GUC German University in Cairo

CSEN 703 - Analysis and Design of Algorithms

Lecture 5 - Quick Sort

Dr. Nourhan Ehab
nourhan.ehab@guc.edu.eg

Department of Computer Science and Engineering
Faculty of Media Engineering and Technology

Outline

Quick Sort - D&C Approach

GUC German University in Cairo

- Divide: Partition $A[p, \ldots, r]$ into two (possibly empty) arrays $A[p, \ldots, q-1]$ and $A[q+1, \ldots, r]$:
  - each element in $A[p, \ldots, q-1] \leq A[q]$
  - each element in $A[q+1, \ldots, r] \geq A[q]$
- Conquer: The subarrays by sorting them.
- Combine: No work needed! Array already sorted.

Quick sort - Pseudo Code

```
1 QuickSort(A, p, r)
2 if p < r then
3 |   q = partition(A, p, r);
4 |   QuickSort(A, p, q − 1) ;
5 |   QuickSort(A, q + 1, r) ;
6 end
```

The complexity lies in the partition procedure which will rearrange the array such that all elements before the pivot has smaller values and all elements after the pivot has bigger values.

Ross was Probably Good at Quick Sort!

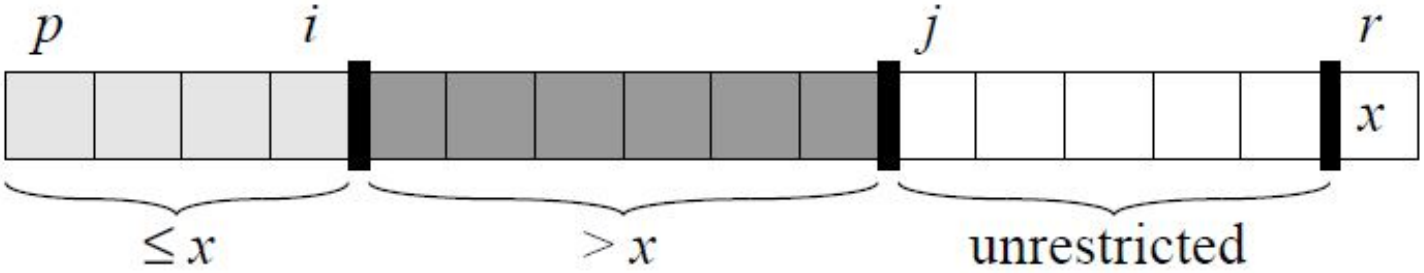
Quick sort - Partition

```
1  Partition(A, p, r)
2  pivot = A[r] ;
3  i = p − 1 ;
4  for j = p to r − 1 do
5  |   if A[j] ≤ pivot then
6  |   |   i + + ;
7  |   |   Exchange A[i] and A[j] ;
8  |   end
9  end
10 Exchange A[i + 1] and A[r] ;
11 return i+1;
```

Trace it on $[8, 1, 6, 4, 0, 3, 9, 5]$.

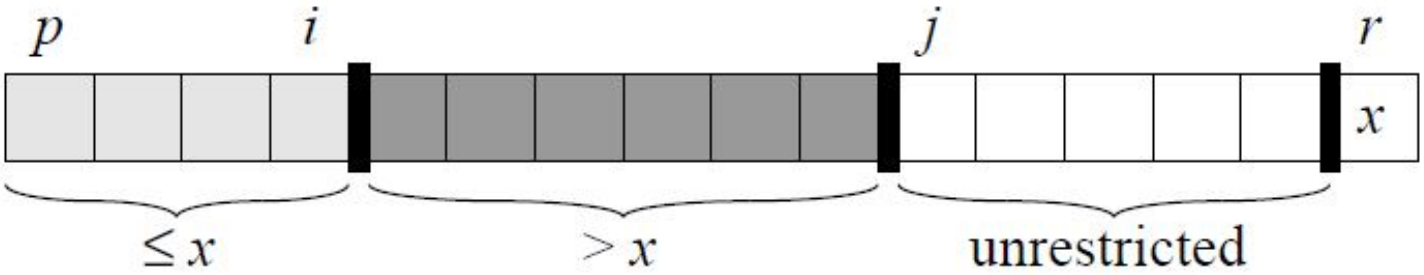
Quick Sort

Quick Sort - Correctness



- Correctness is based on the correctness of partition.

