CSE 595: Advanced Topics in Computer Science
Presentation 8

Zeeshan Shaikh

Department of Computer Science, Stonybrook University

07/22/2021

Topics for today's presentation

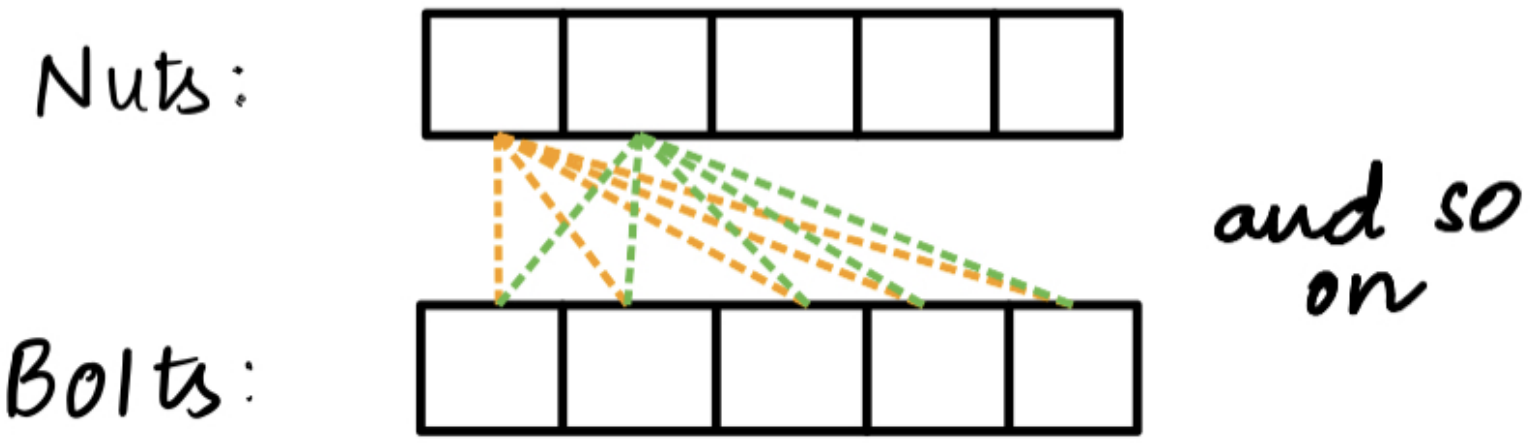
- Nuts and Bolts Problem

Step 1: Problem

- Given: Two arrays: One array holds the nuts and the other array holds the bolts. The nuts and the bolts cannot be compared to each other i.e you can compare the nuts with the bolts and vice-versa only. We need to find the 1 to 1 mapping of bolts and nuts.
- Example: For representing different types of bolts and nuts, we use special characters.
  nuts = [" ∗ ","@","!"]
  bolts = ["!","@"," ∗ "]
  output:
  nuts = ["@"," ∗ ","!"] bolts = ["@"," ∗ ","!"]

Brute Force Method

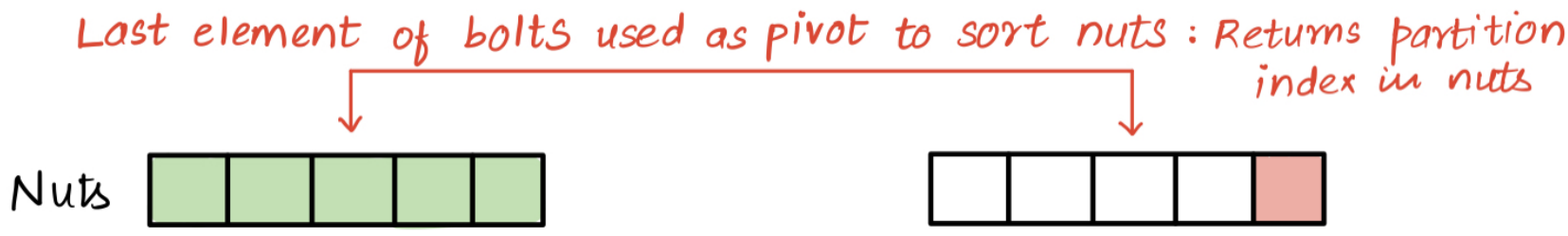
In the Brute Force method, we check every nut against the bolt to see if it is a match.



