SFU CMPT-307 2008-2 Lecture: Week 4

Ján Maňuch

E-mail: jmanuch@sfu.ca

Lecture on May 27, 2008, 5.30pm-8.20pm

Performance of Quicksort

the running time depends on how balanced or unbalanced the partitions are;

and this depends on the choice of pivots

intuitively, if partitions are **balanced**,
then as in case of Mergesort,
the running time is $\mathcal{O}(n \log n)$;

if they are very **unbalanced**,
it can run as slow as Selection-Sort
and the running time is $\Omega(n^2)$

The worst case partitioning

occurs when one partition has $n-1$ elements and the other is empty (0 elements)
assume that such a partitioning happens in each recursion call

recursive call to empty array just returns the control back, which takes $T(0) = \mathcal{O}(1)$ time, so we get the recurrence for the running time:

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) \end{aligned}$$

which is the sum of arithmetic series $\Longrightarrow$
$T(n) = \Theta(n^2)$

Note: this bad partitioning can really happen (see assignment), and so we get the lower bound for the worst running time $\Omega(n^2)$

on the other hand we will prove the same upper bound

Assignment Problem 4.1. (deadline: June 3, 5:30pm)

Show that the running time of the **Quicksort** presented at the lecture is $\Theta(n^2)$ when the elements of the array A are distinct and sorted

(a) in increasing order;

(b) in decreasing order.

Worst-case performance

Let $T(n)$ be the worst-case running time on input of size n. Then

$$T(n) \leq \max_{0 \leq q \leq n-1} (T(q) + T(n-q-1)) + dn$$

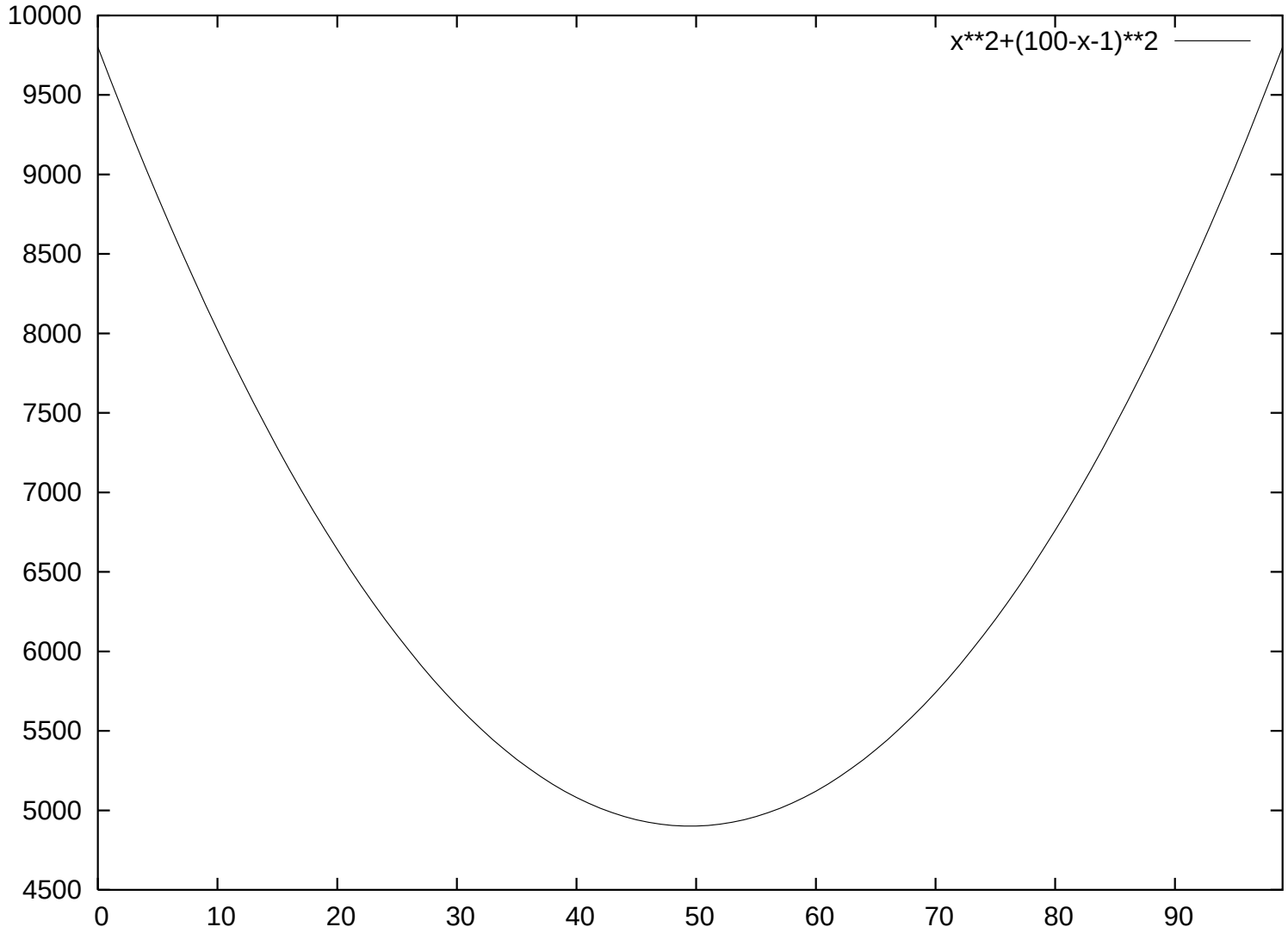
- $dn \in \Theta(n)$ is some upper bound for the time needed for partitioning
- we take the worst of all possible partitioning in each step

