15-451: Algorithms

Lecture 3: Quicksort

1 Probability and Algorithms

1.1 Average Case Running Time

Given an algorithm A and some input x we want to measure the "running time".

- Time: the number of steps in the execution of the algorithm.

Written $T_A(x)$.

Usually one lumps together all inputs of the same size and sets

$$T_A(n) = \max\big(T_A(x) \bigm| x \text{ has size } n \big)$$

This is the *worst-case complexity* since we take the maximum over all inputs of length n. Potentially more interesting is the *average-case complexity*

$$T_A^{\text{avg}}(n) = \sum_{|x|=n} p_x \cdot T_A(x)$$

where p_x is the probability of input x. The average space complexity is defined similarly. Clearly

$$T_A^{\text{avg}}(n) \le T_A(n)$$

but the worst-case complexity may be strictly higher.

Note that the sample space here consists of all instances of size n, and it may not be at all clear what p_x is a realistic setting. But, we still can hope to get some useful information by assuming some distribution (such as the uniform distribution).

1.2 Probabilistic Algorithms

Another important source of probability problems in the theory of algorithms are algorithms that use randomized computations: the execution of the algorithm is not deterministic as usual, but the algorithm may rely on the flip of a coin to determine the next step.

In the simple case, the output will be always be the same, regardless of what probabilistic choices the algorithm has made during the actual computation. However, the length of the computation may depend very crucially on these nondeterministic choices. If the algorithm is properly designed, then the running time will be short for most of these choices. Quicksort is a typical example for this type of algorithm.

A more drastic use of nondeterminism is to have the output depend on the choices made during the computation. For example, one might check whether a number is prime in this fashion. The crucial problem here is to guarantee that the probability for a wrong answer is small.

2 Probability Theory Basics

2.1 Sample Spaces

We need a few basic concepts from probability theory. Think about conducting a (physical) experiment such as flipping a coin, or rolling two dice. The collection of all possible outcomes of the experiment is called a *sample space* S and its elements are the *elementary events*. Often one considers *(compound) events* which are simply collections of elementary events, i.e., subsets of the sample space. For example, we could consider all outcomes of the two-dice experiment where the sum of the two dice is 8, less than 10, even and so on. For the time being we assume that the sample space S is finite.

We want to associate a probability for the occurrence of each event. To this end we use a *probability measure* $\mathsf{Pr}[]$ that assigns a real number to each event:

$$\mathsf{Pr}[] : \mathfrak{P}(S) \to \mathbb{R}.$$

In order for such a function to reflect our intuitive understanding of probability we require the following properties:

- $0 \leq \mathsf{Pr}[A] = \leq 1$
- $\mathsf{Pr}[S] = 1$
- $A \cap B = \emptyset$ implies $\mathsf{Pr}[A \cup B] = \mathsf{Pr}[A] + \mathsf{Pr}[B]$
- If $A_i \cap A_j = \emptyset$ for all $i < j$ then $\mathsf{Pr}[\bigcup_{i \geq 0} A_i] = \sum_{i \geq 0} \mathsf{Pr}[A_i]$

Here $\emptyset$ is the impossible event, and S the certain event, so that the probabilities are 0 and 1, respectively. The two conditions about mutually exclusive events state that probabilities simply add up.

Since events are subsets of the sample space we can express logical constructs in terms of Boolean operations on sets. For example, the event $A \cap B$ corresponds to "event A and also event B". Likewise, $\overline{A} = S - A$ expresses the event "not event A". Note that

- $\mathsf{Pr}[\overline{A}] = 1 - \mathsf{Pr}[A]$.
- $\mathsf{Pr}[A \cup B] = \mathsf{Pr}[A] + \mathsf{Pr}[B] - \mathsf{Pr}[A \cap B]$.

If follows that the impossible event has probability 0: $\mathsf{Pr}[\emptyset] = 1 - \mathsf{Pr}[S] = 0$. The last equation generalizes to union of more than two terms, but is a bit clumsy to state (see the inclusion-exclusion principle in combinatorics). At any rate, we have Bode's inequality:

$$\mathsf{Pr}[A_1 \cup A_2 \cup \ldots \cup A_k] \leq \mathsf{Pr}[A_1] + \mathsf{Pr}[A_2] + \ldots + \mathsf{Pr}[A_k].$$

For conjunctions of events one can show Bonferroni's inequality:

$$\mathsf{Pr}[A_1 \cap A_2 \cap \ldots \cap A_k] \geq \mathsf{Pr}[A_1] + \mathsf{Pr}[A_2] + \ldots + \mathsf{Pr}[A_k] - (k-1).$$

For inifinite sample spaces one has to contend with some mathematical problems that we have blithely ignored. For finite and countably infinite spaces one can indeed form the necessary sums to obtain the probability of a compound event. The sum $\mathsf{Pr}[A] = \sum_{e \in A} \mathsf{Pr}[e]$ converges since $\sum_{e \in S} \mathsf{Pr}[e] = 1$ and the terms are non-negative. But for uncountable spaces one has to use integrals rather than sums. Worse, one often has to make to with measures that are not defined for all subsets $A \subseteq S$, but only for a large class of such sets. In practice these difficulties are irrelevant (see any text on measure theory).

