COMPSCI 330: Design and Analysis of Algorithms April 4

Lecture 19: Randomized Algorithms

Lecturer: Rong Ge *Scribe: Chenwei Wu*

1 Overview

In this lecture, we will learn the basic knowledge of **Randomized Algorithms** and see some examples. Randomized algorithms are algorithms that make use of random decisions, i.e., if you run a randomized algorithms on the same input for multiple times, then they may give you different results or they give you the same results but the internal steps can be different. The reason why people use randomized algorithms is to avoid the worst cases of algorithms, e.g., if an algorithm runs efficiently for most inputs but much slower for other inputs, a randomized version of this algorithm can "average" the running time and runs efficiently for all inputs. An important note is that randomized algorithms still works for all possible inputs although the algorithm itself is randomized.

2 Basic Probability: Recap

In this section we will briefly review the basic knowledge for probability.

2.1 Random Variable

Here we only give an informal definition of random variable. Also note that a random variable is always written as a capital letter.

Definition 1. X is a **random variable** if its value depends on some random phenomenon. $\Pr[X = v]$ denotes the probability that X is equal to v.

For example, if X is the outcome of a fair coin, let $X = 0$ mean the outcome is tail and $X = 1$ mean head, then

$$X = \begin{cases} 0 & w.p.\frac{1}{2} \\ 1 & w.p.\frac{1}{2} \end{cases} .$$

Similarly, if X is the outcome of a dice, then

$$X = \begin{cases} 1 & w.p.\frac{1}{6} \\ 2 & w.p.\frac{1}{6} \\ 3 & w.p.\frac{1}{6} \\ 4 & w.p.\frac{1}{6} \\ 5 & w.p.\frac{1}{6} \\ 6 & w.p.\frac{1}{6} \end{cases} .$$

Definition 2. Things like $X = v$, $X > v$ and $X < v$ are called **events**. Generally, events can also be something like $X \in S$ where S is a fixed set.

2.2 Joint Probability

Sometimes we need to consider the probability that multiple events happen at the same time, so we need this definition of joint probability.

Definition 3. Joint Probability is the probability that two events happen at the same time, e.g., $\Pr[X = i, Y = j]$.

The following lemma provides a way to compute joint probability:

Lemma 1. If two random variables X and Y are **independent**, then

$$\Pr[X = i, Y = j] = \Pr[X = i] \Pr[Y = j].$$

2.3 Conditional Probability

In analysis of randomized algorithms, we also need to measure the probability of an event condition on another event.

Definition 4. Define $\Pr[X = i | Y = j]$ to be the probability of $X = i$ given that we already know $Y = j$.

Lemma 2. If two random variables X and Y are **independent**, then

$$\Pr[X = i | Y = j] = \Pr[X = i].$$

The following theorem shows the relationship between conditional probabilities and joint probabilities.

Theorem 3. (**Bayes rule**)

$$\begin{aligned} \Pr[X = i | Y = j] &= \frac{\Pr[X = i, Y = j]}{\Pr[Y = j]} \\ &= \frac{\Pr[Y = j | X = i] \Pr[X = i]}{\Pr[Y = j]}. \end{aligned}$$

Note that the second equation can be derived from the first equation: If we apply the first equation for $\Pr[Y = j | X = i]$, we have

$$\Pr[Y = j | X = i] = \frac{\Pr[X = i, Y = j]}{\Pr[X = i]},$$

which is equivalent to

$$\Pr[X = i, Y = j] = \Pr[Y = j | X = i] \Pr[X = i].$$

Plug it into the first equation gives the second equation.

2.4 Expectation

Intuitively, the expectation of a random variable is the "average" value of the random variable after infinite times of sampling. It is rigorously defined as below:

Definition 5.

$$\mathbb{E}[X] = \sum_i \Pr[X = i] \cdot i,$$

where i traverses all possible value of X.

There are many properties of the expectation of a random variable:

Lemma 4. (**Linearity of Expectation**)

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

Note that this lemma is true even if X and Y are not independent. This lemma can help us decompose the expectation of a complicated random variable, e.g., using this lemma and we can get

$$\mathbb{E}[3X + 2Y + Z] = 3\mathbb{E}[X] + 2\mathbb{E}[Y] + \mathbb{E}[Z].$$

2.5 Conditional Expectation

Conditional expectation is the expectations of a random variable given the fact that an event occurs. It is very similar to the expectation except that the probabilities are conditional probabilities.

