Randomized Algorithms

- Deterministic and (Las Vegas & Monte Carlo) Randomized Algorithms
- Probability Review
- Probabilistic Analysis of deterministic QUICK-SORT Algorithm
- RANDOMIZED-SELECT and RANDOMIZED-QUICK-SORT
- Max-Cut
- Min-Cut
- MAX-3-SAT and Derandomization
- Closest pair
- Hashing, Bloom filters, Streams, Sampling, Reservoir sampling, Sketch

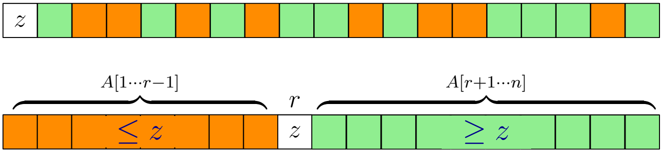
IMDAD ULLAH KHAN

Probabilistic Analysis of QUICKSORT

Algorithm Sorting A using PARTITION

function QUICKSORT(A)
  if $|A| \leq 1$ **then**
    return A
  $z \leftarrow A[1]$
  PARTITION(A, z)
  $r \leftarrow$ RANK(z, A)
  QUICKSORT($A[1 \ldots r-1]$)
  QUICKSORT($A[r+1 \ldots |A|]$)

function PARTITION(A, z)
  $i \leftarrow 1$ $j \leftarrow |A|$
  $r \leftarrow$ RANK(A, z)
  while $i < j$ **do**
    while **do**$A[i] < z$
      $i \leftarrow i + 1$
    while **do**$A[j] > z$
      $j \leftarrow j - 1$
    if $i \neq r$ AND $j \neq r$ **then**
      SWAP($A[i], A[j]$)

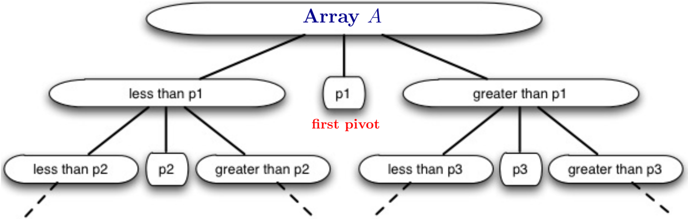


Probabilistic Analysis of QUICKSORT

Algorithm Sorting A using PARTITION

function QUICKSORT(A)
  if $|A| \leq 1$ **then**
    return A
  $z \leftarrow A[1]$
  PARTITION(A, z)
  $r \leftarrow$ RANK(z, A)
  QUICKSORT($A[1 \ldots r-1]$)
  QUICKSORT($A[r+1 \ldots |A|]$)

function PARTITION(A, z)
  $i \leftarrow 1$ $j \leftarrow |A|$
  $r \leftarrow$ RANK(A, z)
  while $i < j$ **do**
    while **do**$A[i] < z$
      $i \leftarrow i + 1$
    while **do**$A[j] > z$
      $j \leftarrow j - 1$
    if $i \neq r$ AND $j \neq r$ **then**
      SWAP($A[i], A[j]$)



Algorithm Sorting A using PARTITION

function QUICKSORT(A)
  if $|A| \leq 1$ **then**
    return A
  $z \leftarrow A[1]$
  PARTITION(A, z)
  $r \leftarrow$ RANK(z, A)
  QUICKSORT($A[1 \dots r-1]$)
  QUICKSORT($A[r+1 \dots |A|]$)

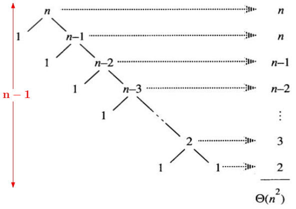
function PARTITION(A, z)
  $i \leftarrow 1$ $j \leftarrow |A|$
  $r \leftarrow$ RANK(A, z)
  while $i < j$ **do**
    while **do**$A[i] < z$
      $i \leftarrow i + 1$
    while **do**$A[j] > z$
      $j \leftarrow j - 1$
    if $i \neq r$ AND $j \neq r$ **then**
      SWAP($A[i], A[j]$)

