Randomized Algorithms: Bounds, Quicksort, and Quickselect

1 Deviation Bounds

We review Markov’s inequality, Chebyshev’s inequality, and the union bound, also giving the standard and entropic Chernoff bounds.

1.1 Markov’s Inequality

Let $X$ be a nonnegative random variable, and consider $a > 0$. The probability that $X$ exceeds $a$ falls off as $1/a$, and this holds for any probability distribution:

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a},$$

where $\mathbb{E}[X]$ denotes the expectation of $X$.

How can we prove this? For any event $E$, we can define the binary-valued indicator random variable $1_E$. $1_E$ is $1$ if and only if $E$ occurs:

$$1_E = \begin{cases} 1 & \text{if } E \text{ occurs,} \\ 0 & \text{otherwise.} \end{cases}$$

We note the useful property that

$$\mathbb{E}[1_E] = \Pr[E].$$

Using this notation, $1_{X \geq a}$ will take value $1$ if $X \geq a$ and $0$ otherwise. Thus, $a1_{X \geq a}$ will take value $a$ if $X \geq a$ and $0$ otherwise. In either case,

$$a1_{X \geq a} \leq X.$$

Taking expectations of both sides and using linearity of expectation, we get

$$a \Pr[X \geq a] \leq \mathbb{E}[X].$$

Dividing both sides by $a$ gives Markov’s inequality.
1.2 Chebyshev’s Inequality

A consequence of Markov’s inequality, Chebyshev’s inequality is a more interesting bound that falls off quadratically. For a (not necessarily nonnegative) random variable $Y$ with expectation $E[Y] = \mu$ and finite non-zero variance $\sigma^2$, we have for any $k > 0$:

$$\Pr[|Y - \mu| \geq k\sigma] \leq \frac{1}{k^2}.$$  

We note that we need only prove

$$\Pr[(Y - \mu)^2 \geq k^2\sigma^2] \leq \frac{1}{k^2}.$$  

Indeed, take $X = (Y - \mu)^2$ (so $X$ is positive, and $E[X]$ is the variance of $Y$, i.e. $\sigma^2$) and $a = k^2\sigma^2$ in Markov’s inequality:

$$\Pr[(Y - \mu)^2 \geq k^2\sigma^2] \leq \frac{\sigma^2}{k^2\sigma^2} = \frac{1}{k^2},$$  

as desired.

1.3 Union Bound

Let $E_1, E_2, \ldots$ be a (not necessarily finite, but countable) collection of events, which need not be independent. Then the probability of at least one event occurring is bounded above by the sum of the individual event probabilities:

$$\Pr \left[ \bigcup_i E_i \right] \leq \sum_i \Pr[E_i].$$  

This is useful e.g. to turn a high-probability bound on one variable to a high-probability bound on $n$ variables.

1.4 Chernoff Bound

Let $X_1, \ldots, X_n$ be independently but not necessarily identically distributed random variables with values in $\{0, 1\}$. We can bound the probability that $\sum_i X_i$ exceeds its expectation by a constant multiplicative factor $\delta$, for any $\delta \in (0, 1]$:

$$\Pr \left[ \sum_i X_i > (1 + \delta)E \left[ \sum_i X_i \right] \right] \leq e^{-\delta^2 E[\sum_i X_i]/3}.$$  

We can also bound the probability that $\sum_i X_i$ falls below its expectation by a constant multiplicative factor $\delta \in (0, 1]$:

$$\Pr \left[ \sum_i X_i < (1 - \delta)E \left[ \sum_i X_i \right] \right] \leq e^{-\delta^2 E[\sum_i X_i]/2}.$$
These bounds are relatively sharp, and are useful for analyzing randomized algorithms. Defining $X = \sum_i X_i$ and $\mu = \mathbb{E}[X]$, we can write these bounds more succinctly:

$$\Pr[X > (1 + \delta)\mu] \leq e^{-\delta^2\mu/3},$$

$$\Pr[X < (1 - \delta)\mu] \leq e^{-\delta^2\mu/2}.$$

### 1.5 Entropic Chernoff Bound

Consider any $\delta > 0$, and define $H_p(x)$, the “relative entropy of $x$ with respect to $p$”:

$$H_p(x) = x \cdot \log \left( \frac{x}{p} \right) + (1 - x) \cdot \log \left( \frac{1 - x}{1 - p} \right).$$