Definition 6.

$$\mathbb{E}[X|Y = j] = \sum_i \Pr[X = i|Y = j] \cdot i.$$

The following lemma shows the connection between the conditional expectation and the expectation:

Lemma 5. (**Law of Total Expectations**)

$$\begin{aligned}\mathbb{E}[X] &= \sum_j \mathbb{E}[X|Y = j] \Pr[Y = j] \\ &= \mathbb{E}_Y \left[\mathbb{E}[X|Y]\right].\end{aligned}$$

The expression on the second line may look confusing, but actually it is defined as the expression on the first line. In the analysis of a randomized algorithm, intuitively, $\mathbb{E}[X]$ is the expected running time of our algorithm, $Y = j$ is the first random decision in our algorithm, and $\mathbb{E}[X|Y = j]$ is the runtime of our algorithm after fixing the first decision.

3 Two Types of Randomized Algorithms

Randomized algorithms can be classified into two types:

- **Las Vegas Algorithm**
  - Always outputs the correct answer
  - Running time is random
  - Analysis: Compute expected running time
- **Monte Carlo Algorithm**
  - Always runs in a fixed amount of time
  - Result may be incorrect
  - Requirement: Result is correct with probability at least $\frac{2}{3}$(can be replaced with any constant larger than $\frac{1}{2}$).

4 Example: Runtime Analysis for Quick Sort

In this section, let's take Quick Sort as an example to show how to analyze a randomized algorithm.

4.1 Quick Sort Algorithm

We are interested in the average/expected running time of Quick Sort which can be categorized into Las Vegas Algorithm.

Algorithm: It first pick a random pivot number, and then divides the array into two smaller sub-arrays: the numbers smaller and larger than the pivot. Finally, we recursively sort the two sub-arrays.

Example: To sort a list of numbers $a[] = \{4, 2, 8, 6, 3, 1, 7, 5\}$, we first pick a random pivot number (say 3), then partition the array into two sub-arrays: $(\{2, 1\}, \{4, 8, 6, 7, 5\})$. After that, we recursively sort the two parts.

Worst Case: Suppose we are extremely unlucky and always pick the smallest element as the pivot, then the running time of this instance will be $\Theta(n^2)$.

Randomness Helps: Although in extreme cases the running time of Quick Sort can be slow, we are using randomness in our algorithm so that for each input the expected running time is fast. We prove this in the next section by induction.

4.2 Analysis of Quick Sort Algorithm

Theorem 6. The expected running time of Quick Sort Algorithm is $O(n \log n)$.

Proof. Let X_n be the running time of Quick Sort on array of length n, note that X_n is a random variable, and we want to analyze $\mathbb{E}[X_n]$.

For simplicity, assume the elements in the array are distinct. Also, define $pivot = i$ to be the event that the first pivot number is equal to the i-th smallest number in the array. Thus, by **Law of Total Expectations**, we get

$$\mathbb{E}[X_n] = \sum_{i=1}^{n} \mathbb{E}[X_n | pivot = i] \cdot \Pr[pivot = i].$$

After we select the i-th smallest number in the array as the pivot, we will divide the other elements into two sub-arrays, whose sizes are $i-1$ and $n-i$, respectively. The division process costs $O(n)$ time, so we assume that this process can be done in An time, where A is a constant. Therefore,

$$\begin{aligned}\mathbb{E}[X_n | pivot = i] &= \mathbb{E}[X_{i-1} + An + X_{n-i} | pivot = i] \\ &= \mathbb{E}[X_{i-1}] + An + \mathbb{E}[X_{n-i}].\end{aligned}$$

Note that in the second equation we remove the condition in the expectation because the condition $pivot = i$ is independent of the three terms.

Also, since the pivot is selected uniformly at random, we know that

$$\forall i \in \{1, 2, \cdots, n\}, \Pr[pivot = i] = \frac{1}{n}.$$

Thus,

$$\mathbb{E}[X_n] = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbb{E}[X_{i-1}] + An + \mathbb{E}[X_{n-i}]\right).$$

This recursion function is similar to that of merge sort, so our guess is $\mathbb{E}[X_n] = O(n \log n)$, and we will prove this by induction.