We guess $T(n) \leq cn^2$ for some constant c.

Inductive step:

$$\begin{aligned} T(n) &\leq \max_{0 \leq q \leq n-1} (T(q) + T(n-q-1)) + dn \\ &\leq \max_{0 \leq q \leq n-1} \left(cq^2 + c(n-q-1)^2\right) + dn \\ &= c \cdot \max_{0 \leq q \leq n-1} \left(q^2 + (n-q-1)^2\right) + dn \end{aligned}$$

Consider the function $F(q) = q^2 + (n - q - 1)^2$ in the range $0 \leq q \leq n - 1$:



This is a quadratic function with the minimum at $(n-1)/2$, and so the expression is maximized when $q = 0$ or $q = n-1$.

This implies

$$\begin{aligned} q^2 + (n-q-1)^2 &\leq (n-1)^2 + [n-(n-1)-1]^2 \\ &= (n-1)^2 \\ &= n^2 - 2n + 1 \end{aligned}$$

and therefore

$$\begin{aligned} T(n) &\leq c \cdot \max_{0 \leq q \leq n-1} \left(q^2 + (n-q-1)^2\right) + dn \\ &\leq c \cdot (n^2 - 2n + 1) + dn \\ &= cn^2 - (2c-d)n + c \\ &\leq cn^2 \end{aligned}$$

for $c \geq d$

Hence, the worst-case running time of **Quicksort** is $\Theta(n^2)$.

The best case partitioning

occurs when partitioning is even:
one partition has size $\lfloor n/2 \rfloor$ and the other $\lceil n/2 \rceil - 1$

(that is for odd n, both have the same size, and for even n, they differ by 1)

we get the recurrence:

$$T(n) = 2T(n/2) + \Theta(n)$$

which has the solution $T(n) = \Theta(n \log n)$

Assignment Problem 4.2. (deadline: June 3, 5:30pm)
Show that the best-case running time of **Quicksort** is $\Omega(n \log n)$, i.e., show that the recurrence

$$T(n) \geq \min_{0 \leq q \leq n-1} (T(q) + T(n-q-1)) + dn$$

is in $\Omega(n \log n)$.

Hint: You can use the fact that the function $f(x) = x \log x + (n-1-x) \log(n-1-x)$ achieves its global minimum at point $x = (n-1)/2$.

The average case partitioning (intuition)

assume that in every recursive step one partition has size $n/10$ and the other $9n/10$ (which seems as a quite unbalanced partitioning)
then we get the recurrence:

$$T(n) = T(9n/10) + T(n/10) + \Theta(n)$$

building the recursion tree shows that we are still in $\mathcal{O}(n \log n)$

In fact the same is true for any constant factor partition!

Efficient implementation

1. optimal procedures for sorting at the bottom of recursion
2. pick the pivots so that we get **even** partitions of subarrays $A[p \ldots r]$ into two smaller subarrays

how to modify **Partition***?*

if we pick $A[q]$ as a pivot instead of $A[r]$, we just swap $A[q]$ and $A[r]$ before step 1 of procedure **Partition**(A, p, r)

how to pick a better pivot?

There are several not-too-bad ways:

- look at, say, 5 **fixed** array elements and pick the median
- pick a **randomly chosen** element
- look at, say 5 **randomly chosen** elements and pick the median
- many more

in practice, the randomized variant usually works the best

Why randomized strategy?

some of inputs (for example, if the input is sorted or almost sorted) have a bad performance

in practical applications it often happens that the inputs are not completely random, so even if our algorithm performs good in average, some application might prefer the inputs with the worst-case performance

Example.

