S_i| = 1$. Let $X_i = 1$ represent whether or not the ith iteration of the algorithm is good. Since there is a chance S_i contains one element, it follows

$$Pr[X_i] \geq \frac{1}{2} \tag{14}$$

Given $X_i = 1$, we can also consider the size of the next set S_{i+1}. In the worst case, a pivot could be chosen at 25% or 75% the length of the array. This would result in an $|S_{i+1}| = \frac{3}{4}|S_i|$, but since this is worst case, we have the bound

$$|S_{i+1}| \leq \frac{3}{4}|S_i| \text{ or } |S_{i+1}| = 1 \tag{15}$$

Thus, if we can achieve $\log_{\frac{4}{3}} n$ good rounds, we will end with $|S_k| = 1$. The $Pr[|S_k| > 1]$ can be described with how many good rounds we have achieved versus how many are needed to reach $|S_k| = 1$[2]. Let the number of good rounds X be defined as $X = \sum_{i=1}^{k} X_i$, then

$$Pr[|S_k| > 1] \leq Pr[X < \log_{\frac{4}{3}} n] \tag{16}$$

or the $Pr[|S_k| > 1]$ is less than or equal to getting fewer than $\log_{\frac{4}{3}} n$ good runs in the first k steps.

We can find the $\mathbb{E}[X]$ using the linearity of expectation

$$\mathbb{E}[X] = \sum_{i=1}^{k} \mathbb{E}[X_i] \geq \frac{k}{2} \tag{17}$$

since $\mathbb{E}[X_i] \geq \frac{1}{2}$.

4 Chernoff Bounds Revisited

Assume $X_1, \dots, X_n$ are independent random variables taking values in $\{0, 1\}$ Let $X = \sum X_i$, $\mathbb{E}[X_i] = \mu_i$, and $\mu = \sum \mu_i = \mathbb{E}[X]$. From lecture 2, we know the additive and multiplicative Chernoff bounds are $Pr[X > \mu + t] \leq \exp\left(-\frac{2t^2}{n}\right)$ and $Pr[X > (1+\epsilon)\mu] \leq \exp\left(-\frac{\min(\epsilon,\epsilon^2)\mu}{3}\right)$, respectively.

Suppose $Y_i = 1 - X_i$ represents the ith event in the QuickSort algorithm is bad and does not find a pivot within in the 25% to 75% range. Then $\mathbb{E}[Y_i] \leq \frac{1}{2}$ and $\mathbb{E}[Y] \leq \frac{k}{2}$. Then we would like to place a Chernoff bound on,

$$Pr[Y \geq k - \log_{\frac{4}{3}} n] \tag{18}$$

[2] Revist. Why?

To apply Chernoff bounds, let $k = 10\log_{\frac{4}{3}} n$. Then,

$$\begin{aligned} Pr[Y > 9\log_{\frac{4}{3}} n] = Pr[Y > \mu + 4\log_{\frac{4}{3}} n] &\leq \exp\left(-\frac{2(4\log_{\frac{4}{3}} n)^2}{10\log_{\frac{4}{3}} n}\right) && (19) \\ &= \exp\left(-3.2\log_{\frac{4}{3}} n\right) && (20) \\ &= n^{-3.2\log_{\frac{4}{3}} e} && (21) \\ &= n^{-\mathcal{O}(1)} && (22) \end{aligned}$$

Note: If $Y < Z$ and $Pr[Z > c] < \delta$, then $Pr[Y > c] < \delta$.

4.1 Independence and Chernoff Bounds

We have applied the Chernoff bound to our QuickSort algorithm, but the issue with using Chernoff bounds on our problem is that the random variables $Y_1, \ldots, Y_n$ are not independent.

To continue using Chernoff bounds, we fall back to the corollary $\mathbb{E}[X_i] \leq \mu_i$. We can define Y_i to be dependent on X_i (as $Y_i = 1 - X_i$) and $\mathbb{E}[Y_i] = \mu$ for $Y_i \geq X_i$. Also, note Y_i is independent of any Y_j for $i \neq j$. The key idea is that complete independence among the events of a random variable in Chernoff bound is not necessary. All we need are the events to be independent of the past events i.e, Y_i is independent of Y_j for $i > j$.

Let us define $X_i \in \{0,1\}$ and $\mathbb{E}[X_i | X_1 = a_1, X_2 = a_2, \ldots, X_{i-1} = a_{i-1}] \leq \mu_i \; \forall \; i, a_1, \ldots, a_{i-1}$. We choose $Y_i \in \{0,1\}$ such that $\mathbb{E}[Y_i] = \mu_i$. We want results in $(Y_i | X_1, \ldots, X_{i-1})$ dominating (is larger than) $(X_i | X_1, \ldots, X_{i-1})$.

All Y_i are independent if $(Y_i | Y_1 = a_1, \ldots, Y_{i-1} = a_{i-1})$ same for all a_i.

$$Y_i = \begin{cases} 1, \text{if } X_i = 1 \\ 1, \text{w.p } \frac{\mu - \mathbb{E}[X_i | X_1, \ldots, X_{i-1}]}{\mathbb{E}[X_i | X_1, \ldots, X_{i-1}]} \\ 0, \text{otherwise} \end{cases} \tag{23}$$