2.2 Computing Probability

The axioms from above do not actually determine the numerical values of a probability measure. However, they reduce everything to the probability of the basic events:

$$\mathsf{Pr}[A] = \sum_{e \in A} \mathsf{Pr}[e].$$

If all the elementary events have the same probability we speak about a *uniform probability distribution.* It follows from the first property that in this case

$$\mathsf{Pr}[e] = 1/\left|S\right|$$

and therefore

$$\mathsf{Pr}[A] = \left|A\right| / \left|S\right|.$$

For example, assuming that our two dice are fair the probability of getting a sum of 8 is 1/9.

In the non-uniform case we assume that we have knowledge about the values of $\mathsf{Pr}[e]$. For example, for a coin we may assume that the probability of heads is p where $0 < p < 1$. Then the probability for tails is forced to be $1 - q$.

2.3 Random Variables

The outcome of an experiment is often associated with a numerical quantity, such as the sum of rolled dice. For a coin flip we may think of heads as producing 1 and tails as producing 0. Or we could measure the distance of a thrown dart to the center of the target.

Correspondingly one defines a *random variable* as a function

$$X : S \to \mathbb{R}.$$

Note that one can add random variables: $(X+Y)(e) = X(e)+Y(e)$. Likewise, we can multiply by are real, or perform other arithmetic operations with random variables. We won't deal with variables that can assume uncountably many values. The function $F_X(x) = \mathsf{Pr}[X = x] = \mathsf{Pr}[\{\, e \mid X(e) = x \,\}]$ is the *probability density function* of X and describes the likelihood of hitting a particular value.

The most important concept associated with random variables is the average value, or *expected value* (sometimes also called the *mean*). It is defined by

$$\mathsf{E}[X] = \sum_{e \in S} \mathsf{Pr}[e] \cdot X(e)$$

If X is clear from context one often writes μ for the expected value. One can easily show that expectation is linear in the sense that

$$\mathsf{E}[aX + bY] = a\mathsf{E}[X] + b\mathsf{E}[Y]$$

where a and b are real constants.

Another important consideration is the spread of values of the random variable around its mean. The *variance* of X is defined to be

$$\mathsf{V}[X] = \mathsf{E}[(X - \mathsf{E}[X])^2].$$

It follows from linearity that

$$\mathsf{V}[X] = \mathsf{E}[X^2] - \mathsf{E}[X]^2.$$

Since variance measures the square of the deviation from the mean one often also uses the *standard deviation* defined to be $\sigma = \sqrt{\mathsf{V}[X]}$.

Example: Rolling a Die

Here is a rather too fastidious explanation of the average value obtained by rolling one fair die. We can think of the sample as $S = \{\text{one}, \text{two}, \ldots, \text{six}\}$ (actually, an six-element set will do). Since the die is fair we have $\mathsf{Pr}[e] = 1/6$ for all $e \in S$. Now define the random variable X to be the number of spots. Then the mean is

$$\mathsf{E}[X] = \sum_{e \in S} \mathsf{Pr}[e] \cdot X(e) = \sum_{x \in [6]} 1/6x = 7/2.$$

For the variance we have

$$\mathsf{E}[X^2] = \sum_{x \in [6]} 1/6x^2 = 91/6$$

so that $\mathsf{V}[X] = 35/12$ and $\sigma \approx 1.71$.

We can describe the likelihood of a random variable assuming values far away from the mean as follows.

Lemma 2.1 *Chebyshev's Inequality*

$\mathsf{Pr}[|X - \mu| \geq c] \leq \sigma^2/c^2$.

Another useful bound (in particular for repeated trials) is the following.

Lemma 2.2 *Markov's Inequality*

Let X be a random variable assuming positive integer values. Then $\mathsf{Pr}[X \geq k\mu] \leq 1/k$.

