COMP3711: Design and Analysis of Algorithms

Tutorial 4

HKUST

Question 1: Matching Nuts and Bolts

See External Powerpoint Slides.

Question 2

In class we learned the Randomized Selection algorithm.
Via a geometric series analysis we showed that it runs in
$O(n)$ expected time.

In this tutorial you will rederive this result in a different way,
via the *Indicator Random Variable* method used to analyze Quicksort.

Recall the Randomized Selection algorithm to find the k-th **smallest element**.

Pick a random pivot, partition to divide the array into 3 parts:

left subarray, pivot item, right subarray.

Compare to pivot and then either stop immediately or recursively solve the problem in the left **OR** the right part of the array.

- As in Quicksort, denote the elements in **sorted order** by $z_1, \cdots, z_n$.
  (so we are searching for z_k)
- We use same random model for choosing the pivot as in Quicksort.

Question 2 (iii)

Define

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

(I) First Prove the following three facts

(a) $i \le k \le j$: $\Pr[X_{ij} = 1] = 2/(j - i + 1)$.

(b) $i < j < k$: $\Pr[X_{ij} = 1] = 2/(k - i + 1)$.

(c) $k < i < j$: $\Pr[X_{ij} = 1] = 2/(j - k + 1)$.

(II) By the indicator random variable technique, expected total number of comparisons is

$$\sum_{i<j} E[X_{ij}] = \sum_{i<j} \Pr[X_{ij} = 1]$$

(III) Then use (I) to show that this sum is $O(n)$.

Solution 2

Calculate $E\left[X_{ij}\right] = \Pr[X_{ij} = 1]$, i.e,

Probability $[z_i$ and z_j $(i < j)$ are compared by *RandSelect* when searching for $z_k]$.

Consider the following 3 cases separately:

(a) $i \le k \le j,$ *(b)* $k > j,$ *(c)* $k < i,$

Analysis will be variation of analysis of Quicksort.
The three cases will need to be approached slightly differently.

(a) $i \le k \le j$**: Will consider the first pivot chosen in** $\{z_i, \cdots, z_j\}$**.**

(b) $i < j < k$**: Will consider the first pivot chosen in**$\{z_i, \cdots, z_j, \cdots, z_k\}$**.**

(c) $k < i < j$**: Will the first pivot chosen in** $\{z_k, \cdots, z_i, \cdots, z_j\}$**.**

Solution 2 (cont)

Calculate $E\left[X_{ij}\right] = \Pr[X_{ij} = 1]$, i.e,

Probability $[z_i$ and z_j $(i < j)$ are compared by *RandSelect* when searching for $z_k]$.

(a) $i \le k \le j$: This is same analysis as in Quicksort:

Consider the first pivot chosen in $\{z_i, \cdots, z_j\}$:

If the pivot is z_i or z_j , then z_i and z_j will be compared.

If the pivot is in $\{z_{i+1}, \cdots, z_{j-1}\}$, then they will not be compared.

So, only two choices out of the $j - i + 1$ choices cause z_i, z_j to be compared and

$$\Pr\left[X_{ij} = 1\right] = 2/(j - i + 1).$$

Solution 2 (cont)

Calculate $E[X_{ij}] = \Pr[X_{ij} = 1]$, i.e,

Probability $[z_i$ and z_j $(i < j)$ are compared by *RandSelect* when searching for $z_k]$.

(a) $i \le k \le j$: $\Pr[X_{ij} = 1] = 2/(j - i + 1)$.

(b) $i < j < k$: This is a bit different than Quicksort

Consider the first pivot chosen in$\{z_i, \cdots, z_j, \cdots, z_k\}$. Call this z_t.

(i) If $t = i$ or $t = j$ then one of z_i, z_j is the pivot and the other one is compared to that pivot by the partition procedure.

(ii) If $i < t < j$ then, after partitioning around z_t, the side containing z_i is thrown away and z_i is never used again $\Rightarrow$ z_i and z_j are never compared.

(iii) If $j < t \le k$ then, after partitioning around z_t , *both* z_i and z_j are on the side that gets thrown away
$\Rightarrow z_i$ and z_j are never involved in *any* further comparisons. So

$$\Pr[X_{ij} = 1] = 2/(k - i + 1).$$

Solution 2 (cont)

Calculate $E[X_{ij}] = \Pr[X_{ij} = 1]$, i.e,

Probability $[z_i$ and z_j $(i < j)$ are compared by *RandSelect* when searching for $z_k]$.