$T(n)$: runtime of QUICKSORT on $|A| = n$

Worst case: pivot is always min or max of A

$$T(n) = \begin{cases} T(n-1) + T(0) + O(n) & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

$\mathbf{T(n) = O(n^2)}$



Algorithm Sorting A using PARTITION

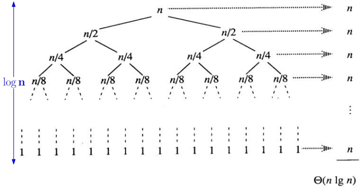
```
function QUICKSORT(A)
    if |A| ≤ 1 then
        return A
    z ← A[1]
    PARTITION(A, z)
    r ← RANK(z, A)
    QUICKSORT(A[1 ... r − 1])
    QUICKSORT(A[r + 1 ... |A|])
```

```
function PARTITION(A, z)
    i ← 1      j ← |A|
    r ← RANK(A, z)
    while i < j do
        while doA[i] < z
            i ← i + 1
        while doA[j] > z
            j ← j − 1
        if i ≠ r AND j ≠ r then
            SWAP(A[i], A[j])
```

$T(n)$: runtime of QUICKSORT on $|A| = n$

Best case: pivot is always median of array

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + O(n) & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$



$\mathbf{T(n) = O(n \log n)}$

What is the average case running time of QUICKSORT?

Average over what?

In probabilistic analysis we use probability in the analysis of a deterministic algorithm

We have or assume knowledge about the distribution of the input

The average is over the distribution

For QUICKSORT:

Assume all permutations of n numbers in A are equally likely

- ranks of numbers in A is a uniform random permutation of $[1 \cdots n]$

Algorithm Sorting A using PARTITION

```
function QUICKSORT(A)
    if |A| ≤ 1 then
        return A
    z ← A[1]
    PARTITION(A, z)
    r ← RANK(z, A)
    QUICKSORT(A[1 . . . r − 1])
    QUICKSORT(A[r + 1 . . . |A|])
```

```
function PARTITION(A, z)
    i ← 1      j ← |A|
    r ← RANK(A, z)
    while i < j do
        while doA[i] < z
            i ← i + 1
        while doA[j] > z
            j ← j − 1
        if i ≠ r AND j ≠ r then
            SWAP(A[i], A[j])
```

An element of A can be chosen as pivot at most once

- All subsequent processing is done on the two subarrays

Algorithm Sorting A using PARTITION

```
function QUICKSORT(A)
    if |A| ≤ 1 then
        return A
    z ← A[1]
    PARTITION(A, z)
    r ← RANK(z, A)
    QUICKSORT(A[1 . . . r − 1])
    QUICKSORT(A[r + 1 . . . |A|])
```

```
function PARTITION(A, z)
    i ← 1      j ← |A|
    r ← RANK(A, z)
    while i < j do
        while A[i] < z do
            i ← i + 1
        while A[j] > z do
            j ← j − 1
        if i ≠ r AND j ≠ r then
            SWAP(A[i], A[j])
```

Elements of A are compared to pivots only

- No comparison in the outer function
- In PARTITION elements are compared only with z (the pivot)

Algorithm Sorting A using PARTITION

```
function QUICKSORT(A)
    if |A| ≤ 1 then
        return A
    z ← A[1]
    PARTITION(A, z)
    r ← RANK(z, A)
    QUICKSORT(A[1 . . . r − 1])
    QUICKSORT(A[r + 1 . . . |A|])
```

```
function PARTITION(A, z)
    i ← 1      j ← |A|
    r ← RANK(A, z)
    while i < j do
        while  A[i] < z  do
            i ← i + 1
        while  A[j] > z  do
            j ← j − 1
        if i ≠ r AND j ≠ r then
            SWAP(A[i], A[j])
```

A pair of elements of A are compared only when one of them is a pivot

- Comparisons always involve pivot

Algorithm Sorting A using PARTITION

```
function QUICKSORT(A)
    if |A| ≤ 1 then
        return A
    z ← A[1]
    PARTITION(A, z)
    r ← RANK(z, A)
    QUICKSORT(A[1 ... r − 1])
    QUICKSORT(A[r + 1 ... |A|])
```

```
function PARTITION(A, z)
    i ← 1      j ← |A|
    r ← RANK(A, z)
    while i < j do
        while A[i] < z do
            i ← i + 1
        while A[j] > z do
            j ← j − 1
        if i ≠ r AND j ≠ r then
            SWAP(A[i], A[j])
```