2.4 Conditional Probability and Independence

Partial knowledge of the outcome of an experiment may affect the probability of an event. For example, if we roll a die and the result is even, the probability of getting a 4 is 1/3 rather than 1/6 in the general case. This leads to the notion of *conditional probability*:

$$\mathsf{Pr}[\,A \mid B\,] = \frac{\mathsf{Pr}[A \cap B]}{\mathsf{Pr}[B]}.$$

Two events A and B are *independent* if no additional information can be obtained from one about the other:

$$\mathsf{Pr}[A \cap B] = \mathsf{Pr}[A] \cdot \mathsf{Pr}[B].$$

Note that in this case

$$\mathsf{Pr}[\, A \mid B \,] = \mathsf{Pr}[A].$$

Lemma 2.3 *Bayes' Theorem*

$\mathsf{Pr}[A]\,\mathsf{Pr}[\, B \mid A \,] = \mathsf{Pr}[B]\,\mathsf{Pr}[\, A \mid B \,].$

Likewise, two random variables are *independent* if $\mathsf{Pr}[X = x, Y = y] = \mathsf{Pr}[X = x] \cdot \mathsf{Pr}[Y = y]$.

Lemma 2.4 *For independent random variables we have* $\mathsf{E}[XY] = \mathsf{E}[X] \cdot \mathsf{E}[Y]$.

2.5 Bernoulli Trials

One important type of experiment is a repetition of some basic experiment. The individual outcomes are assumed to be independent of each other. We can then ask how many times an event associated with the For example, suppose we flip a coin n times and observe X, the number of heads. We can define an *indicator variable* X_i that is 1 if the ith repetition produces heads, and 0 otherwise. Then $X = X_1 + X_2 + \ldots + X_n$. If the coin has bias p then

$$\mathsf{Pr}[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$$

as a simple counting argument shows. One can easily check that $\sum_{k=0}^{n} \mathsf{Pr}[X = k] = 1$.

X is said to be *binomially* distributed. We have

$$\mathsf{E}[X] = np \qquad \mathsf{V}[X] = npq.$$

If we count the number of times till heads appear we get a random variable Y such that $\mathsf{Pr}[Y = k] = p(1-p)^{k-1}$ for $k \geq 1$. In this case we speak of a *geometric* distribution and have

$$\mathsf{E}[Y] = 1/p \qquad \mathsf{V}[Y] = q/p^2.$$

3 Quicksort

Divide-and-conquer approach:

- partition (costly)
- recursively sort
- join (trivial, do nothing)

Partion is linear time, but the length of the resulting subarrays depends crucially on the choice of the pivot element.

Compare this to Mergesort:

- split (trivial)
- recursively sort
- merge (costly)

Length of the subarrays is always $n/2$, and the merge process is linear time. Hence the running time is of the form $T_{\text{MS}}(n) = 2T_{\text{MS}}(n/2) + c \cdot n$, and it is not hard to see that this recurrence has solution $T_{\text{MS}}(n) = \Theta(n \log n)$.

As we will see, Quicksort has worst case behavior $\Theta(n^2)$, but on average the running time is $\Theta(n \log n)$. In order to get a competitive algorithm we need to beat the constants in Mergesort. To this end we will insist that selection of the pivot is $O(1)$; needless to say there is nothing we can do about $\Theta(n)$ for partitioning. The reason this might well be possible is that in Mergesort, the typical comparison is
a[i] < a[j]

but in Quicksort it is
a[i] < p

so the pivot p can be kept in a fast register.

Also note that Mergesort requires more memory: $2n + \Theta(\lg n)$ in the usual recursive implementation.

3.1 Pivot Selection

Crucial problem: how to choose the pivot.

Problem: if the pivot is always the largest (or smallest) element we get quadratic running time: we only split off a single element at each step, so the recursion has depth n and the amount of work at level i is $\Theta(n - i)$.

Ideally we would like to choose the median as the pivot, but it takes way too much work to compute the median (though it can be done in linear time).

Given a subarray of the form `a[l..r]` the pivot is typically chosen as

- special position: a_l, a_r, $a_{(l+r)/2}$,
- the median of a small sample of elements, say, a_l, $a_{(l+r)/2}$ and a_r,
- at random, median of a small random sample.

Note that randomized Quicksort requires a random number generator, we will simply assume that a cheap source of randomness is available.

4 Analysis

In most probabilistic arguments it helps greatly to make several simplifying assumptions about the algorithm in question. As long as our assumptions do not decrease the running time that is perfectly acceptable.

For example, we may suppose that we only sort permutations of $[n]$: the absence of duplicates in the input array can only slow down the algorithm (we could eliminate all elements equal to the pivot, not just the pivot itself). We assume uniform distribution of the input permutations.

For definiteness, we assume the left subarray contains all elements $< p$, the right all elements $> p$. (Sometimes it may also be useful to assume that the pivot is still contained in one of the two arrays; again, this assumption could only slow down the algorithm.)