Time Complexity: $O(n^2)$

Decrease and Conquer Approach

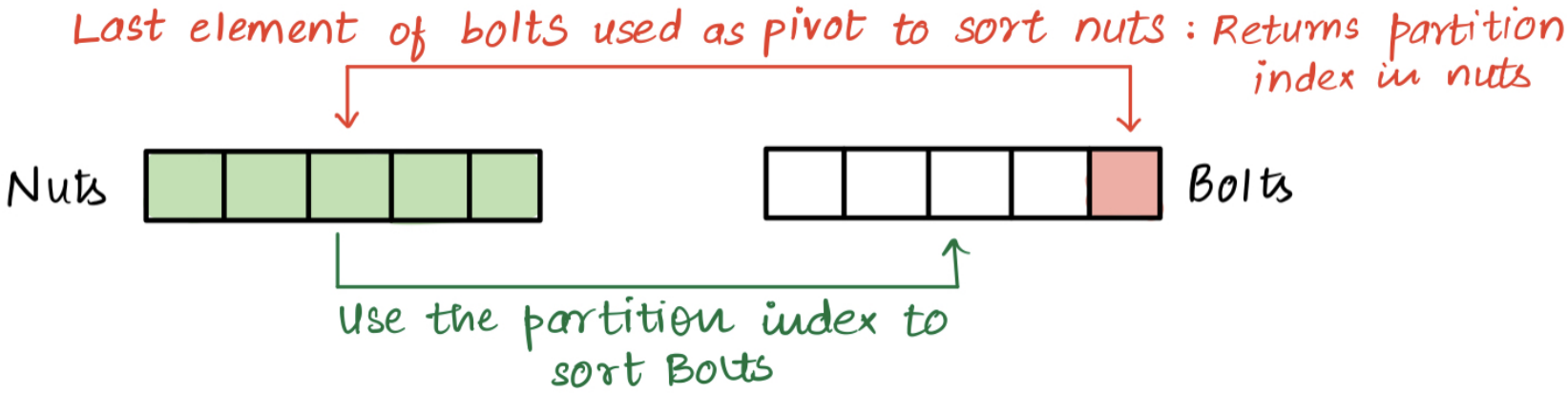
In the Decrease and Conquer, the approach is quite similar to quicksort, with some minor modifications.



Since we cannot compare the nuts to each other or the bolts to each other, we use the last element of the Bolt as a pivot to sort the array holding nuts.

