CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Why does Quick Sort behave as expected almost always ?

It is well known that the average running time of quick sort is $O(n \log n)$ time. We analysed randomized quick sort last week and showed that expected running time of quick sort is also $O(n \log n)$. However, it is not just this fact that makes randomized quick sort one of the most efficient sorting algorithm in practice. Actually the running time of randomized quick sort does not deviate from its expected value too much. More precisely, the probability that the number of comparisons which randomized quick sort performs exceeds $(1+\epsilon)$ times is less than $n^{-4\epsilon \ln \ln n}$. To get a feel of what this means, suppose we perform randomized quick sort on one million numbers (so $\ln \ln n > 2$). The probability the number of comparisons are at least 25% more than the expected is at most 10^{-12}. Note that this is indeed a very very small probability from a practitioner's perspective as well. Second point to observe is that the probability of deviation reduces as the input size increases. In other words, as the number of elements to be sorted increases, behavior of randomized quick sort becomes even more *deterministic*. In common literature, one says that the running time of quick sort is *concentrated* around $O(n \log n)$. In this chapter, we shall provide a theoretical explanation for such a behavior of randomized quick sort. However, our aim is limited to showing that the probability of deviation of the running time of quick sort from expected value decreases inverse polynomially.

The analysis uses very elementary tools which we outline in the following section. The novelty of the proof lies in the right perspective of the randomized quick sort. This right perspective along with an effective application of the elementary tools mentioned below makes the entire analysis simple, short, and very inspiring.

1 Tools Used

1.1 Union theorem

This theorem, also known as Boole's inequality, states the following: Let $\mathcal{E}_1, \cdots, \mathcal{E}_k$ be k events defined over a probability space. Then

$$\mathbf{P}[\cup \mathcal{E}_i] \leq \sum_i \mathbf{P}[\mathcal{E}_i] \tag{1}$$

This theorem can be used very effectively in the following situations. Suppose there is an event $\mathcal{E}$ whose probability appears quite difficult to estimate. If we can formulate some k events $\mathcal{E}_i, i \leq k$, such that $\mathcal{E}$ is equal to $\cup_i \mathcal{E}_i$ and calculating probability of the events $\mathcal{E}_i$ is easy, then we can get an estimate on $\mathbf{P}[\mathcal{E}]$ using Equation 1. (How would it be useful for analysis of quick sort ? Explore ...).

1.2 A simple coin tossing problem

Suppose we have a coin which gives HEADS with probability at least 1/2. It can be observed that out of m tosses, expected number of HEADS will be at least $m/2$.

Intuitively, one feels that the probability of getting significantly fewer HEADS should be *very less*. To formalize this intuition, let χ denote the event that in $8t$ tosses of a fair coin, we get fewer than t HEADS. What is $\mathbf{P}[\chi]$? It just requires elementary probability theory and Stirling's approximation for factorial to calculate it.

Lemma 1.1 *The probability of event χ is bounded by $\left(\frac{3}{4}\right)^{8t}$*

The reader is strongly encouraged to prove Lemma 1.1 himself/herself).

2 Analysis of Quick Sort

Let S be the set of n elements to be sorted by randomized quick sort. Let e_i denote the ith smallest element from S. The running time of randomized quick sort is dominated by the number of comparison performed. The latter is a random variable and let us denote it by X. We want to show that

$$\mathbf{P}[X > cn \log_b n] < \frac{1}{n^d} \tag{2}$$

for some suitable constants b, c, d. We shall find out the values of these constants later as we proceed.

Let $\mathcal{E}$ denote the event "$X > cn \log_b n$". Spending a few minutes over the above inequality will make you realize that there is no direct and easy way to prove it. We faced similar difficulty while calculating $\mathbf{E}[X]$ in some past lecture. The difficulty arises due to the fact that the distribution of X looks too complex to comprehend owing to our global/macroscopic approach of viewing randomized quick sort. So we shall follow a microscopic approach wherein we shall focus on just one element of the set S during randomized quick sort.

Note that randomized quick sort is a recursive procedure where each recursive call receives an input set $A \subseteq S$, and does the following task. First it selects a pivot element randomly uniformly from A, compares every element of the set with the pivot element, and thus splits the input set into two subsets $A_{<x}$, and $A_{>x}$. Each of these subsets is sorted recursively (unless they are empty). How does this entire algorithm look like from point of view of an element e_i.

In the first (original) call, if e_i is not selected as pivot element, then e_i is compared with the pivot element, say x, and then it *joins* the recursive call on one of the two subset $S_{<x}$ or $S_{>x}$ to which it belongs, and proceeds. In this manner, e_i continues participating in a sequence of recursive calls until it gets selected as a pivot element. Try to realize that the element e_i virtually *leaves* the algorithm at the end of the recursive call in which it gets selected as a pivot element. Let X_i denote the random variable for the number of recursive calls in which e_i participates before being selected as a pivot element. It can be observed that

$$X = \sum_i X_i$$

In order to use the above equality for proving Inequality 2, let us define event $\mathcal{E}_i$ as the event that "$X_i > c \log_b n$".

Question 2.1 *What is the relationship between event $\mathcal{E}$ and events $\mathcal{E}_i$'s ?*

It follows from simple *averaging* principle, that whenever $X > c \log_b n$, there is some j such that $X_j > c \log_b n$. In other words, whenever $\mathcal{E}$ occurs, at least one of $\mathcal{E}_j$ for some j also occurs. Hence,

$$\mathcal{E} \subseteq \cup_{j \le k} \mathcal{E}_j$$

Hence by union theorem, it follows that $\mathbf{P}[\mathcal{E}] \le \sum_{j \le k} \mathcal{E}_j$. This leads to the following crucial observation.