Quick Sort - Correctness

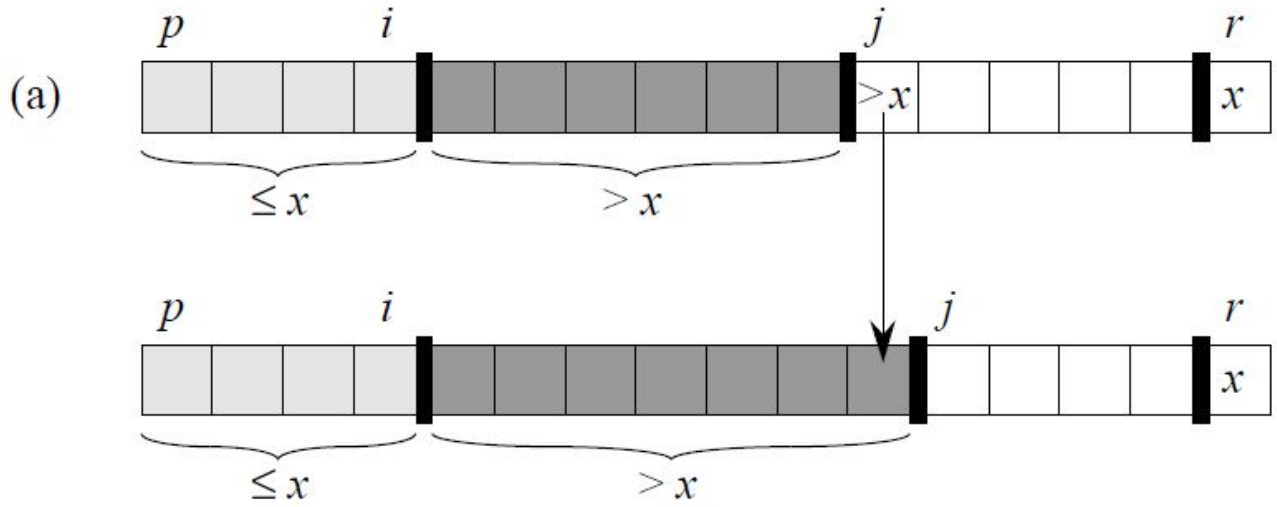


- Correctness is based on the correctness of partition.
- Loop Invariant:
  At the beginning of each iteration of the loop of lines 4–9, for any array index k:
  - If $p \leq k \leq i$, $then A[k] \leq pivot$.
  - If $i+1 \leq k \leq j-1$, then $A[k] > pivot$.
  - If $k = pivot$, then $A[k] = pivot$.
- The indices between j and $r-1$ are not covered by any of the three cases, and the values in these entries have no particular relationship to the pivot.

Quick Sort - Correctness Initialization

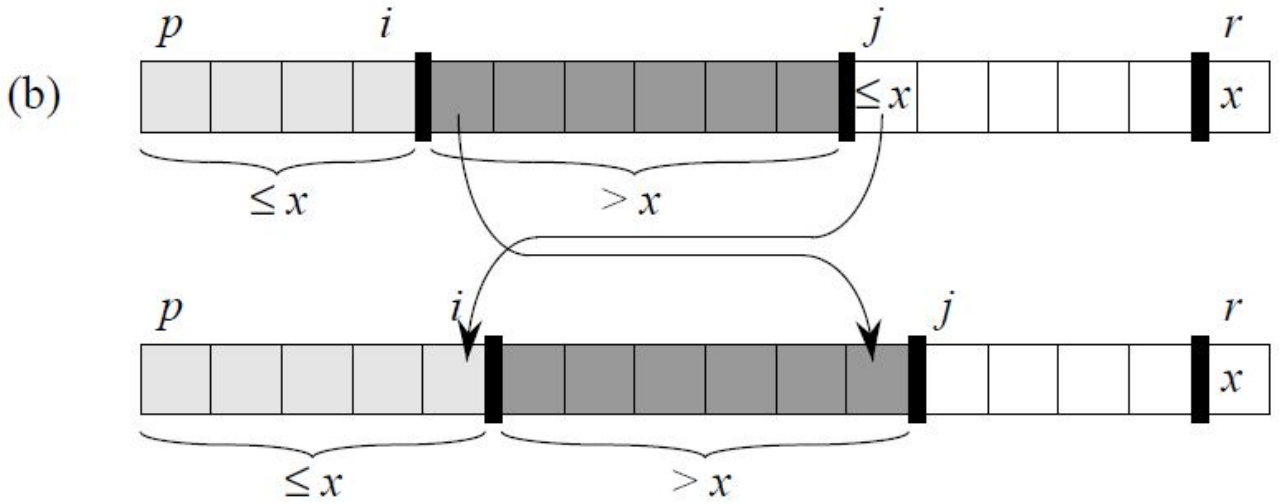
1. Initialization: Prior to the first iteration, $i = p - 1$ and $j = p$. Because no values lie between p and i and no values lie between $i + 1$ and $j - 1$, the first two conditions of the loop invariant are trivially satisfied. The assignment in line 2 satisfies the third condition.

