Probabilistic Analysis and Randomized Quicksort

Pedro Ribeiro

DCC/FCUP

2014/2015

On this lecture

- Introduction to **randomized (probabilistic)** algorithms
- Worst-case **expected** time bound
- Basic probabilistic **concepts**
  - events
  - random variables
  - linearity of expectation.
- Analysis of a randomized version of the **Quicksort** algorithm

Worst-Case Running Time

- We are interested on how the running time **scales** with input size
- Normally we are interested in **worst case** running time (worst case of all inputs of a given size)

Definition - Worst-Case Running Time

I - some input
$\mathbf{T(I)}$ - running time for input I
$\mathbf{T(n)}$ - worst-case running time for input size n

$$\mathbf{T(n) = max\{T(I)\}_{inputs\ I\ of\ size\ n}}$$

Average-Case Running Time

- We could be interested in **average-case**
- Measure performance in **"typical"** inputs
    - What is a typical input?
- There are algorithms with **large** gap between "average performance" and worst case.
- Can we **improve** worst-case by adding randomization?

Quicksort

Quicksort Algorithm

Given array with n elements

1. Pick an element p of the array as the **pivot** (or halt if the array has size 0 or 1).
2. **Split** the array into sub-arrays LESS, EQUAL, and GREATER by comparing each element to the pivot
   - **LESS** has all elements less than p
   - **EQUAL** has all elements equal to p
   - **GREATER** has all elements greater than p.
3. **recursively** sort LESS and GREATER.

The algorithm is not fully specified: how to pick the **pivot?**

Naive Quicksort

For a first version let's do the following:

Naive Quicksort

Run Quicksort as given before, each time choosing the **leftmost** element as the pivot

You can see an animation in the VisuAlgo website.

Naive Quicksort

Worst Case

What is the **worst case running time**?

Image the array is **already sorted**:

- The pivot will be the smallest element
- In step 2, all other elements will go to GREATER
- Since the GREATER array will be sorted, the process will continue, each time with one less element
- This will result in $\mathbf{\Omega(n^2)}$ time.
- Since step 1 is executed at most n times, and step 2 takes at most n steps, time will be $\mathbf{O(n^2)}$
- Thus, the worst-case running time is $\mathbf{\Theta(n^2)}$

Naive Quicksort

Average Case

- It turns out that the **average-case** running time is $\mathbf{O(n \log n)}$ (averaged over all different orderings of n elements)
- Small consolation if the inputs we have are the bad ones... (ex: almost sorted arrays)
- Can we get around this problem?

Randomized Quicksort

Let's now do the following:

Randomized Quicksort

Run Quicksort as given before, each time choosing **random** element as the pivot

You can see an animation in the VisuAlgo website.

Randomized Quicksort

What is now the **worst case running time**?

- We will prove that for **any** given input array I, the expected timeof this algorithm, $\mathbb{E}[\mathbf{T}(\mathbf{I})]$, is $\mathbf{O}(\mathbf{n}\log\mathbf{n})$.
- This is the **Worst-Case Expected Time** bound.
- Better than the average-case bound: we are **no longer assuming anything** from the input!
    - Ex: if the input is almost sorted, it will not affect this.
- Peculiar: making the algorithm **probabilistic** gives us **more control** over the running time.

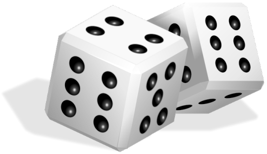
To prove these bounds, we will now focus into the basic concepts of probabilistic analysis.

Basics of Probabilistic Analysis

- Consider rolling **two dice** and observing the results.
- We call this an **experiment**.
- It has **36 possible outcomes**:

  1-1, 1-2, 1-3, 1-4, 1-5, 1-6, 2-1, 2-2, 2-3, ..., 6-4, 6-5, 6-6
- Each of these outcomes has probability $\mathbf{1/36}$ (assuming fair dice)
- What is the probability of the sum of dice being 7?

  Add the probabilities of all the outcomes satisfying this condition:
  1-6, 2-5, 3-4, 4-3, 5-2, 1-6 (probability is $\mathbf{1/6}$)



Basics of Probabilistic Analysis

In the language of probability theory, this setting is characterized by a **sample space** S and a **probability measure** p.

- **Sample Space** is constituted by all possible outcomes, which are called **elementary events**
- In a **discrete probability distribution** (d.p.d.), the probability measure is a function $p(e)$ (or $Pr(e)$) over elementary events e such that:
  - $p(e) \geq 0$ for all $e \in S$
  - $\sum_{e \in S} p(e) = 1$
- An **event** is a subset of the sample space.
- For a d.p.d. the probability of an event is just the **sum** of the probabilities of its elementary events.