Observation 2.1 *In order to establish Inequality 2, it suffices if we can show that the following holds for every $i \le n$.*

$$\mathbf{P}[\mathcal{E}_i] = \mathbf{P}[X_i > c \log_b n] \le \frac{1}{n^{d+1}}$$

2.1 Bounding $\mathbf{P}[\mathcal{E}_i]$

The event $\mathcal{E}_i$ corresponds to "$X_i > c \log_b n$". To get a good bound on $\mathbf{P}[\mathcal{E}_i]$, let us take a much closer look at the randomized quick sort from perspective of e_i. See Figure 1. The first recursive call in which it participates has input size n. Let the pivot element selected in this call be e_j for some $j < i$. Then e_i is compared with e_j in this call, and then the next recursive call in which e_i participates will have input $\{e_{j+1}, \cdots, e_n\}$. Let the pivot element selected in this call be $e_k, k > i$. Then e_i will be compared with e_j, and the next recursive call in which e_i participates has input $\{e_{j+1}, ..., e_{k-1}\}$. This goes on till the recursive call in which e_i is selected as pivot element. Definitely it will happen in at most n steps. What we shall show that it will happen much sooner with high probability.

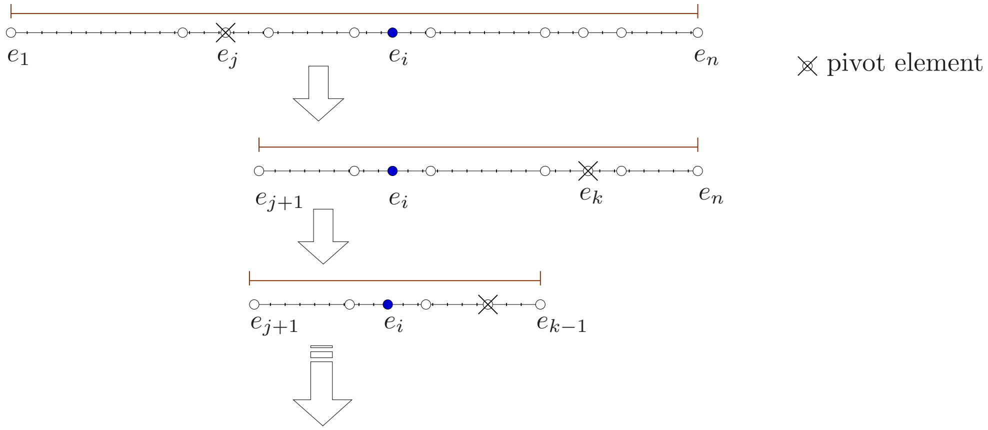

Figure 1: The recursive calls in which e_i participates.

Consider any recursive call invoked during randomized quick sort on S. Let its input be a set A of k elements. We define *central half* of A as the set of those elements from A whose rank within A is in the range $[k/4, 3k/4]$.

Question 2.2 *What is the probability that recursive call on A selects a pivot element from the central half of A ?*

It is 1/2 owing to the uniformity employed in the random selection of pivot element. Well, what will be the consequence if pivot element is selected from central half ? If this happens, the size of each of the two sets on which we invoke randomized quick sort recursively will be bounded by $3/4|A|$. Let us call a recursive call *good* if it happens, and *bad* otherwise.

Observation 2.2 *Any recursive call in which e_i participates is going to be good with probability at least 1/2.*

Question 2.3 *What can be the largest number of good recursive calls in which e_i participates ?*

Each recursive call reduces the size of the set containing e_i to at most 3/4th of the initial size. Hence it follows that there can be at most $\log_{4/3} n$ good recursive calls in which e_i can participate.

So using the above insights and observations, the following box captures the essence of randomized quick sort from perspective of e_i.

> During randomized quick sort, e_i participates in a sequence of recursive calls each of which is going to be good independently with probability at least 1/2. Furthermore, e_i leaves (algorithm) on or before participating in $\log_{4/3} n$ good recursive calls.

Choosing $c = 8$ and $b = 4/3$, the event $\mathcal{E}_i$ can be restated as: *e_i participates in more than $8 \log_{4/3} n$ recursive calls and fewer than $\log_{4/3} n$ of them are good.*

We want to find $\mathbf{P}[\mathcal{E}_i]$. Does all this remind you of some probability exercise you saw recently ? The reader is advised to take a pause here before proceeding further. Recall the simple coin tossing experiment we discussed in the beginning. Try to realize the exact correspondence between the event $\mathcal{E}_i$ defined above and the event χ defined in simple coin tossing problem. Using Lemma 1.1, we can conclude that

$$\mathbf{P}[\mathcal{E}_i] \leq \left(\frac{3}{4}\right)^{8 \log_{4/3} n} = \frac{1}{n^8}$$

Combining the above bound and Observation 2.1, we can state the following theorem.

Theorem 2.1 *Probability that randomized quick sort on an input of size n performs more than $8n \log_{4/3} n$ comparisons is less than n^{-7}.*

3 Points to Ponder

With an open mind and scientific spirit, keep pondering over the following points.

1. Try to go through the entire analysis of randomized quick sort on your own and try to view it from various angles. Try to realize the role of each tool and importance of each question on the way to prove Theorem 2.1.

2. The above analysis of quick sort provides inverse polynomial bound on the probability of deviation of quick sort from its expected behaviour. Try to enquire if this (inverse polynomial) is the best one can obtain using the analysis given above.

3. Try to make a list of useful tips which you learnt from the analysis given above. This might be useful for other problems in future.