- transactions on an account are usually kept in order of their times
- people usually write checks in order by check number, but they are cashed with some delays
- many people want the checks listed in order by check number, hence we have to convert time-of-transaction ordering to check-number ordering
- this is sorting of nearly sorted input, which performs very badly in Quicksort we have considered

Solution.
To sort nearly ordered inputs in $\Theta(n \log n)$ time using **Quicksort** it's enough to choose a pivot $x = A[(p + r)/2]$ from the middle of subarray $A[p \ldots r]$.

However, one could design inputs on which this modification of **Quicksort** would perform badly.

what should we do to avoid the worst-case inputs?

use **randomization**!

Basically, two ways:

1. permute the input randomly before running standard version of **Quicksort**
2. leave the input as it is, but use some random pivot-selection strategy

Properties of randomized algorithms

- each time we run the algorithm on the same input, the execution depends on the random choices and is likely different from the previous execution
- *no particular* input shows the worst-case performance
- it can still happen that during a single run of the randomized algorithm it performs badly
  if the random generator produces "unlucky" numbers:
  - in the 1st approach it would permute input so that it's (almost) sorted in increasing order ("bad input")
  - in the 2nd approach it would choose pivots so that it's always the minimal or the maximal element of the subarray ("worst-case partitioning")

  but it can happen only with a very small probability

Randomized Quicksort

we will consider the 2nd approach above, called *random sampling*

In the original procedure, we have partition the subarray $A[p \ldots r]$ using the right-most element $A[r]$ as a pivot.

Now, we will use a **randomly chosen** element from $A[p \ldots r]$:

1. pick an index $z \in \{p, \ldots, r\}$ independently on other choices and uniformly at random
2. exchange $A[z] \leftrightarrow A[r]$
3. run the original procedure **Partition**

Randomized-Partition(A, p, r)

1: $z \leftarrow$ Random(p, r)
2: exchange $A[z] \leftrightarrow A[r]$
3: return Partition(A, p, r)

Note: procedure **Random**(a, b) returns an integer from the set $\{a, a+1, \ldots, b-1, b\}$ each with the same probability $1/(b-a+1)$ ("uniform distribution")

Randomized-Quicksort(A, p, r)

1: **if** $p < r$ **then**
2: $q \leftarrow$ Randomized-Partition(A, p, r)
3: Randomized-Quicksort$(A, p, q-1)$
4: Randomized-Quicksort$(A, q+1, r)$
5: **end if**

Before we can start analyzing performance of **Randomized-Quicksort** we need to recall basics of *probability theory*

Probability

Defined in terms of a **probability space** or **sample space** S (or Ω), a **set** whose elements $s \in S$ (or $\omega \in \Omega$) are called **elementary events**.

you can view elementary events as possible outcomes of an experiment.

Examples:

- flip a coin: $S = \{\text{head}, \text{tail}\}$
- roll a die: $S = \{1, 2, 3, 4, 5, 6\}$
- pick a random pivot in $A[p \ldots, r]$:
  $S = \{p, p+1, \ldots, r\}$ – indexes of the pivot

Here, we are talking only about **finite discrete** probability spaces.

An **event** is a subset of the probability space

Examples:

- roll a die; $A = \{2, 4, 6\} \subset \{1, 2, 3, 4, 5, 6\}$ is the event of having an even outcome
- flip two distinguishable coins:
  $S = \{HH, HT, TH, TT\}$, and $A = \{TT, HH\} \subset S$ is the event of having the same outcome with both coins

We say S (the entire sample space) is a **certain event**, and $\emptyset$ is the **empty** or **null event**

We say events A and B are **mutually exclusive** if $A \cap B = \emptyset$

Axioms

A **probability distribution** $P()$ on S is mapping from events of S to real numbers in interval $[0, 1]$ such that

1. $P(A) \geq 0$ for all $A \subseteq S$
2. $P(S) = 1$ (normalization)
3. $P(A) + P(B) = P(A \cup B)$ for any two **mutually exclusive** events A and B, i.e., $A \cap B = \emptyset$.