Basics of Probabilistic Analysis

- A **random variable** is a function from elementary events to integers or reals:

  Ex: let X_1 be a random variable representing result of first die and X_2 representing the second die.
  $X = X_1 + X_2$ would represent the sum of the two
  We could now ask: what is the probability that $X = 7$?

- One property of a random variable we care is **expectation**:

Equation 3.1 - Expectation

For a discrete random variable X over sample space S, the expected value of X is:

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

Basics of Probabilistic Analysis

- In **words**: expectation of a random variable X is just its average value over S, where each elementary event e is weighted according to its probability.

  Ex: If we roll a single die, the expected value is 3.5
  (all six elementary events have equal probability).

- One possible rewrite of the previous equation, grouping elementary events:

Equation 3.2 - Expectation

$$\mathbb{E}[X] = \sum_a Pr(X = a)a$$

Basics of Probabilistic Analysis

- More generally:

Equation 3.3 - Expectation

For any partition of the sample space into disjoint events $A_1, A_2, \ldots$:

$$\mathbb{E}[X] = \sum_i \sum_{e \in A_i} Pr(e)X(e) = \sum_i Pr(A_i)\, \mathbb{E}[X|A_i]$$

- $\mathbb{E}[X|A_i]$ is the expected value of X given A_i, defined to be: $\frac{1}{Pr(A_i)} \sum_{e \in A_i} Pr(e)X(e)$.

Basics of Probabilistic Analysis

An important fact about expected values is **Linearity of Expectation**:

Theorem 3.1 - linearity of expectation

For any two random variables X and Y: $\mathbb{E}[\mathbf{X} + \mathbf{Y}] = \mathbb{E}[\mathbf{X}] + \mathbb{E}[\mathbf{Y}]$

Proof for discrete random variables, from equation 3.1:

$$\mathbb{E}[X + Y] = \sum_{e \in S} Pr(e)(X(e) + Y(e)) =$$

$$= \sum_{e \in S} Pr(e)X(e) + \sum_{e \in S} Pr(e)Y(e) = \mathbb{E}[X] + \mathbb{E}[Y]$$

- This is very important for the analysis of algorithms: complicated variables become sum of simple variables which we can analyse separately.

A first example

Suppose we unwrap a fresh deck of cards and **shuffle** it until the cards are completely random.

How many cards do we expect to be in the same position as they were at the start?

- X: number of cards that end in the same position as they started
- We are looking for $\mathbb{E}[X]$!
- By linearity of expectation we can write this as a sum of X_i, where $X_i = 1$ if the i-th card ends up in position i, and $X_i = 0$ otherwise.
- $Pr(X_i = 1) = 1/n$ where n is the number of cards!
- $Pr(X_i = 1)$ is also $\mathbb{E}[X_i]$
- $\mathbb{E}[X] = \mathbb{E}[X_1 + \ldots + X_n] = \mathbb{E}[X_1] + \ldots + \mathbb{E}[X_n] = 1$

Analysing Randomized Quicksort

For simplicity, let's assume no two elements are equal (in the end, it will be easy to see that equal keys could only improve performance).

We will use lg to represent $\log_2$

Theorem 3.2 - comparisons in randomized quicksort

The expected number of comparisons made by randomized quicksort on an array of size n is at most $2n \lg n$.

Let us prove this theorem in two ways.

Analysing Randomized Quicksort
Method 1

- When we pick the pivot we perform $n-1$ comparisons to split it (compare all elements to it)
- Depending on the pivot:
  - LESS of size 0 and GREATER of size $n-1$, or
  - LESS of size 1 and GREATER of size $n-2$, or
  - ...
  - LESS of size $n-1$ and GREATER of size 0, or
- All of these are equally likely with probability $1/n$

Analysing Randomized Quicksort

Method 1 (continuation)

- The expected number of comparisons can be written as $T(n)$:

Equation 3.4 - expected number of comparisons

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}(T(i) + T(n-i-1))$$

- By regrouping and getting rid of $T(0)$:

Equation 3.5 - expected number of comparisons

$$T(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} T(i)$$

- Now we can solve this by... **"guess and prove"**! :)

Analysing Randomized Quicksort
Method 1 (continuation)

We need a **good guess**:

- Most pivots should "roughly" divide in the middle
- This suggests a guess of the form $cn \lg n$ for some constant c

Analysing Randomized Quicksort

Method 1 (continuation)

Once we have a guess, we will need to evaluate a summation.