Set $p = \mu/n$. Then we have the following entropic Chenoff bound, of which the standard Chernoff bound is a consequence:

$$\Pr[Y \geq (1 + \delta)\mu] \leq e^{-nH_p((1+\delta)\mu/n)}.$$

### 2 Quicksort

We will now analyze randomized quicksort. To sort an array $A$ of $n$ elements, we choose a pivot element $A[i]$ randomly; pivot the array around $A[i]$ in $\Theta(n)$ time so that all elements whose values are less than the pivot come before the pivot, while all elements with values greater than the pivot come after it; and recurse on the two resulting subarrays. Once the recursion reaches a subarray of size 1 (a singleton), we stop.

If we choose $A[i]$ to be the median using a selection algorithm, the two halves will be perfectly balanced, and our recurrence for the running time $T(n)$ will be $T(n) = 2T(n/2) + \Theta(n)$, with solution $T(n) = \Theta(n \log n)$. This approach gives poor constant factors.

If we are lucky and choose $A[i]$ to be close to the median in sorted order, the two subarrays will be relatively balanced. We then call $A[i]$ a “good pivot.” If we choose $A[i]$ to be close to the min or max of $A$, then we will have done $\Theta(n)$ pivoting work to split the array into two unbalanced subarrays, and so we call $A[i]$ a “bad pivot.”

To show that quicksort runs in $\Theta(n \log n)$ time with high probability (or even $\Theta(n \log n)$ expected time), we need to show that we do not choose too many bad pivots, because bad pivots lead to subarrays that are too large and do not produce singletons quickly enough. For a fixed $\theta \in (0, 1/2)$, we call a pivot that results in each subarray having at least $\theta n$ elements “good”. We call this a $\theta - (1 - \theta)$ split. The probability of achieving such a split is $1 - 2\theta$, because the number of elements in the first subarray is uniformly random between 0 and $(n - 1)$, and we need it to contain between $\theta n$ and $(1 - \theta) n$ elements to achieve a good split.

It is convenient to analyze the number of levels of recursion performed in quicksort, because the total time per level of recursion is $O(n)$. Indeed, imagine a $30\% - 70\%$ split. Pivoting the first subarray will take time $O(0.3n)$, while pivoting the second subarray will take time $O(0.7n)$, for total time $O(n)$ for the second level of recursion. If a further round of pivoting results in a
20% \(-10\% - 35\% - 35\%\) split, pivoting will take time \(O(0.2n) + O(0.1n) + O(0.35n) + O(0.35n)\), or total time \(O(n)\) for the third level of recursion, and so on.

### 2.1 Paranoid Quicksort

Consider a variant of randomized quicksort, where we repeatedly choose a pivot until we achieve a \(25\% - 75\%\) split or better. Each time we choose a pivot, we have a \(50\%\) chance of achieving such a split, so we need to choose \(1/0.5 = 2\) pivots in expectation before we are satisfied. The time to test each pivot choice is \(\Theta(n)\), so we spend \(2 \cdot \Theta(n) = \Theta(n)\) time in expectation per level of recursion.

Imagining our recursion tree, the larger subarray at each level shrinks by a factor \(0.75\) or better, so we have \(\leq \log_{1/0.75} n = \Theta(\lg n)\) levels of recursion. This means that we do \(\Theta(n \lg n)\) work in expectation to sort an array of \(n\) numbers.