Decrease and Conquer Continued

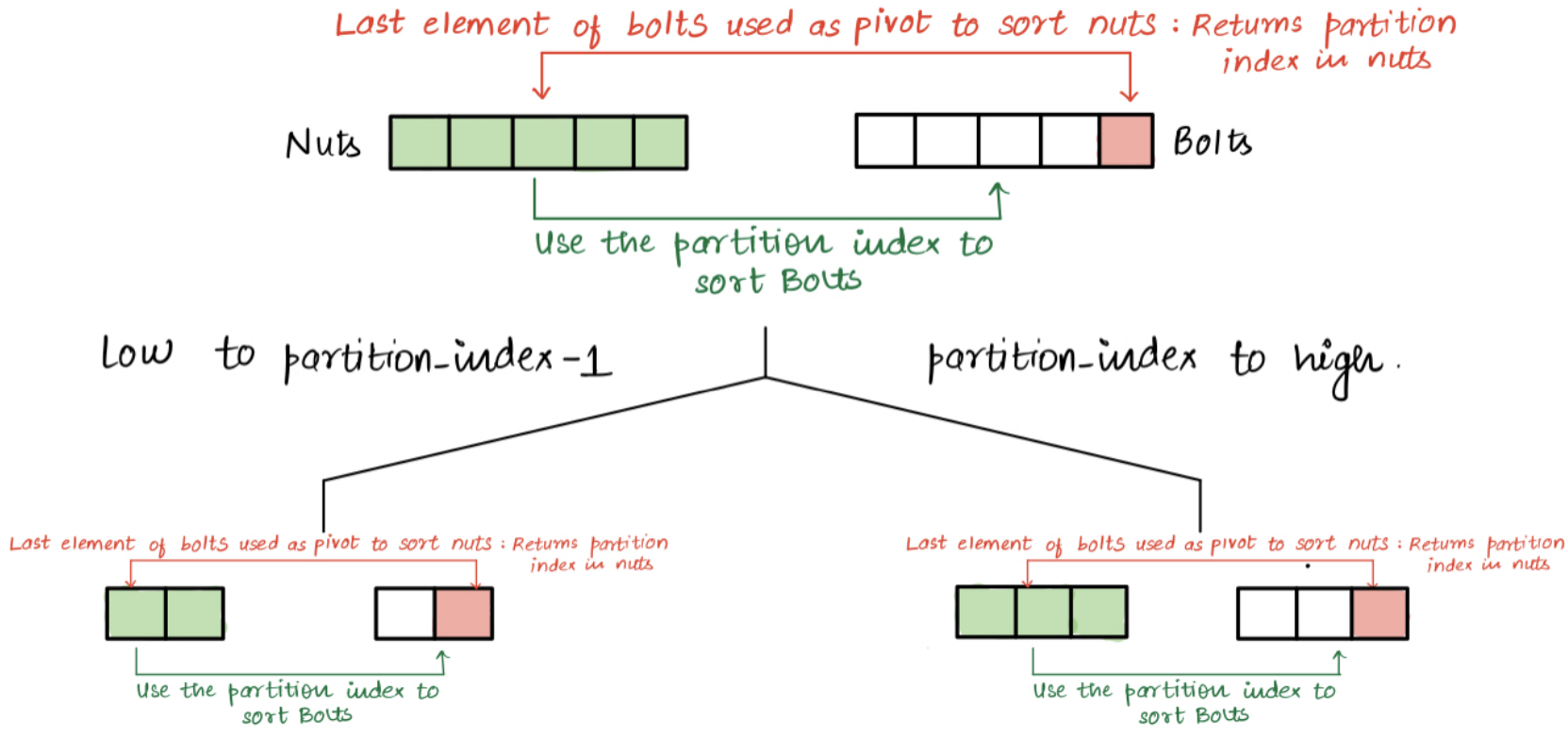
Once the previous step is complete, we will have a "partition index" from the Nuts array.



The partition index so calculated is used to sort the bolts array.

Decrease and Conquer Continued

However, the array is not completely sorted yet.



After the same, we recursively call the function by diving both the arrays over the partition index so acheived till will get to single cells.

Instructions

Given: two arrays: nuts and bolts. Two iterators low and high initialized to 0 and n repsectively.

- Recursive function: sortArrays()

Algorithm 1

```
1: if low < high then
2:     pivot = partition(nuts, low, high, bolts[high])
3:     partition(bolts, low, high, nuts[pivot])
4:     sortArrays(nuts, bolts, low, pivot-1)
5:     sortArrays(nuts, bolts, pivot, high)
6: end if
```