Induction Hypothesis: $\forall k \in \mathbb{N}^*, \mathbb{E}[X_k] \le Ck \log_2 k$ (C is chose later).

Base Case: $\mathbb{E}[X_1] = 0$.

Inductive Step: Assume $\mathbb{E}[X_k] \le Ck \log_2 k$ for all $k < n$, then for $\mathbb{E}[X_n]$, we have (For notation simplicity, we also use the fact that $\mathbb{E}[X_0] = 0$, but in induction hypothesis we are not including 0 because $\log_2 0$ is not well-defined.)

$$\begin{aligned}\mathbb{E}[X_n] &= \frac{1}{n} \sum_{i=1}^{n} \left(\mathbb{E}[X_{i-1}] + An + \mathbb{E}[X_{n-i}]\right) \\ &= An + \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[X_{i-1}] + \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[X_{n-i}] \\ &= An + \frac{1}{n} \left(\mathbb{E}[X_0] + \cdots + \mathbb{E}[X_{n-1}]\right) + \frac{1}{n} \left(\mathbb{E}[X_{n-1}] + \cdots + \mathbb{E}[X_0]\right) \\ &= An + 2 \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[X_{i-1}] \\ &\le An + 2 \cdot \frac{1}{n} \sum_{i=2}^{n} C(i-1) \log_2 (i-1).\end{aligned}$$

Note that if $i \leq \frac{n}{2}$, then $\log_2(i-1) \leq \log_2 \frac{n}{2}$ because $\log_2 x$ is an increasing function for x. Similarly, for all $i \leq n$, we have $\log_2(i-1) \leq \log_2 n$. Also note that $\log_2 \frac{n}{2} = \log_2 n - 1$. Therefore,

$$\begin{aligned}
\mathbb{E}[X_n] &= An + 2 \cdot \frac{1}{n} \left(\sum_{i=2}^{\frac{n}{2}} C(i-1) \log_2 \frac{n}{2} + \sum_{i=\frac{n}{2}+1}^{n} C(i-1) \log_2 n \right) \\
&= An + 2 \cdot \frac{1}{n} \left(\sum_{i=2}^{n} C(i-1) \log_2 n - \sum_{i=2}^{\frac{n}{2}} C(i-1) \right) \\
&\leq An + Cn \log_2 n - C \cdot \frac{n}{4}.
\end{aligned}$$

We choose $C = 4A$ and get

$$\mathbb{E}[X_n] \leq Cn \log_2 n.$$

This finishes the induction, i.e., our hypothesis is correct. Therefore, the expected running time of Quick Sort is bounded by $4An \log_2 n = O(n \log n)$. □

5 Example: Runtime Analysis for Quick Selection

In this section, let's see another example of randomized algorithm, which is Quick Selection algorithm.

5.1 Quick Selection Algorithm

The goal of Quick Selection algorithm is to find the k-th smallest element in an array. The idea of this algorithm is very similar to Quick Sort.

Algorithm: It first pick a random pivot number from array a, and then divides the array into two smaller sub-arrays: the left sub-array contains the numbers smaller than the pivot and the right sub-array contains the numbers larger than the pivot. After that, it counts the number of elements in the left sub-array. Assume there are $i-1$ elements in the left sub-array, i.e., the pivot is the i-th smallest element in the original array. If $i > k$, we recurse on the left sub-array, if $i < k$, we recurse on the right sub-array, and if $i = k$, we directly return the pivot. (Note that this algorithm only recurses on one side.)

Example: To selected the 5th smallest number in a list of numbers $a[] = \{4, 2, 8, 6, 3, 1, 7, 5\}$, we first pick a random pivot number (say 3), then partition the array into two sub-arrays: $(\{2, 1\}, \{4, 8, 6, 7, 5\})$. After that, we recurse on the right sub-array, i.e., we want to find the 2nd smallest number in $\{4, 8, 6, 7, 5\}$. We keep doing this until the algorithm returns, and the output of this algorithm should be 5.

Worst Case: Suppose we are extremely unlucky and always pick the smallest/largest element as the pivot, then the running time of this instance can be $\Theta(n^2)$.