A pair of elements of A are compared at most once

- After a comparison the two elements always go to different parts

- Let the sorted order of elements of A be $z_1, z_2, \ldots, z_n$
- Z_{ij} : elements between z_i and z_j (inclusive) $\triangleright$ $|Z_{ij}| = j - i + 1$

$$X_{ij} = \begin{cases} 1 & \text{if } z_i \text{ is compared with } z_j \\ 0 & \text{else} \end{cases}$$

Comparison can be at anytime of the execution, not in a specific call

Total number of comparison (through execution of the algorithm) is

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

sum over all possible pairs

$$E(X) = E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E\left[X_{ij}\right]$$

- Let the sorted order of elements of A be $z_1, z_2, \ldots, z_n$
- Z_{ij} : elements between z_i and z_j (inclusive) $\triangleright$ $|Z_{ij}| = j - i + 1$

$$E(X) = E\Big[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\Big] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E\left[X_{ij}\right]$$

Consider the sequence Z_{ij} : $z_i, z_{i+1}, \ldots, \ldots, z_j$

Initially they are all in the same array A

- They split only when some z_k for $i \leq k \leq j$ is pivot
- z_i and z_j are compared only if **they are in the same (sub) array and either z_i and z_j is pivot**
- If the first pivot in Z_{ij} is other than z_i and z_j, then Z_{ij} is split and z_i and z_j never get compared $\triangleright$ $X_{ij} = 0$

- Let the sorted order of elements of A be $z_1, z_2, \ldots, z_n$
- Z_{ij} : elements between z_i and z_j (inclusive) $\triangleright$ $|Z_{ij}| = j - i + 1$

$$E(X) = E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}]$$

Consider the sequence Z_{ij} : $z_i, z_{i+1}, \ldots, \ldots, z_j$

z_i and z_j are compared **iff** z_i or z_j is the first pivot among numbers in Z_{ij}

$$E[X_{ij}] = Pr[z_i \text{ or } z_j \text{ is the first among } Z_{ij} \text{ chosen as pivot}]$$

z_i (or z_j) will be the pivot if it is the first one (among them) and ....

The probability that z_i is before all in Z_{ij} is $\frac{1}{j-i+1}$

z_i and z_j are compared if and only if among all numbers in Z_{ij}, either z_i or z_j is the first pivot

$$E[X_{ij}] = Pr[z_i \text{ or } z_j \text{ is the first among } Z_{ij} \text{ chosen as pivot}]$$

The probability that z_i is before all in Z_{ij} is $\frac{1}{j-i+1}$

$Pr[z_i$ or z_j is the first among Z_{ij} chosen as pivot$] = \frac{2}{j-i+1}$

$$E(X) = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{ij}] = \frac{2}{j-i+1}$$

Substitute $k = j - i$

$$E(X) = \sum_{i=1}^{n}\sum_{k=1}^{n-i} = \frac{2}{k+1} < \sum_{i=1}^{n}\sum_{k=1}^{n} \frac{2}{k} \leq 2n\log n$$

Probabilistic Analysis of QUICKSORT

- Cannot guarantee randomly ordered input array
- Permute array to make it a random permutation
  - ▷ Generating a random permutation is an interesting exercise
- Worst case is less likely if pivot is the median of 3 or 4 elements
- Average/worst/best case is $O(n \log n)$ if pivot is always the median
- RANDOMIZED-QUICKSORT chooses a random pivot

```
function RAND-QUICKSORT(A)
  if |A| ≤ 1 then
    return A
  randIndex ← RANDOM(1, |A|)
  z ← A[randIndex]
  PARTITION(A, z)
  r ← RANK(z, A)
  RAND-QUICKSORT(A[1 ... r − 1])
  RAND-QUICKSORT(A[r + 1 ... |A|])
```

Analysis is exactly the same with Indicator random variables