   Generalization: for any finite sequence of pairwise mutually exclusive events $A_1, A_2, \ldots$

$$P\left(\bigcup_i A_i\right) = \sum_i P(A_i)$$

$P(A)$ is called **probability** of event A

Properties of probability that that follows from axioms:

1. $P(\emptyset) = 0$
2. If $A \subseteq B$ then $P(A) \leq P(B)$
3. With $\bar{A} = S - A$, we have $P(\bar{A}) = P(S) - P(A) = 1 - P(A)$
4. For any A and B (**not** necessarily mutually exclusive),

$$\begin{aligned} P(A \cup B) &= P(A) + P(B) - P(A \cap B) \\ &\leq P(A) + P(B) \end{aligned}$$

Considering discrete sample spaces, we have for any event A

$$P(A) = \sum_{s \in A} P(s)$$

If S is finite, and $P(s \in S) = 1/|S|$, then we have **uniform probability distribution** on S (that's what's usually referred to as "picking an element of S at random")

Conditional probabilities

when you already have partial knowledge

Example: a friend rolls two fair dice (prob. space is $\{(x, y) : x, y \in \{1, \ldots, 6\}\}$) and tells you that one of them shows a 6. What's the probability for a $(6, 6)$ outcome?

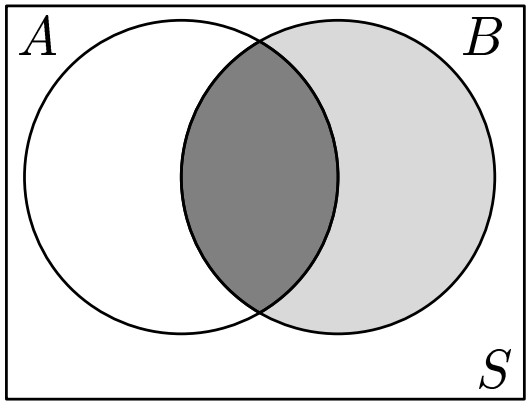
The information eliminates outcomes without any 6, i.e., all combinations of 1 through 5. There are $5^2 = 25$ of them. The original prob. space has size $6^2 = 36$, thus we are left with $36 - 25 = 11$ outcomes where at least one 6 is involved.

These are equally likely, thus the sought probability must be $1/11$.

The **conditional probability** of an event A given that another event B occurs is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

given $P(B) \neq 0$



In the example:

$$\begin{aligned} A &= \{(6,6)\} \\ B &= \{(6,x) : x \in \{1,\ldots,6\}\} \cup \\ &\quad \{(x,6) : x \in \{1,\ldots,6\}\} \end{aligned}$$

with $|B| = 11$ (the $(6,6)$ is in both parts) and thus $P(A \cap B) = P(\{(6,6)\}) = 1/36$ and

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/36}{11/36} = \frac{1}{11}$$

Independence

We say two events are **independent** if

$$P(A \cap B) = P(A) \cdot P(B)$$

which is equivalent to (if $P(B) \neq 0$) to

$$P(A|B) \stackrel{def}{=} \frac{P(A \cap B)}{P(B)} = \frac{P(A) \cdot P(B)}{P(B)} = P(A)$$

Events $A_1, A_2, \ldots, A_n$ are **pairwise independent** if

$$P(A_i \cap A_j) = P(A_i) \cdot P(A_j)$$

for all $1 \leq i < j \leq n$.

They are **(mutually) independent** if every k-subset $A_{i_1}, \ldots, A_{i_k}$, $2 \leq k \leq n$ and $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ satisfies

$$P(A_{i_1} \cap \cdots \cap A_{i_k}) = P(A_{i_1}) \cdots P(A_{i_k})$$

Example: Throw two fair dice, one green and one red.

Consider 2 events:

- A: the event that their sum is 7; $P(A) = |A|/36 = 6/36 = 1/6$
- B: the event that the red die shows an even number; $P(B) = 1/2$

Are they independent?

$$P(A \cap B) = P(\{(1,6),(3,4),(5,2)\}) = 3/36 = 1/12$$

Test for independence

$$\boxed{P(A \cap B) = P(A)P(B)\ ?}$$

$$P(A \cap B) = \frac{1}{12} = \frac{1}{6} \cdot \frac{1}{2} = P(A) \cdot P(B)$$

Therefore, the events are independent.

Assignment Problem 4.3. (deadline: June 3, 5:30pm)
Consider a probability space $S = \{1, 2, \ldots, 8\}$ (outcome of a throw of 8-sided die). Find and example of three events A, B, C of S such that A, B, C are pairwise independent, but events A and $B \cap C$ are not (i.e. $P(A) \cdot P(B \cap C) \neq P(A \cap B \cap C)$).