### 2.2 Quicksort

We will now analyze the standard randomized quicksort algorithm. Call a \(10\% - 90\%\) or better split “good.” Such a split occurs with probability \(80\%\). For every array element \(\alpha\) and for every recursion level \(t\), let \(X_{\alpha,t}\) be the indicator random variable for the event of choosing a bad pivot in \(\alpha\)’s subarray at level \(t\):

\[
X_{\alpha,t} = \begin{cases} 
1 & \text{if, at level } t, \text{ quicksort chose a bad pivot for the subarray containing } \alpha, \\
0 & \text{otherwise.}
\end{cases}
\]

In case \(\alpha\) becomes a singleton before level \(t\) (for instance, if quicksort terminates), we define \(X_{\alpha,t}\) differently:

\[
X_{\alpha,t} = \begin{cases} 
1 & \text{with probability } 0.2, \\
0 & \text{with probability } 0.8.
\end{cases}
\]

We note that \(\mathbb{E}[X_{\alpha,t}] = 0.2\) in either case, and (holding element \(\alpha\) fixed and letting depth \(i\) vary) the \(X_{\alpha,i}\) are independent \(\{0, 1\}\)-valued random variables. We will show that after \(\Theta(\lg n)\) levels of recursion, there are not too many bad events (with high probability), and enough good events to ensure that \(\alpha\) becomes a singleton.

Indeed, fix \(\alpha\) and take \(T = c \cdot \log n\) for a constant \(c\) to be chosen. We wish to bound

\[
\Pr\left[ \sum_{i=1}^{T} X_{\alpha,i} \geq (1 + \delta) \mathbb{E} \left[ \sum_{i=1}^{T} X_{\alpha,i} \right] \right],
\]
for a constant $\delta$ to be chosen. We apply a Chernoff bound and simplify the exponential:

$$\Pr \left[ \sum_{i=1}^{T} X_{\alpha,i} \geq (1 + \delta) \mathbb{E} \left[ \sum_{i=1}^{T} X_{\alpha,i} \right] \right] \leq e^{-\delta^2 \mathbb{E}[\sum_{i=1}^{T} X_{\alpha,i}] / 3}$$

$$= e^{-\delta^2 (\sum_{i=1}^{T} \mathbb{E}[X_{\alpha,i}]) / 3}$$

$$= e^{-\delta^2 T / 15}$$

$$= e^{-\delta^2 \log n / 15} = n^{-\delta^2 / 15}.$$ 

Taking $c = 30$ and $\delta = 1$, we have

$$\Pr \left[ \sum_{i=1}^{T} X_{\alpha,i} \geq 2 \cdot \mathbb{E} \left[ \sum_{i=1}^{T} X_{\alpha,i} \right] \right] \leq 1/n^2.$$ 

Let $E_\alpha$ be the event that $\sum_{i=1}^{T} X_{\alpha,i} \geq 2 \cdot \mathbb{E}[\sum_{i=1}^{T} X_{\alpha,i}]$ – we have many bad pivots for $\alpha$. By the union bound,

$$\Pr[\text{many bad pivots for some } \alpha] = \Pr \left[ \bigcup_{\alpha} E_\alpha \right] \leq \sum_{\alpha} \Pr[E_\alpha] \leq n \cdot 1/n^2 = 1/n.$$ 

That is, with probability at least $1 - 1/n$, we have

$$\#(\text{bad events involving } \alpha) = \sum_{i=1}^{T} X_{\alpha,i} < 2 \cdot \mathbb{E} \left[ \sum_{i=1}^{T} X_{\alpha,i} \right] = (2/5)T,$$

for every element $\alpha$ in our array. Equivalently, with probability at least $1 - 1/n$, we have

$$\#(\text{good events involving } \alpha) \geq (3/5)T.$$ 

Suppose that this holds, and consider the subarray containing $\alpha$ at each level of the recursion. With every good pivot choice, its size shrinks by a factor of $0.9$ or less. Why? A good pivot choice yields a $10\%-90\%$ split or better, so the subarray is broken into two pieces, each of which have size $\leq 0.9$ times the original. After $T = 30 \log n$ levels of recursion, we see at least $(3/5)T = 18 \log n$ good pivots. The size of the subarray containing $\alpha$ shrinks from $n$ to

$$\leq 0.9^{18 \log n} \cdot n = n^{18 \log 0.9} \cdot n = n^{1 + 18 \log 0.9} < 1,$$

where we use $1 + 18 \log 0.9 \approx -0.90 < 0$. With high probability $1 - 1/n$, every array element becomes a singleton, and quicksort terminates after $30 \log n$ levels of recursion.