(a) $i \le k \le j$: $\Pr[X_{ij} = 1] = 2/(j - i + 1)$.

(b) $i < j < k$: $\Pr[X_{ij} = 1] = 2/(k - i + 1)$.

(c) $k < i < j$: Let z_t be the first pivot chosen in $\{z_k, \cdots, z_i, \cdots, z_j\}$.

(i) If $t = i$ or $t = j$ then one of z_i, z_j is the pivot and the other one is compared to that pivot by the partition procedure.

(ii) If $i < t < j$, then, after partitioning around z_t, the side containing z_j is thrown away and z_j is never used again $\Rightarrow z_i$ and z_j are never compared.

(iii) If $k \le t < i$, then, after partitioning around z_t,
both z_i and z_j are on the side that gets thrown away
$\Rightarrow z_i$ and z_j are never involved in *any* further comparisons.

$$\Pr[X_{ij} = 1] = 2/(j - k + 1).$$

Solution 2 (cont)

(I) Define $X_{ij} = 1$ if z_i and z_j are compared by the algorithm, and 0 otherwise. We have proven the three cases below:

(a) $i \leq k \leq j$: $\Pr[X_{ij} = 1] = 2/(j - i + 1)$.

(b) $i < j < k$: $\Pr[X_{ij} = 1] = 2/(k - i + 1)$.

(c) $k < i < j$: $\Pr[X_{ij} = 1] = 2/(j - k + 1)$.

(II) By the indicator random variable technique, the expected total number of comparisons is

$$\sum_{i<j} E[X_{ij}] = \sum_{i<j} \Pr[X_{ij} = 1]$$

For each of these different cases, we will show that the sum of the probabilities is $O(n)$, so the total sum of all of the probabilities is $O(n)$.

Solution 2 (cont)

The probabilities in case (a) $i \le k \le j$ $\Pr[X_{ij} = 1] = 2/(j - i + 1)$ are:

i:	1	2	3	4	$\cdots$	$k-2$	$k-1$	k
$j = k$	$\frac{2}{k}$	$\frac{2}{k-1}$	$\frac{2}{k-2}$	$\frac{2}{k-3}$	$\cdots$	$\frac{2}{3}$	$\frac{2}{2}$	$\frac{2}{1}$
$j = k+1$	$\frac{2}{k+1}$	$\frac{2}{k}$	$\frac{2}{k-1}$	$\frac{2}{k-2}$	$\cdots$	$\frac{2}{4}$	$\frac{2}{3}$	$\frac{2}{2}$
$j = k+2$	$\frac{2}{k+2}$	$\frac{2}{k+1}$	$\frac{2}{k}$	$\frac{2}{k-1}$	$\cdots$	$\frac{2}{5}$	$\frac{2}{4}$	$\frac{2}{3}$
$\cdots$	$\cdots$							$\cdots$
$j = n$	$\frac{2}{n}$	$\frac{2}{n-1}$	$\frac{2}{n-2}$	$\frac{2}{n-3}$	$\cdots$			$\frac{2}{n-k+1}$

Each diagonal sums up to $O(1)$.
There are $O(n)$ diagonals, so the total is $O(n)$.

Solution 2 (cont)

The probabilities in case (b) $i < j < k$ $\Pr[X_{ij} = 1] = 2/(k - i + 1)$ are:

i:	1	2	3	4	$\cdots$	$k-2$
$j = 2$	$\frac{2}{k}$					
$j = 3$	$\frac{2}{k}$	$\frac{2}{k-1}$				
$j = 4$	$\frac{2}{k}$	$\frac{2}{k-1}$	$\frac{2}{k-2}$			
$\cdots$	$\cdots$					
$j = k-1$	$\frac{2}{k}$	$\frac{2}{k-1}$	$\frac{2}{k-2}$	$\frac{2}{k-3}$	$\cdots$	$\frac{2}{3}$

Each column sums up to $O(1)$.
There are $O(n)$ columns, so the total is $O(n)$.

Solution 2 (cont)

The probabilities in case (c) $i < j < k$: $\Pr[X_{ij} = 1] = 2/(k - i + 1)$ are:

i:	$k+1$	$k+2$	$k+3$	$k+4$	$\cdots$	$n-1$
$j = k+2$	$\frac{2}{3}$					
$j = k+3$	$\frac{2}{4}$	$\frac{2}{4}$				
$j = k+4$	$\frac{2}{5}$	$\frac{2}{5}$	$\frac{2}{5}$			
$\cdots$	$\cdots$					
$j = n$	$\frac{2}{n-k+1}$	$\frac{2}{n-k+1}$	$\frac{2}{n-k+1}$	$\frac{2}{n-k+3}$	$\cdots$	$\frac{2}{n-k+1}$

Each row sums up to $O(1)$.
There are $O(n)$ rows, so the total is $O(n)$.
Thus, the total sum over all three cases is still $O(n)$.

