SFU CMPT-307 2008-2 Lecture: Week 5

Ján Maňuch

E-mail: jmanuch@sfu.ca

Lecture on June 3, 2008, 5.30pm-8.20pm

Analysis of Randomized-Quicksort

We want to analyse **expected running time**

Already have some intuition:
if splits are (more or less) balanced, then good performance

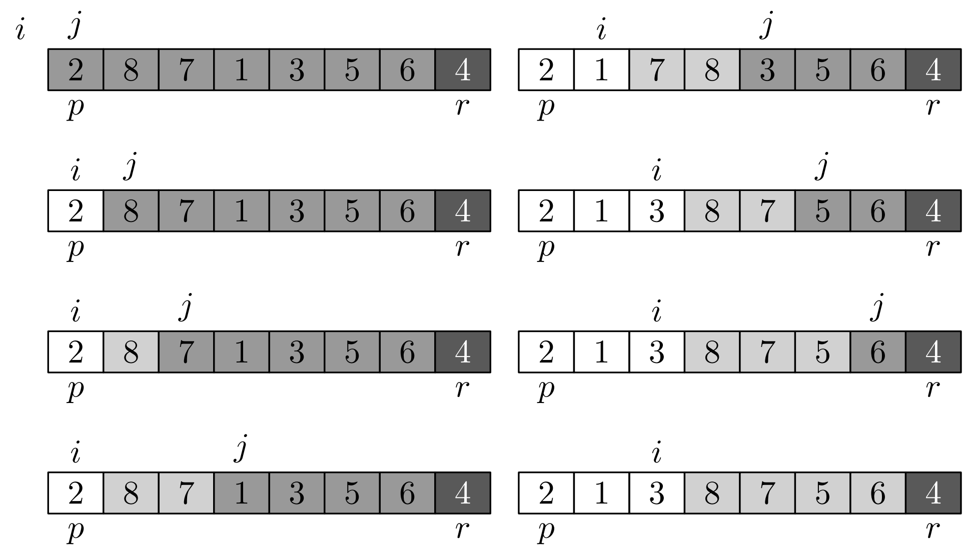
Some observations:

- running time is dominated by time spent in Partition()
- each time Partition() is called, a pivot is selected
- this pivot is **never again** included in further recursive calls
- thus at most n calls to Partition() over **entire** execution

Recall the Partition algorithm

Partition(A, p, r)

$x \leftarrow A[r]$ */* choose a pivot x */*
$i \leftarrow p - 1$
for $j \leftarrow p$ **to** $r - 1$ **do**
  if $A[j] \leq x$ **then**
    $i \leftarrow i + 1$
    exchange $A[i] \leftrightarrow A[j]$
  end if
end for
exchange $A[i+1] \leftrightarrow A[r]$
return $i + 1$



- one call to Partition() takes $O(1)$ plus amount proportional to # of iterations of the loop
- each iteration compares pivot to some other element
- thus bounding **total** # of comparisons yields bound on **total** time spent in loop (which dominates overall running time)

Lemma. Let X be # of comparisons over entire execution on n-element array. Then running time is $O(n + X)$.

Proof. At most n calls to partition, each of which

- does constant amount of work, and then
- executes the loop some # of times

Each iteration of loop performs one comparison

Seems we need to bound X, total # of comparisons

Not going to analyze # of comparison in **each** call to Partition(), but rather total #

Convenience: rename elements of A as $z_1, z_2, \ldots, z_n$ with z_i being i-th smallest element.

Let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

Question: does algorithm compare z_i and z_j and how often?

Observation: each pair of elements is compared **at most once** (comparisons only to pivot, and that one never again)

Define random variables

$$X_{ij} = \begin{cases} 1 & z_i \text{ is compared to } z_j \text{ at some time} \\ 0 & \text{otherwise} \end{cases}$$

Each pair compared at most once, thus

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

is total # of comparisons during entire run

Interested in expectations:

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(z_i \text{ is compared to } z_j) \end{aligned}$$

2nd equation is because of linearity of expectation,
3rd because X_{ij} is so-called **Bernoulli** (or $0-1$) random variable: by definition, $E[X_{ij}] = \sum_x x \cdot P(X_{ij} = x)$, and with Bernoulli random variable we have

$$E[X_{ij}] = 0 \cdot P(X_{ij} = 0) + 1 \cdot P(X_{ij} = 1) = P(X_{ij} = 1)$$

So, now we only need to bound the probability $P(z_i$ is compared to $z_j)$

Let's do it the other way around: when are they **not** compared?