Let X_n be the random variable: size of the array in the "left" subarray. Set

$$p_i = \mathsf{Pr}[X_n = i].$$

where $i = 0, \dots, n-1$. Then

$$\mathsf{E}[X_n] = \sum_{i<n} i \cdot p_i$$

which hopefully will be close to $n/2$. Now let $t(n)$ be the average running time of quicksort on a permutation of length $[n]$. Then

$$t(n) = \begin{cases} c & \text{if } n \leq 1, \\ \sum_{i<n} p_i(t(i) + t(n-i-1)) + cn & \text{otherwise.} \end{cases}$$

4.1 Brute Force

Since the pivot is chosen at random, $p_i = 1/n$ where $i = 0, \dots, n-1$ (the left "bucket" consists of elements strictly less than the pivot).

Hence $\mathsf{E}[X_n] = \sum_{i<n} i/n = (n-1)/2$, which looks promising.

To determine the expected running time we compute

$$\begin{aligned} t(n) &= 1/n \sum_{i<n} \big(t(i) + t(n-i-1)\big) + cn \\ &= 2/n \sum_{i<n} t(i) + cn \end{aligned}$$

and thus

$$\begin{aligned} n \cdot t(n) &= 2 \sum_{i<n} t(i) + cn^2 \\ (n+1) \cdot t(n+1) &= 2 \sum_{i \leq n} t(i) + c(n+1)^2 \\ t(n+1) &= (n+2)/(n+1) \cdot t(n) + c(2n+1)/(n+1) \end{aligned}$$

which comes down to (really $\leq$)

$$t(n) = \frac{n+1}{n} \cdot t(n-1) + d.$$

The homogeneous solution for the last equation is easy: $h(n) = n + 1$. But then the solution looks like

$$\begin{aligned} h(n) \sum_{i \le n} d/h(i) &= d(n+1) \sum_{i=1}^{n} 1/i \\ &= d(n+1) H_n \end{aligned}$$

4.2 Digression: Solving the Recurrence

It is helpful to generalize a bit and try to tackle recurrences of the form

$$t(n) = \begin{cases} c & \text{if } n \le 1, \\ a(n)t(n-1) + f(n) & \text{otherwise.} \end{cases}$$

This would be easy except for the $a(n)$ term. Is there any way to eliminate this term? The trick is to consider the corresponding homogeneous equation

$$h(n) = \begin{cases} 1 & \text{if } n \le 1, \\ a(n)h(n-1) & \text{otherwise.} \end{cases}$$

Suppose we somehow have found a solution h for the homogeneous equation. Now set $t'(n) = t(n)/h(n)$ and $f'(n) = f(n)/h(n)$. Then by dividing the original equation by $h(n)$ we get

$$t'(n) = t'(n-1) + f'(n)$$

and we simply unroll this into a sum $t'(n) = \sum_{i \le n} f'(i)$. Substituting back we get

$$t(n) = h(n) \sum_{i \le n} f'(i) = \sum_{i \le n} \frac{f(i)}{h(i)}.$$

4.3 Indicator Variables and Linearity of Expectation

A (boring) Game

Pick two positive integers $a \le b$.

Generate a random integer $p \in [b]$.

- If $p = a$ you win.
- If $p < a$ you lose.
- If $p > a$ play again with $(a, p-1)$.

What is the probability $p(a, b)$ of winning?

Clearly $p(1, b) = 1$ and $p(a, a) = 1/a$.

Claim: Always $p(a, b) = 1/a$.

Proof. Arguing inductively, the probability of winning is $1/b + (b-a)/b \cdot 1/a = 1/a$. The first term comes from hitting a, the second from hitting the "play again" region. □

The key idea for the analysis is to associate the total cost of Quicksort with the number of comparisons made: the only non-recursive work part is partition, and the number of steps in partition is proportional to the number of comparisons made (no element is moved without comparison to the pivot).

Suppose we quicksort a permutation of $[n]$.

Let X_{ij} be an indicator variable

$$X_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are compared,} \\ 0 & \text{otherwise.} \end{cases}$$

Here $1 \le i < j \le n$ so we don't count double.

Then the expected number of comparisons is

$$X = \sum_{i<j} X_{ij}$$

The probability of $X_{i,j} = 1$ is $2/(j-i+1)$: this is essentially the same argument as for the game above. Hence

$$\begin{aligned} \mathsf{E}[X] &= \sum_{i<j} \mathsf{E}[X_{ij}] \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} 2/(j-i+1) \\ &= 2 \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} 1/k \\ &= 2 \sum_{i=1}^{n} (H_{n-i+1} - 1) \\ &= 2 \sum_{i=1}^{n} H_i - 2n \\ &= 2(n+1)H_n - 4n = \Theta(n \log n) \end{aligned}$$

For the last step use the identity $\sum_{i=1}^{n} H_i = (n+1)H_n - n$ (which is easy to prove by induction).