Question 3

The analysis of the expected running time of randomized quicksort in the lecture notes assumes that all element values are distinct. In this problem, we examine what happens when they are not.

(a) Suppose that all element values are equal.
What would be randomized quicksort's running time in this case?

(b) The PARTITION procedure taught returns an index q such that each element of $A[p \cdots q-1]$ is $\leq$ $A[q]$ and each element of $A[q+1 \cdots r\ ]$ is > $A[q]$.

Modify PARTITION to produce a new procedure PARTITION' (A, p, r), which permutes the elements of $A[p \cdots r]$ and returns two indices q, t, where $p \leq q \leq t \leq r$, such that

- all elements of $A[q \cdots t]$ are equal,
- each element of $A[p \cdots q-1]$ is less than $A[q]$, and
- each element of $A[t+1 \cdots r]$ is greater than $A[q]$

Like PARTITION, your PARTITION' procedure should take $\Theta(r-p)$ time.

Question 3 (cont)

(c) Modify the QUICKSORT procedure to produce QUICKSORT’(A, p, r) that calls PARTITION’ and recurses only on partitions of elements not known to be equal to each other.

(d) ADVANCED PROBLEM
Our analysis of QUICKSORT in class assumed that all elements were distinct. Using QUICKSORT’, how would you adjust the analysis (binary tree plus indicator random variables) in the lecture notes to avoid the assumption that all elements are distinct?

Solution 3

(a) Suppose that all element values are equal.
What would be randomized quicksort's running time in this case?

The partition procedure will always partition the elements into two subsets where one subset contains $n-1$ elements and the other subset contains 0 elements because all the element values are the same. So the running time is

$$T(n) = T(n-1) + \Theta(n)$$

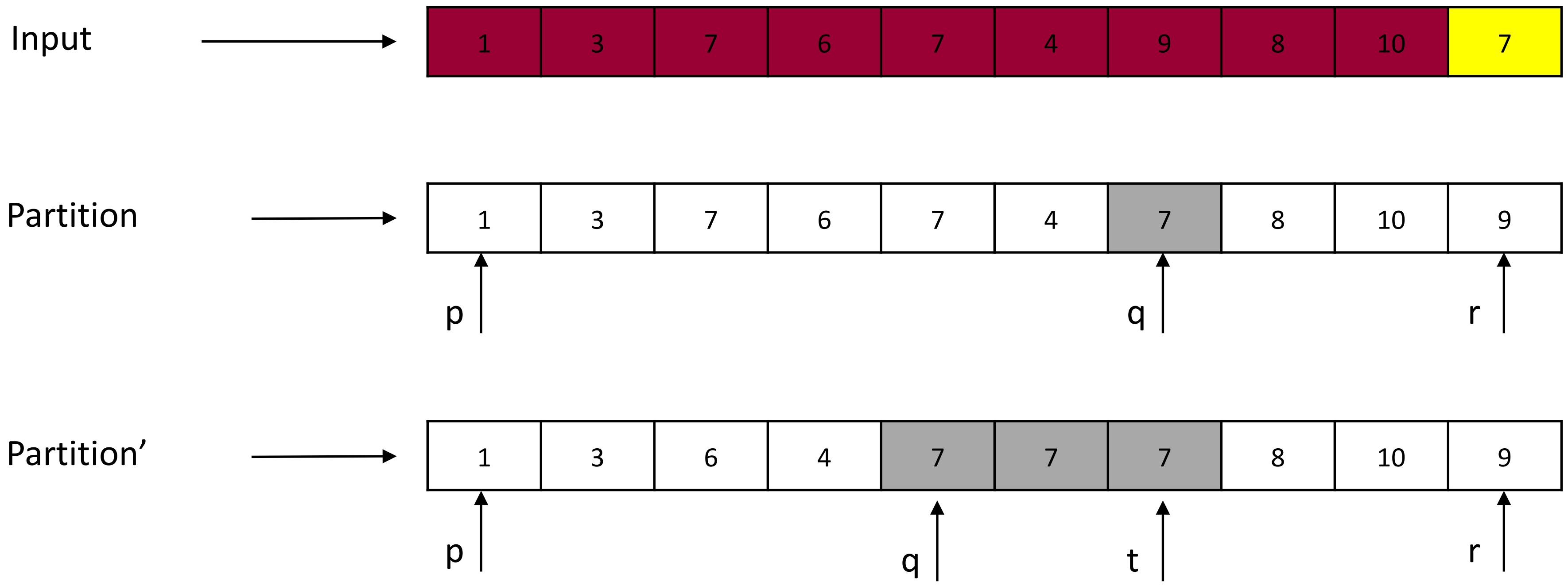
which is $\Theta(n^2)$.