- One way is to upper bound the summation using an integral. In particular:

Equation 3.6 - bounding summations with integrals

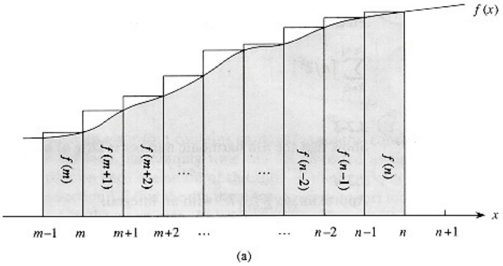
if $f(x)$ is an increasing function, then:

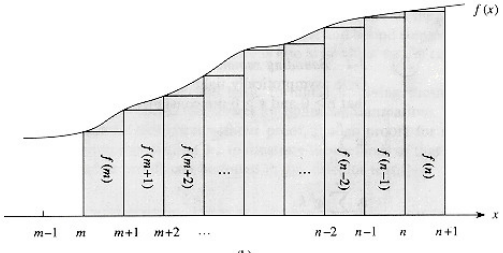
$$\sum_{i=1}^{n-1} \leq \int_1^n f(x)dx$$

Recall that integral is the "area under the curve" of $f(x)$

Analysing Randomized Quicksort

Method 1 (continuation)





Analysing Randomized Quicksort

Method 1 (continuation)

- $\int (cx \lg x) = (c/2)x^2 \lg x - cx^2/4$

Ready for the analysis! **(guess and prove by induction)**

For the base case we have that $T(0) = 0$ (no comparisons needed).
Arguing by induction that $T(i) \leq ci \lg i$ for $i < n$:

$$\begin{aligned}
T(n) &\leq (n-1) + \tfrac{2}{n}\sum_{i=1}^{n-1}(ci \lg i) && \text{(eq 3.5)}\\
&\leq (n-1) + \tfrac{2}{n}\int_1^n (cx \lg x)dx && \text{(bounding by integral)}\\
&\leq (n-1) + \tfrac{2}{n}((c/2)n^2 \lg n - cn^2/4 + c/4)\\
&\leq cn \lg n, \text{ for } c = 2 && \text{(We have proved what we wanted!)}
\end{aligned}$$

Analysing Randomized Quicksort

In terms of the number of comparisons it makes, **Randomized Quicksort is equivalent to randomly shuffling the input and then handing it off to Naive Quicksort**.

So, we have also proven that Naive Quicksort has $O(n \lg n)$ average-case running time.

Analysing Randomized Quicksort

Method 2

Let's try to do something more similar to the card shuffle example.

Let X_{ij} be a random variable with value:

- 1 if the algorithm does compare the i-th smallest and j-th smallest elements in the course of sorting
- 0 if it does not

Let X denote the total number of comparisons made by the algorithm. Since we never compare the same pair of elements twice, we have:

$$X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}$$

And therefore

$$\mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$$

Analysing Randomized Quicksort

Method 2 (continuation)

Consider one X_{ij}, for $i < j$.

Consider e_k to be the element on the k-th position, and imagine the elements in sorted order.

- If the pivot is between e_i and e_j, then they will go to separate buckets and we never compare them
- If the pivot is e_i or e_j, then we do compare them
- If the pivot is smaller than e_i or greater than e_j, then they will go to the same bucket and we need to choose another pivot

Analysing Randomized Quicksort

Method 2 (continuation)

At each step, the probability that $X_{ij} = 1$, conditioned on the event that the games ends in that step, is exactly $2/(j - i + 1)$ (choosing e_i or e_j in the interval e_i to e_j).

Therefore, overall, the probability that $X_{ij} = 1$ is $2/(j - i + 1)$.

In other words, e_i is compared to e_{i+1} with probability 1, e_i is compared to e_{i+2} with probability $2/3$, e_i is compared to e_{i+2} with probability $2/4$, and so on. So:

$$E[x] = \sum_{i=1}^{n} 2\left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n-i+1}\right)$$

Analysing Randomized Quicksort

Method 2 (continuation)

$$E[x] = \sum_{i=1}^{n} 2(\tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \ldots + \tfrac{1}{n-i+1})$$

The quantity $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$ is denoted as H_n, and is called the n-th **Harmonic Number**.

This is known to be in the range $[\lg n, 1 + \lg n]$
(consider the integral of $1/n$)

Therefore:

$E[x] < 2n(H_n - 1) \leq 2n \lg n$ (We have proved what we wanted!)