5.2 Analysis of Quick Selection Algorithm

Theorem 7. The expected running time of Quick Selection Algorithm is $O(n)$.

Proof. Let X_n be the running time of Quick Sort on array of length n, note that X_n is a random variable, and we want to analyze $\mathbb{E}[X_n]$.

For simplicity, assume the elements in the array are distinct. Also, define $pivot = i$ to be the event that the first pivot number is equal to the i-th smallest number in the array. Thus, by **Law of Total Expectations**, we get

$$\mathbb{E}[X_n] = \sum_{i=1}^{n} \mathbb{E}[X_n | pivot = i] \cdot \Pr[pivot = i].$$

After we select the i-th smallest number in the array as the pivot, we will divide the other elements into two sub-arrays, whose sizes are $i-1$ and $n-i$, respectively. The division process costs $O(n)$ time, so we assume that this process can be done in An time, where A is a constant. Also, we recurse on the left sub-array if $k < i$, recurse on the right sub-array if $k > i$, and directly return the pivot if $k = i$. Therefore, if $k < i$, we have

$$\begin{aligned}\mathbb{E}[X_n | pivot = i] &= \mathbb{E}[X_{i-1} + An | pivot = i] \\ &= \mathbb{E}[X_{i-1}] + An.\end{aligned}$$

Note that in the second equation we remove the condition in the expectation because the condition $pivot = i$ is independent of the two terms.

Similarly, if $k > i$, we have

$$\mathbb{E}[X_n | pivot = i] = \mathbb{E}[X_{n-i}] + An,$$

and if $k = i$, we have

$$\mathbb{E}[X_n | pivot = i] = An.$$

Also, since the pivot is selected uniformly at random, we know that

$$\forall i \in \{1, 2, \cdots, n\}, \Pr[pivot = i] = \frac{1}{n}.$$

Thus,

$$\mathbb{E}[X_n] = \frac{1}{n}\sum_{i=1}^{k-1} \mathbb{E}[X_{n-i}] + \frac{1}{n}\sum_{i=k+1}^{n} \mathbb{E}[X_{i-1}] + An.$$

We guess that $\mathbb{E}[X_n] = O(n)$, and we will prove this by induction.

Induction Hypothesis: $\forall k \in \mathbb{N}, \mathbb{E}[X_k] \le Ck$ (C is chosen later).

Base Case: $\mathbb{E}[X_0] = 0$.

Inductive Step: For positive integer n, assume $\mathbb{E}[X_k] \leq Ck$ for all $k < n$, then for $\mathbb{E}[X_n]$:

$$\begin{aligned}
\mathbb{E}[X_n] &= \frac{1}{n}\sum_{i=1}^{k-1}\mathbb{E}[X_{n-i}] + \frac{1}{n}\sum_{i=k+1}^{n}\mathbb{E}[X_{i-1}] + An \\
&\leq \frac{1}{n}\sum_{i=1}^{k-1}C(n-i) + \frac{1}{n}\sum_{i=k+1}^{n}C(i-1) + An \\
&= \frac{C}{n}\left(\sum_{i=1}^{k-1}(n-i) + \frac{1}{n}\sum_{i=k+1}^{n}(i-1)\right) + An \\
&= \frac{C}{n}\left(\frac{(2n-k)(k-1)}{2} + \frac{(n+k-1)(n-k)}{2}\right) + An \\
&= \frac{C}{n}\left(\frac{-2k^2 + 2(n+1)k + (n^2-3n)}{2}\right) + An \\
&= \frac{C}{n}\left(\frac{-2(k-\frac{n+1}{2})^2 + (\frac{3}{2}n^2 - 2n + \frac{1}{2})}{2}\right) + An \\
&\leq \frac{C}{n}\left(\frac{3}{4}n^2 - n + \frac{1}{4}\right) + An \\
&\leq \frac{C}{n}\left(\frac{3}{4}n^2\right) + An \\
&= \frac{3}{4}Cn + An.
\end{aligned}$$

We choose $C = 4A$ and get

$$\mathbb{E}[X_n] \leq \frac{3}{4}Cn + \frac{1}{4}Cn = Cn.$$

This finishes the induction, i.e., our hypothesis is correct. Therefore, the expected running time of Quick Selection is bounded by $4An = O(n)$. $\square$