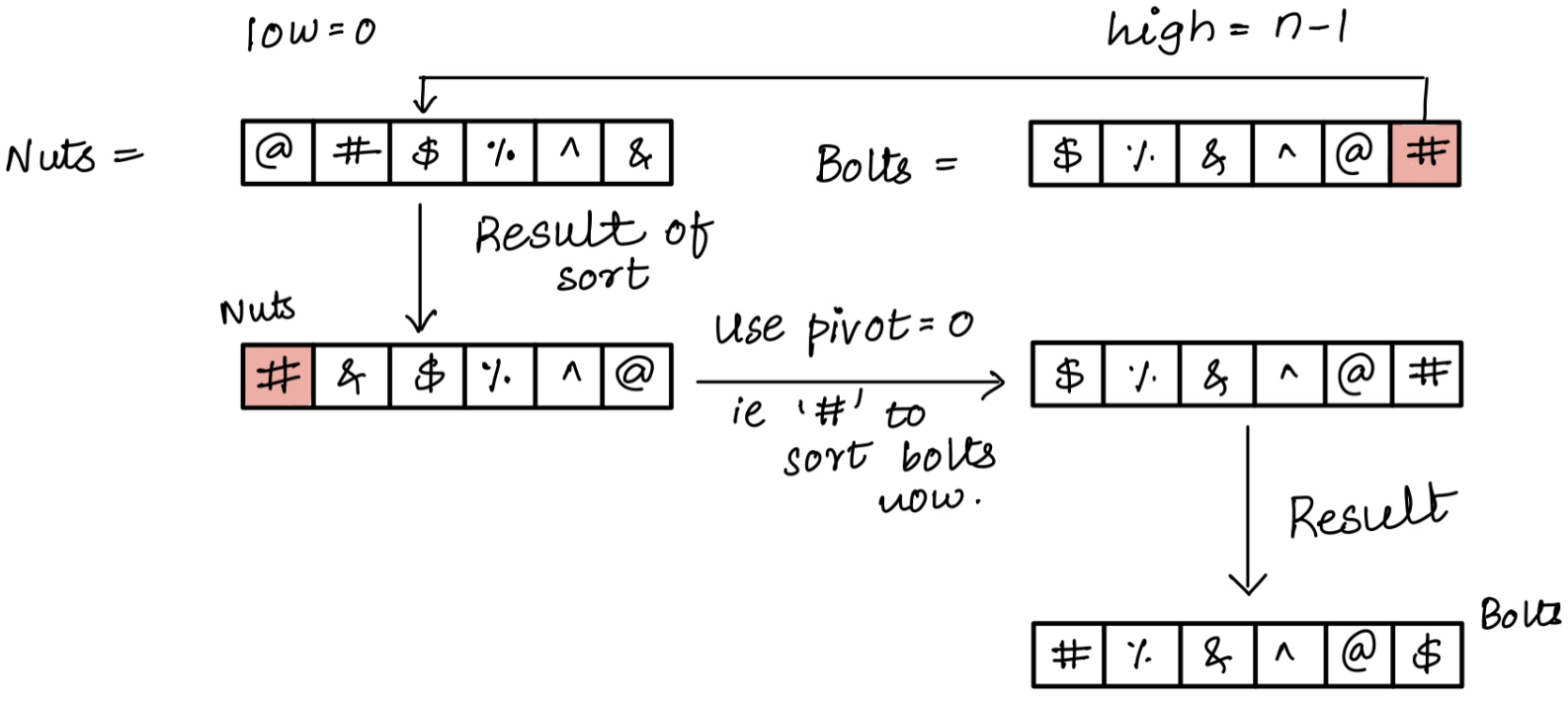
Instructions(Continued)

- Entry in the table for every other case

Algorithm 2 Partition function

```
1:  Initialize i = low
2:  for j = low to high do
3:      if arr[j] < pivot then
4:          swap arr[j] and arr[i]
5:          i++
6:      end if
7:      if arr[j] = pivot then
8:          swap arr[j] and arr[high]
9:          j − −
10:     end if
11: end for
```

Example


low = 0
high = n-1
Nuts =
@ # $ % ^ &
Bolts =
$ % & ^ @ #
Result of sort
Nuts
& $ % ^ @
use pivot = 0
ie '#' to sort bolts now.
$ % & ^ @ #
Result
Bolts
% & ^ @ $

Example

Now, this is what they look like

Nuts = | # | & | $ | % | ^ | @ |

Bolts = | # | % | & | ^ | @ | $ |

Recursive call over these array such that:

Call 1: low = low, high = pivot - 1

Call 2: low = pivot, high = high

Repeat till low < high is satisfied.

Complexity Analysis

Complexity Analysis		
Algorithm	Time	Space
Nuts and Bolt Problem	$O(N^2)$	$O(logn)$

Average case time complexity for this algorithm is O(nlogn) but the worst case is the scenario when the pivot for splitting is at the end of the array, bringing it to $O(n^2)$