(b) The PARTITION procedure taught returns an index q such that each element of $A[p \cdots q-1]$ is $\leq$ $A[q]$ and each element of $A[q+1 \cdots r]$ is > $A[q]$.

Modify PARTITION to produce a new procedure PARTITION' (A, p, r), which permutes the elements of $A[p \cdots r]$ and returns two indices q, t, where $p \leq q \leq t \leq r$, such that

- all elements of $A[q \cdots t]$ are equal,
- each element of $A[p \cdots q-1]$ is less than $A[q]$, and
- each element of $A[t+1 \cdots r]$ is greater than $A[q]$

Like PARTITION, your PARTITION' procedure should take $\Theta(r-p)$ time.



Solution 3

(b) PARTITION’(A, p, r):

```
    $i$ = RANDOM$(p, r)$
    exchange $A[i]$ with $A[r]$

    $x \leftarrow A[r]$;   $i \leftarrow p - 1$;   $k \leftarrow p - 1$
    % During execution, all items in $A[i+1 \cdots k]$ will have partition value $x$
    % If $i + 1 < k$, no values so far have partition value $x$

    for $j \leftarrow p$ to $r - 1$
        if $A[j] = x$ then
            $k \leftarrow k + 1$
            exchange $A[k]$ with $A[j]$
        else if $A[j] < x$ then
            $i \leftarrow i + 1$;  $k \leftarrow k + 1$
            exchange $A[k]$ with $A[j]$
            exchange $A[i]$ with $A[k]$

    exchange $A[k+1]$ with $A[r]$ % $A[i+1 \cdots k]$ will have partition value $x$
    return $(i + 1, k + 1)$
```

Solution 3

(c) Modify the QUICKSORT procedure to produce QUICKSORT’(A, p, r) that calls PARTITION’ and recurses only on partitions of elements not known to be equal to each other.

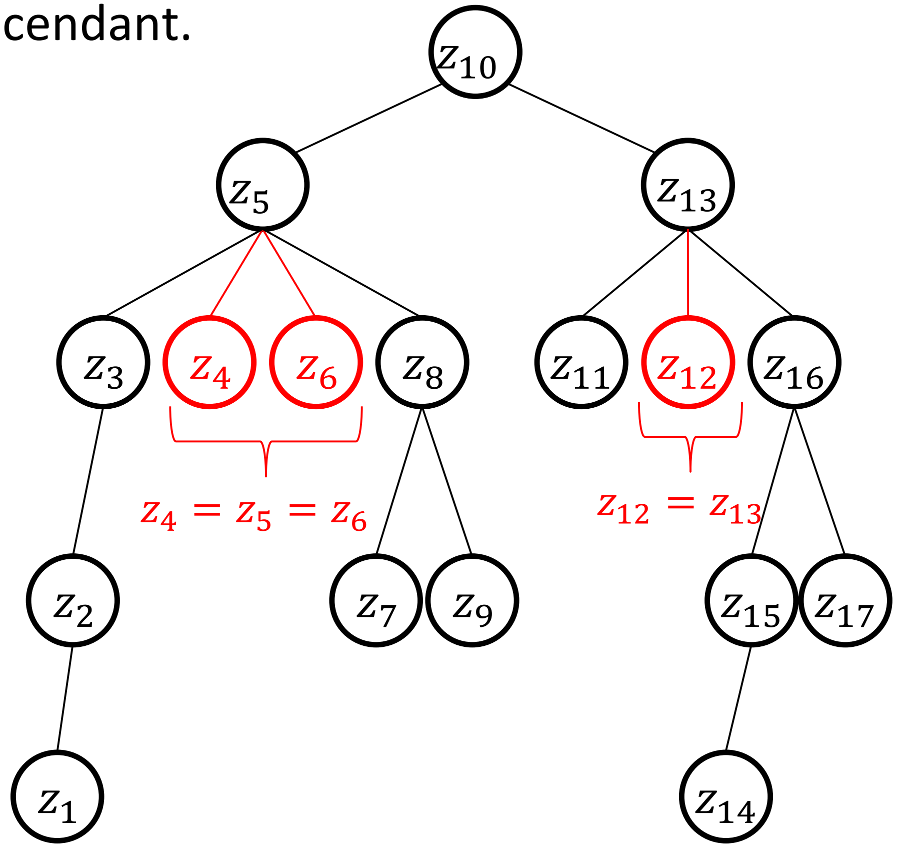
```
QUICKSORT’$(A, p, r)$:
    if $p \geq r$ then return
    $(q, t)$ = PARTITION’$(A, p, r)$
    QUICKSORT’$(A, p, q - 1)$
    QUICKSORT’$(A, t + 1, r)$
```