- once a pivot x with $z_i < x < z_j$ is chosen, z_i and z_j **cannot** be compared at any subsequent time (they are in different branches of the recursion tree)

Note: elements of Z_{ij} are (initially) not necessarily in adjacent positions in (subarray of) A. Could look like

$$[\cdots z_j \cdots z_i \cdots x \cdots]$$

However, after partitioning (given $z_i < x < z_j$)

$$[\cdots z_i \cdots]\, x\, [\cdots z_j \cdots]$$

- **prior** to the point where some element from Z_{ij} is chosen, the whole set Z_{ij} is together in one partition.
- if z_i is chosen as a pivot **before any other item** in Z_{ij}, then then z_i will be compared to each item in Z_{ij}, except itself
- similar for z_j

Thus, z_i and z_j are compared **if and only if** the first element to be chosen as a pivot from Z_{ij} is either z_i or z_j (again, at this time Z_{ij} can be mixed with other elements)

Example: consider an input $[3, 5, 1, 2, 10, 9, 7, 8, 6, 4]$
Assume that the first pivot is 7. After the first call to Partition()

$$[3, 5, 1, 2, 4, 6]\ 7\ [8, 9, 10]$$

7 is compared to **every other** number, but, say, 2 will **never** be compared to, say, 9

Since elements in Z_{ij} are in the same partition before any of them is chosen as a pivot, each one has the same probability of being the first one chosen (among all from Z_{ij}).

$|Z_{ij}| = j - i + 1$, thus probability that any given element is the first one chosen as a pivot is $1/(j - i + 1)$

Note: This is **not** the probability that

- a given element is chosen as a pivot during the execution of the algorithm;
- neither that a given element is chosen as a pivot during a (any) partitioning step;
- but **it is** the probability that a given element is chosen as a pivot during partitioning in which one of the elements of Z_{ij} is chosen as a pivot.

$$\begin{aligned}
& P(z_i \text{ is compared to } z_j) \\
&\quad = \quad P(z_i \text{ or } z_j \text{ is first pivot chosen from } Z_{ij}) \\
&\quad \stackrel{(*)}{=} \quad P(z_i \text{ is first pivot chosen from } Z_{ij}) + \\
&\qquad\quad P(z_j \text{ is first pivot chosen from } Z_{ij}) \\
&\quad = \quad \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&\quad = \quad \frac{2}{j-i+1}
\end{aligned}$$

(*) follow because the events are mutually exclusive

Now we have

$$\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(z_i \text{ is compared to } z_j) \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}
\end{aligned}$$

Let start by replacing $j - i$ with k:

$$\begin{aligned} E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\ &< \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k} = 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k} \\ &< 2 \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k} = 2 \sum_{i=1}^{n-1} O(\log n) = O(n \log n) \end{aligned}$$

Harmonic number $H_n = 1/1 + 1/2 + \ldots 1/n$.

$$H_n = \ln n + \mathcal{O}(1) = \Theta(\log n)$$

Result: Randomized-Partition yields expected (overall) running time of Quicksort of order $O(n \log n)$

Assignment Problem 5.1. (deadline: June 10, 5:30pm)
Show that expected running time of **Randomized-Quicksort** is $\Omega(n \log n)$. In fact it's enough to show that $E[X] = \Omega(n \log n)$.

Hint: From the lecture notes we note that

- $E[X] = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1}$; and
- $H_n \geq \ln n$.

Use these two facts to show that for some $c > 0$ and n_0, $E[X] \geq c.n \ln n$, for all $n \geq n_0$.