Quick Sort - Correctness Maintenance 1



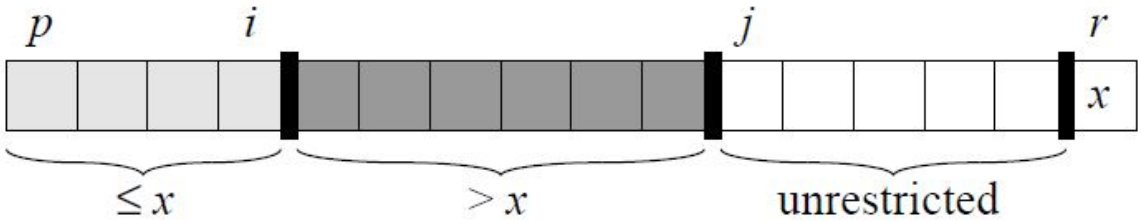
2. Maintenance: We have two cases.
   a. If $A[j] > x$, the only action in the loop is to increment j . After j is incremented, condition 2 holds for $A[j-1]$ and all other entries remain unchanged.

Quick Sort - Correctness Maintenance 2



2. **Maintenance:** We have two cases.
   b. If $A[j] \leq x$, the loop increments i , swaps $A[i]$ and $A[j]$, then and then increments j . Because of the swap, we now have that $A[i] \leq x$, and Condition 1 is satisfied. Similarly, we also have that $A[j-1] > x$, since the item that was swapped into $A[j-1]$ is greater than x.

Quick Sort - Termination



2. Termination: At termination, $j = r$. Therefore, every entry in the array is in one of the three sets described by the invariant, and we have partitioned the values in the array into three sets: those less than or equal to x, those greater than x, and a singleton set containing x.

Quick Sort - Analysis

- The runtime depends on the result of partition.

Quick Sort - Analysis

- The runtime depends on the result of partition.
  - If the resulting subarrays are balanced, then $T(n) = 2T(\frac{n}{2}) + \Theta(n)$.
    This is $\Theta(nlog(n))$ by Case 2 of the master theorem.
  - If the resulting subarrays are not balanced, then $T(n) = T(n-1) + \Theta(n)$. This is in $\Theta(n^2)$.

Quick Sort - Analysis

- The runtime depends on the result of partition.
  - If the resulting subarrays are balanced, then $T(n) = 2T(\frac{n}{2}) + \Theta(n)$.
    This is $\Theta(nlog(n))$ by Case 2 of the master theorem.
  - If the resulting subarrays are not balanced, then $T(n) = T(n-1) + \Theta(n)$. This is in $\Theta(n^2)$.
- The best case is as fast as merge sort. The worst case is as bad as insertion sort, but happens when the array is already sorted. The average case is as good as the best case.

Outline

Points to Take Home

1. Quick Sort.
2. Reading Material:
   - Chapter 7: Sections 7.1 and 7.2.

Next Lecture: Quick Sort!

Due Credits

The presented material is based on:

1. Previous editions of the course at the GUC due to Dr. Wael Aboulsaadat, Dr. Haythem Ismail, Dr. Amr Desouky, and Dr. Carmen Gervet.
2. Stony Brook University's Analysis of Algorithms Course.
3. MIT's Introduction to Algorithms Course.
4. Stanford's Design and Analysis of Algorithms Course.