Solution 3

(d) Advanced Material: Analyze Partition’

Relabel the elements from small to large as $z_1, z_2, \cdots, z_n$, breaking ties (among equal items) arbitrarily.

We use a multi-ary tree representation which is similar to the binary tree representation except that the middle children of a pivot correspond to the elements equal to the pivot. Similarly, two elements are compared iff they are ancestor-descendant.



Solution 3

As in the analysis in class let z_i, z_j be any pair of elements and $Z = \{z_i, \cdots, z_j\}$.

In our previous analysis of Quicksort (distinct elements) we said that the items in Z were all in the same subarray (and therefore not compared to each other) until one of them was chosen as a pivot.

Every item had equal chance of being chosen as a pivot and since z_i would be compared to z_j if and only if one of z_i, z_j was chosen, z_i, z_j had probability $2/(j - i + 1)$ of being compared with each other.

Solution 3

In this new case let z_i' be the leftmost item equal to z_i (which might be z_i) and z_j' the rightmost item equal to z_j (which might be z_j).
Set $Z' = \{z_i', \cdots, z_j'\}$.

Until one of the items in Z' is chosen as a pivot all of the items in Z' are in the same subarray.

After one of the items in Z' is chosen as a pivot, z_i and z_j are no longer in the same subarray and will never be compared later.

Furthermore, when one of Z' is chosen as a pivot, z_i and z_j are compared to each other at if and only if one of them was chosen as a pivot.

Every item in Z' is equally likely to be chosen as a pivot, so

the probability of z_i or z_j being chosen is

$$\frac{2}{j' - i' + 1} \le \frac{2}{j - i + 1}$$

Solution 3

Let $X_{ij} = 1$ if z_i is compared with z_j.

From the previous slide

$$E(X_{ij}) = \Pr[z_i \text{ and } z_j \text{ are compared}] \leq 2/(j - 1 + 1)$$

Thus

$$E(\#\text{of comparisons}) = \sum_{i<j} E(X_{ij})$$

$$\leq \sum_{i<j} 2/(j - i + 1) = O(n \log n)$$

Where $\sum_{i<j} 2/(j - i + 1) = O(n \log n)$ was shown in class.

Question 4

Consider the heap implementation of a Priority Queue shown in class that keeps its items in an Array $A[]$.

Let Decrease-Key(i, x) be the operation that compares x to $A[i]$:

If $x \geq A[i]$, it does nothing.

If $x < A[i]$, it sets $A[i] = x$ and, if necessary, fixes $A[]$ so that it remains a Heap.

Show how to implement Decrease-Key(i, x) in $O(\log n)$ time, where n is the number of items in the Heap.

Note: We will use the operation Decrease-Key(i, x) later in the semester.

Solution 4

Recall the code for inserting an item into a heap. We inserted the item into location i which was the LAST item in the heap and then bubbled it up to its proper place.

The solution to this problem uses almost exactly the same code as the insertion except that i might no longer be the last value.

If we do change the value in $A[i]$ to be $A[i] = x$, everything *below* node i still satisfies the heap condition and node i satisfies the heap condition (because we've LOWERED its value).

The only problem might be that ancestors of i might have values that are **greater** than $A[i]$.
We can fix this by *bubbling* $A[i]$ up to its proper value.

Solution 4

```
begin
    if $x < A[i]$
        $A[i] = x$;
    $j = i$;
    while $A[j] < A[\lfloor \frac{j}{2} \rfloor]$ and $j > 1$ do
        // $A[j]$ is less than its parent
        Swap $A[j]$ and $A[\lfloor \frac{j}{2} \rfloor]$; // Bubble Up
        $j = \lfloor \frac{j}{2} \rfloor$
    end
end
```

See external PowerPoint for example