Algorithms for Data Science

CSOR W4246

Eleni Drinea
Computer Science Department

Columbia University

Randomized quicksort

Outline

Today

Pseudocode for randomized Quicksort

```
Randomized-Quicksort(A, left, right)
    if |A| = 0  then return //A is empty
    end if
    split = Randomized-Partition(A, left, right)
    Randomized-Quicksort(A, left, split − 1)
    Randomized-Quicksort(A, split + 1, right)
```

```
Randomized-Partition(A, left, right)
    b = random(left, right)
    swap(A[b], A[right])
    return Partition(A, left, right)
```

Subroutine `random`(i, j) returns a random number between i and j inclusive.

Expected running time of randomized Quicksort

- Let $T(n)$ be the expected running time of `Randomized-Quicksort`.
- We want to bound $T(n)$.
- `Randomized-Quicksort` differs from `Quicksort` only in how they select their pivot elements.

$\Rightarrow$ We will analyze `Randomized-Quicksort` based on `Quicksort` and `Partition`.

Pseudocode for Partition

`Partition`$(A, left, right)$

$pivot = A[right]$	line 1
$split = left - 1$	line 2
for $j = left$ to $right - 1$ **do**	line 3
if $A[j] \leq pivot$ **then**	line 4
`swap`$(A[j], A[split + 1])$	line 5
$split = split + 1$	line 6
end if	
end for	
`swap`$(pivot, A[split + 1])$	line 7
return $split + 1$	line 8

Few observations

1. *How many times is* `Partition` *called?*

Few observations

1. *How many times is* `Partition` *called?*
At most n.

2. Further, each `Partition` call spends some work
   1. outside the for loop

   2. inside the for loop

Few observations

1. *How many times is* `Partition` *called?*
At most n.

2. Further, each `Partition` call spends some work
   1. outside the for loop
      - every `Partition` spends constant work ouside the for loop
      - at most n calls to `Partition`

      $\Rightarrow$ total work outside the for loop in all calls to `Partition` is $O(n)$
   2. inside the for loop

Few observations

1. *How many times is* `Partition` *called?*
At most n.

2. Further, each `Partition` call spends some work
   1. outside the for loop
      - every `Partition` spends constant work ouside the for loop
      - at most n calls to `Partition`

      $\Rightarrow$ total work outside the for loop in all calls to `Partition` is $O(n)$
   2. inside the for loop
      - let X be the total number of comparisons performed at line 4 in all calls to `Partition`
      - each comparison may require some further constant work (lines 5 and 6)

      $\Rightarrow$ total work inside the for loop in all calls to `Partition` is $O(X)$

Towards a bound for $T(n)$

X = the total number of comparisons in **all** `Partition` calls.

The running time of `Randomized-Quicksort` is

$$O(n + X).$$

Since X is a random variable, we need $E[X]$ to bound $T(n)$.

Towards a bound for $T(n)$

X = the total number of comparisons in **all** `Partition` calls.

The running time of `Randomized-Quicksort` is

$$O(n + X).$$

Since X is a random variable, we need $E[X]$ to bound $T(n)$.

Fact 1.

Fix any two input items. During the execution of the algorithm, they may be compared at most once.

Towards a bound for $T(n)$

X = the total number of comparisons in **all** `Partition` calls.

The running time of `Randomized-Quicksort` is

$$O(n + X).$$

Since X is a random variable, we need $E[X]$ to bound $T(n)$.

Fact 1.

Fix any two input items. During the execution of the algorithm, they may be compared at most once.

Proof.

Comparisons are only performed with the *pivot* of each `Partition` call. After `Partiton` returns, *pivot* is in its final location in the output and will not be part of the input to any future recursive call. □

Simplifying the analysis

- There are n numbers in the input, hence $\binom{n}{2} = \frac{n(n-1)}{2}$ distinct (unordered) pairs of input numbers.
- From Fact 1, the algorithm will perform at most $\binom{n}{2}$ comparisons.
- *What is the expected number of comparisons?*

Simplifying the analysis

- There are n numbers in the input, hence $\binom{n}{2} = \frac{n(n-1)}{2}$ distinct (unordered) pairs of input numbers.
- From Fact 1, the algorithm will perform at most $\binom{n}{2}$ comparisons.
- *What is the expected number of comparisons?*

To simplify the analysis

- relabel the input as $z_1, z_2, \ldots, z_n$, where z_i is the i-th smallest number.
- **assume** that all input numbers are distinct; thus $z_i < z_j$, for $i < j$.

Writing X as the sum of indicator random variables

Let X_{ij} be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

Writing X as the sum of indicator random variables

Let X_{ij} be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by $\quad X = \sum_{1 \leq i < j \leq n} X_{ij}$.

Writing X as the sum of indicator random variables

Let X_{ij} be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by $X = \sum_{1 \leq i < j \leq n} X_{ij}$.