Note: a difference between **average** and **expected** running time:

- Average running time is the average *over all possible inputs*.
- Expected running time is, given some input, the average running time of your randomized algorithm on this input *over all possible random choices*.

Randomized-QuickSort — the first approach

- we want to randomly permute the input array
- we need to generate a random permutation in reasonable time (at most $\mathcal{O}(n \log n)$, but preferably $\mathcal{O}(n)$)

Permute by sorting

- assign to each element a random priority $\mathcal{P}[i]$
- sort the array by priorities:
  after sorting, if $\mathcal{P}[i]$ is the j-th smallest priority, then $A[i]$ will be in position j of the output

Example:
initial array $A = \{1, 2, 3, 4\}$
random priorities $\mathcal{P} = \{36, 3, 97, 19\}$
after sorting by priorities we get permutation
$A' = \{2, 4, 1, 3\}$

Permute-By-Sorting$(A[1 \ldots n])$

```
for i ← 1 to n do
    P[i] ← Random(1, n^3)
end for
sort A using P as sort keys
return A
```

the procedure takes $\Omega(n \log n)$ time (due to sorting)

with probability at least $1 - 1/n$ the keys generated are unique

assume, for simplicity, that the generated keys are unique

we should analyze the algorithm to prove that it generates all possible permutations of the input with **uniform distribution**

Analysis.

Fix a permutation $\pi \in S_n$. What is the probability that input will be permuted according to π?
($A[1]$ will be in position $\pi(1)$,
$A[2]$ in position $\pi(2)$,$\ldots$,
$A[n]$ in position $\pi(n)$)

That is: what's the probability that $\mathcal{P}[i]$ is the $\pi(i)$-th smallest priority for all i?

define **events**, $i = 1, \ldots, n$,

E_i is the event that $\mathcal{P}[i]$ is
the $\pi(i)$-th smallest priority

That is: what's the probability that all events occur?

$$\boxed{P(E_1 \cap E_2 \cap \cdots \cap E_n) =?}$$

Assignment Problem 5.2. (deadline: June 10, 5:30pm)
Show by mathematical induction that for any n and events $A_1, A_2, \ldots, A_n$ we have the equality:

$$P(A_1 \cap A_2 \cap \cdots \cap A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_2 \cap A_1) \cdots P(A_n|A_{n-1} \cap \cdots \cap A_2 \cap A_1)$$

- What is the probability $P(E_1)$, i.e., that $\mathcal{P}[1]$ is the $\pi(1)$-th smallest element?
  Since, each $\mathcal{P}[i]$ is chosen from the same distribution, each has equal chance that it's the $\pi(1)$-th smallest (uniform distribution). Hence, $P(E_1) = 1/n$.

- What is $P(E_2|E_1)$, i.e., the probability that $\mathcal{P}[2]$ is the $\pi(2)$-th smallest priority under assumption that $\mathcal{P}[1]$ is already fixed.
  $n-1$ priorities are not fixed, each of them can be $\pi(2)$-th smallest one. Uniform distribution, again. That is: the probability that it is $\mathcal{P}[2]$ is $1/(n-1)$.

- In general, if events $E_1, \ldots, E_i$ has happened, i.e., $\mathcal{P}[1], \ldots, \mathcal{P}[i]$ are already fixed then $n-i$ priorities are not fixed, and each of them can be $\pi(i+1)$-th smallest one.

$$\boxed{P(E_{i+1}|E_i \cap \ldots E_1) = 1/(n-i)}$$

Hence,

$$P(E_1 \cap E_2 \cap \cdots \cap E_n) =$$
$$P(E_1) \cdot P(E_2|E_1) \cdot P(E_3|E_2 \cap E_1) \cdots$$
$$P(E_n|E_{n-1} \cap \cdots \cap E_2 \cap E_1)$$
$$= \frac{1}{n} \cdot \frac{1}{n-1} \cdots \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{n!}$$

We have shown that probability that we get a fixed permutation of the input is $1/n!$, hence every permutation is *equally likely* produced — a **uniform distribution**.

Note: It's not enough to show that the probability that element $A[i]$ is permuted to a position j is $1/n$, i.e., that $P(E_i) = 1/n$.