Random variables

A **random variable** X is a function from a probability space S to the set of real numbers, i.e., it assigns some value to elementary events

Event "$X = x$" is defined to be $\{s \in S : \ X(s) = x\}$

Example: roll three dice

- $S = \{s = (s_1, s_2, s_3) \mid s_1, s_2, s_3 \in \{1, 2, \ldots, 6\}\}$
  $|S| = 6^3 = 216$ possible outcomes
- Uniform distribution: each element has probability $1/|S| = 1/216$
- Let random variable X be the sum of dice, i.e.,
  $X(s) = X(s_1, s_2, s_3) = s_1 + s_2 + s_3$

$P(X = 7) = 15/216$ because

115	214	313	412	511
124	223	322	421	
133	232	331		
142	241			
151				

Important: With a random variable X, writing $P(X)$ does **not** make any sense;
$P(X = \text{something})$ **does**, (because it's an **event**)

Clearly, $P(X = x) \geq 0$ and $\sum_x P(X = x) = 1$ (from probability axioms)

If X and Y are random variables then $P(X = x \text{ and } Y = y)$ is called **joint probability distribution** of X and Y.

$$P(Y = y) = \sum_x P(X = x \text{ and } Y = y)$$

$$P(X = x) = \sum_y P(X = x \text{ and } Y = y)$$

Independence of random variables

Random variables X, Y are **independent** if for all x, y, events "$X = x$" and "$Y = y$" are independent

Recall: events A and B are independent iff $P(A \cap B) = P(A) \cdot P(B)$.

Now: X, Y are independent iff for all x, y,

$$P(X = x \text{ and } Y = y) = P(X = x) \cdot P(Y = y)$$

Intuition:

$$\begin{aligned} A &:= [X = x] = [X = x \text{ and } Y = ?] \\ B &:= [Y = y] = [X = ? \text{ and } Y = y] \end{aligned}$$

$$A \cap B := [X = x \text{ and } Y = y]$$

Expected values of random variables

Also called **expectations** or **means**

Given a random variable X, its expected value is

$$E[X] = \sum_x x \cdot P(X = x)$$

Well-defined if sum is finite or converges absolutely

Sometimes written μ_X (or μ if context is clear)

Example: roll a fair six-sided die, let X denote expected outcome

$$\begin{aligned} E[X] &= 1 \cdot 1/6 + 2 \cdot 1/6 + 3 \cdot 1/6 + \\ &\quad 4 \cdot 1/6 + 5 \cdot 1/6 + 6 \cdot 1/6 \\ &= 1/6 \cdot (1 + 2 + 3 + 4 + 5 + 6) \\ &= 1/6 \cdot 21 \\ &= 3.5 \end{aligned}$$

Another example: flip three fair coins

For each head you win \$4, for each tail you lose \$3

Let a random variable X denote your profit. Then the probability space is

$$\{\text{HHH,HHT,HTH,THH,HTT,THT,TTH,TTT}\}$$

and

$$\begin{aligned} E[X] &= 12 \cdot P(\text{3H}) + 5 \cdot P(\text{2H}) - \\ &\quad -2 \cdot P(\text{1H}) - 9 \cdot P(\text{0H}) \\ &= 12 \cdot 1/8 + 5 \cdot 3/8 - 2 \cdot 3/8 - 9 \cdot 1/8 \\ &= \frac{12 + 15 - 6 - 9}{8} = \frac{12}{8} = 1.5 \end{aligned}$$

which is intuitively clear: each single coin contributes an expected win of 0.5

Linearity of expectations

Important:

$$E[X + Y] = E[X] + E[Y]$$

whenever $E[X]$ and $E[Y]$ are defined

True even if X and Y are **not independent**

Exercise 4.1. Roll three 6-sided dice. Consider the following two random variables:

X = the sum of dice

Y = the difference between the die with the maximal outcome and the die with the minimal outcome

(a) Find out whether X and Y are independent.

(b) Find expected values of X and Y.

Some more properties

Given random variables X and Y with expectations, a constant a

- $E[aX] = aE[X]$

  (note: aX is a random variable)
- for constants a, b,
  $E[aX + bY] = E[aX] + E[bY] = a \cdot E[X] + b \cdot E[Y]$

- if X, Y **independent**, then

$$\begin{aligned}
E[XY] &= \sum_z zP(XY = z) \\
&= \sum_z \sum_{xy=z} zP(X = x \text{ and } Y = y) \\
&= \sum_x \sum_y xyP(X = x \text{ and } Y = y) \\
&= \sum_x \sum_y xyP(X = x)P(Y = y) \\
&= \left(\sum_x xP(X = x)\right)\left(\sum_y yP(Y = y)\right) \\
&= E[X]E[Y]
\end{aligned}$$