$E[X] = ?$

Writing X as the sum of indicator random variables

Let X_{ij} be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by $\quad X = \sum_{1 \leq i < j \leq n} X_{ij}.$

By linearity of expectation

$$E[X] = E\left[\sum_{1 \leq i < j \leq n} X_{ij} \right] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1]$$

Writing X as the sum of indicator random variables

Let X_{ij} be an indicator random variable such that

$$X_{ij} = \begin{cases} 1, & \text{if } z_i \text{ and } z_j \text{ are ever compared} \\ 0, & \text{otherwise} \end{cases}$$

The total number of comparisons is given by $\quad X = \sum_{1 \le i < j \le n} X_{ij}.$

By linearity of expectation

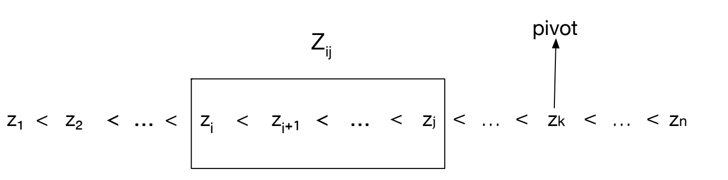
$$E[X] = E\left[\sum_{1 \le i < j \le n} X_{ij}\right] = \sum_{1 \le i < j \le n} E[X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1]$$

Goal: compute $\Pr[X_{ij} = 1]$, that is, the probability that two *fixed* items z_i and z_j are ever compared.

Fix two items z_i and z_j. When are they compared?

Notation: let $Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$

Consider the initial call `Partition`$(A, 1, n)$. Assume it picks z_k **outside** Z_{ij} as *pivot* (see figure below).



1. z_i and z_j are **not** compared in this call *(why?)*.
2. All items in Z_{ij} will be greater (or smaller) than z_k, so they will **all be input to the same subproblem** after `Partition`$(A, 1, n)$ returns.

In the first `Partition` with $pivot \in Z_{ij} = \{z_i, \ldots, z_j\}$

The first `Partition` call that picks its *pivot* from Z_{ij} determines if z_i, z_j are ever compared. Three possibilities:

1. $pivot = z_i$

2. $pivot = z_j$

3. $pivot = z_\ell$, for some $i < \ell < j$

In the first `Partition` with $pivot \in Z_{ij} = \{z_i, \ldots, z_j\}$

The first `Partition` call that picks its *pivot* from Z_{ij} determines if z_i, z_j are ever compared. Three possibilities:

1. $pivot = z_i$

   z_i is compared with every element in $Z_{ij} - \{z_i\}$, thus with z_j too. z_i is placed in its final location in the output and will not appear in any future calls to `Partition`.

2. $pivot = z_j$

   z_j is compared with every element in $Z_{ij} - \{z_j\}$, thus with z_i too. z_j is placed in its final location in the output and will not appear in any future recursive calls.

3. $pivot = z_\ell$, for some $i < \ell < j$

   z_i and z_j are never compared *(why?)*

So z_i and z_j are compared when ...

... either of them is chosen as *pivot* in that first Partition call that chooses its *pivot* element from Z_{ij}.

Now we can compute $\Pr[X_{ij} = 1]$:

$$\begin{aligned}\Pr[X_{ij} = 1] \quad = \quad & \Pr[z_i \text{ is chosen as } \textit{pivot} \text{ by the first } \texttt{Partition} \\ & \quad \text{that picks its } \textit{pivot} \text{ from } Z_{ij}, \ \ \textbf{or} \\ & \quad z_j \text{ is chosen as } \textit{pivot} \text{ by the first } \texttt{Partition} \\ & \quad \text{that picks its } \textit{pivot} \text{ from } Z_{ij}] \qquad (1)\end{aligned}$$

The union bound

Suppose we are given a set of events $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, and we are interested in the probability that **any** of them happens.

Union bound: Given events $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$, we have

$$\Pr\left[\bigcup_{i=1}^{n} \varepsilon_i\right] \leq \sum_{i=1}^{n} \Pr[\varepsilon_i].$$

Union bound for mutually exclusive events: Suppose that $\varepsilon_i \cap \varepsilon_j = \emptyset$ for each pair of events. Then

$$\Pr\left[\bigcup_{i=1}^{n} \varepsilon_i\right] = \sum_{i=1}^{n} \Pr[\varepsilon_i].$$

Computing the probability that z_i and z_j are compared

Since the two events in equation (1) are mutually exclusive, we obtain

$$\begin{aligned}
\Pr[X_{ij} = 1] &= \Pr[z_i \text{ is chosen as } \textit{pivot} \text{ by the first } \texttt{Partition} \\
&\qquad \text{call that picks its } \textit{pivot} \text{ from } Z_{ij}] \\
&+ \Pr[z_j \text{ is chosen as } \textit{pivot} \text{ by the first } \texttt{Partition} \\
&\qquad \text{call that picks its } \textit{pivot} \text{ from } Z_{ij}] \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}, \qquad (2)
\end{aligned}$$

since the set Z_{ij} contains $j - i + 1$ elements.

From $\Pr[X_{ij} = 1]$ to $E[X]$

$$
\begin{aligned}
E[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Pr[X_{ij} = 1] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= 2 \sum_{i=1}^{n-1} \sum_{\ell=2}^{n-i+1} \frac{1}{\ell} \qquad (3)
\end{aligned}
$$

Note that $\sum_{\ell=1}^{k} \frac{1}{\ell} = H_k$ is the k-th harmonic number, such that

$$\ln k \leq H_k \leq \ln k + 1 \qquad (4)$$

Hence $\sum_{\ell=2}^{n-i+1} \frac{1}{\ell} \leq \ln(n-i+1)$. Substituting in (3), we get

$$E[X] \leq 2 \sum_{i=1}^{n-1} \ln(n-i+1) \leq 2 \sum_{i=1}^{n-1} \ln n = O(n \ln n)$$

From $E[X]$ to $T(n)$

- Equations (3), (4) also yield a lower bound of $\Omega(n \ln n)$ for $E[X]$ *(show this!)*.

- Hence $E[X] = \Theta(n \ln n)$. Then the expected running time of `Randomized-Quicksort` is

$$T(n) = \Theta(n \ln n)$$