Assignment Problem 5.3. (deadline: June 10, 5:30pm)
Prove that in the array $\mathcal{P}$ in procedure **Permute-By-Sorting**, the probability that all elements (priorities) are unique is exactly

$$\prod_{i=1}^{n}\left(1-\frac{i-1}{n^3}\right)$$

Then prove that this formula is greater than $1-1/n$.

Hints:

- Define the events

  E_i is the event that $\mathcal{P}[i]$ is different from $\mathcal{P}[1],\ldots,\mathcal{P}[i-1]$

  In fact, we are looking for probability $P(E_1\cap\cdots\cap E_n)$. Use the same technique as on the lecture, to compute this probability.

- For the second part, first show that the product is larger than $(1-1/n^2)^n$ and then use Binomial Theorem.

Assignment Problem 5.4. (deadline: June 10, 5:30pm)

Consider the following procedure for generating a uniform random permutation:

Permute-By-Cyclic$(A[1 \ldots n])$

$offset \leftarrow$ **Random**$(1, n)$
for $i \leftarrow 1$ **to** n **do**
    $dest \leftarrow i + offset$
    if $dest > n$ **then**
        $dest \leftarrow dest - n$
    end if
    $B[dest] \leftarrow A[i]$
end for
return B

Show that each element $A[i]$ has a $1/n$ probability of being permuted to any particular position in B. Is the resulting permutation (of the procedure) uniformly random?

Faster procedure for permuting

Permute-In-Place$(A[1 \ldots n])$

1: **for** $i \leftarrow 1$ **to** n **do**
2: swap $A[i] \leftrightarrow A[\textbf{Random}(i, n)]$
3: **end for**
4: return A

works in linear time $\mathcal{O}(n)$!

but does it really produce a uniform random permutation?

k**-permutation** — a sequence containing k elements of a set with n elements

there are $n!/(n-k)!$ possible k-permutations *loop invariant:*

- prior to the i-th iteration of the loop on lines 1–3, the subarray $A[1 \ldots i-1]$ contains any of $(i-1)$-permutations with probability $\dfrac{1}{n!/(n-i+1)!} = (n-i+1)!/n!$

Initialization: prior to the 1st iteration, the subarray is empty;
there is only one 0-permutation, the empty sequence, and hence the probability that the subarray contains the empty sequence is $1 = (n-1+1)!/n!$

Maintenance: assume that just before the i-th iteration, each possible $(i-1)$-permutation appears in $A[1 \ldots i-1]$ with probability $(n-i+1)!/n!$
we will show that after the i-th iteration, each possible i-permutation appears in $A[1 \ldots i]$ with probability $(n-i)!/n!$

Maintenance: *(continued)*

pick an i-permutation $\langle x_1, \ldots, x_{i-1}, x_i \rangle$

consider 2 events:

- E_1 — the first $i-1$ iterations has produced the $(i-1)$-permutation $\langle x_1, \ldots, x_{i-1} \rangle$ in $A[1 \ldots i-1]$
  by the loop invariant, $P(E_1) = (n-i+1)!/n!$
- E_2 — the i-th iteration puts elements x_i in position $A[i]$

the i-permutation $\langle x_1, \ldots, x_{i-1}, x_i \rangle$ is placed in $A[1 \ldots i]$ *if and only if* both, E_1 and E_2 occur

by Assignment 5.2,

$$P(E_1 \cap E_2) = P(E_1) \cdot P(E_2 | E_1)$$

- $P(E_2|E_1) = 1/(n-i+1)$

$$\begin{aligned} P(E_1 \cap E_2) &= P(E_1) \cdot P(E_2|E_1) \\ &= \frac{(n-i+1)!}{n!} \cdot \frac{1}{n-i+1} = \frac{(n-i)!}{n!} \end{aligned}$$

Termination: $i = n + 1$, i.e., each possible permutation (= n-permutation) appears in $A[1 \dots n]$ with probability $1/n!$

Hence, **Permute-In-Place** produces a uniform random permutation.