With $\Theta(n)$ total work per level of recursion, quicksort runs in time $\Theta(n \log n)$ with high probability. Because it runs in $O(n^2)$ time in the worst case, we can also find the expected running time $\mathbb{E}[T(n)]$:

$$\mathbb{E}[T(n)] \leq (1 - 1/n)\Theta(n \log n) + (1/n)O(n^2) = O(n \log n).$$
3 Quickselect

We use the idea of randomly pivoting an array to design a simple algorithm for selection. To select the i\text{th} element (in sorted order) of an array of size \( n \), we first pick a random pivot and pivot the array accordingly. Letting \( k \) be the length of the left partition, selecting the \( i \)\text{th} element from the full array is equivalent to either:

- Selecting the \( i \)\text{th} element from the left partition, if \( i \leq k \), or
- Selecting the \((i - k)\)\text{th} element from the right partition, if \( i > k \).

We can do this recursively.

Let \( E_k \) be the event that our pivot appears in position \( k \) in the (conceptual) sorted array. If \( E_k \) occurs, we will partition our array into subarrays of size \( k \) and \( n - k \), and we will recurse on one of these subarrays. Let \( T(n) \) be the expected running time for quickselect on an array of size \( n \). Supposing that pivoting takes time \( \leq c_1 n \), we have

\[
T(n) \leq c_1 n + \sum_{k=1}^{n} \Pr[E_k] \max\{T(k), T(n - k)\}.
\]

We note that \( \Pr[E_k] = 1/n \). Although we do not know which subarray we will recurse on, \( \max\{T(k), T(n - k)\} \) is an upper bound on the expected cost of this recursion. Without loss of generality, let \( n \) be even and exploit the symmetry between \( k \) and \( n - k \):

\[
T(n) \leq c_1 n + \frac{2}{n} \cdot \sum_{k=1}^{n/2} \max\{T(k), T(n - k)\}.
\]

We will show that \( T(n) = O(n) \) by substitution and induction on the claim that \( T(n) \leq c_1 c_2 n \). We can choose \( c_2 \) high enough so that the base case of our induction holds. Now suppose that \( T(n') \leq c_1 c_2 n' \) for every \( n' < n \). We have:

\[
T(n) \leq c_1 n + \frac{2}{n} \cdot \sum_{k=1}^{n/2} \max\{T(k), T(n - k)\} \\
\leq c_1 n + \frac{2}{n} \cdot \sum_{k=1}^{n/2} \max\{c_1 c_2 k, c_1 c_2 (n - k)\} \\
\leq c_1 \cdot \left(n + \frac{2c_2}{n} \cdot \sum_{k=1}^{n/2} \max\{n, n - k\}\right).
\]
Bounding the sum from above, we have

\[
\sum_{k=1}^{n/2} \max\{n, n - k\} = \sum_{k=1}^{n/2} (n - k)
\]
\[
= \sum_{k=n/2}^{n-1} k
\]
\[
\leq (n/2) \cdot (3/4)n = (3/8)n^2.
\]

It follows that

\[
T(n) \leq c_1 \cdot \left( n + \frac{2c_2}{n} \cdot (3/8)n^2 \right)
\]
\[
= c_1 \cdot n + (3/4)c_2n = c_1n \cdot (1 + (3/4)c_2).
\]

Choosing \( c_2 \geq 4 \) yields \( T(n) \leq c_1c_2n \) as desired, completing our induction and proving that quickselect takes \( O(n) \) time in expectation.