Non-Recursive QuickSelect Algorithm

One step closer to understanding the non-recursive QuickSelect algorithm.

Table of Content

What is QuickSelect?

QuickSelect is a selection algorithm to find the kth smallest element in an unordered list.

Spot the difference

QuickSort

it recurs for both sides (after finding pivot)

QuickSelect

It recurs only for the part that contains the k-th smallest element.

Example

Apply non-recursive QuickSelect algorithm to find the 5th smallest element of the following list of nine numbers: 4, 1, 10, 8, 7, 12, 9, 2, 15. Here, k =5.

Strategy

SOLVE IT AGAIN AFTER WE FINISH

The logic is simple, we will use decrease and conqure by variable. So, if index of the partitioned element is more than k, then we will loop on the left part. If index is the same as k, we have found the k-th smallest element and we will return the element. If index is less than k, then we will loop on the right part.

What is decrease and conquer?

- constant
- constant factor
- variable

The target of decrease and conquer strategy is to decrease the input size in every iteration. So, here in the Variable decrease and conquer the size reduction pattern
varies from one iteration of the algorithm to another.

Input size is n

Input:

An array A[0..n-1] of orderable elements and 1<=k<=n

Output:

The value of the k-th smallest element in A[0..n-1]

using Lomuto partition-based algorithm

algorithm Design

first, we have to create a QuickSelect function that takes A[0..n-1] and k

```
function quickselect(A[0..n-1], k):
```

assign leftmost and rightmost elements of the current subarray.

```
function quickselect(A[0..n-1], k):
        left<-0
        right<-n-1
```

loop until left exceeds right & assign i and j as pointers for the partitioning

```
function quickselect(A[0..n-1], k):
        left<-0,  right<-n-1
        while left <= right:
           p<- A[left]  // pivot
           i<-left,  j<-r
```

do nested loop until left exceeds right

```
function quickselect(A[0..n-1], k):
        left<-0,  right<-n-1
        while left <= right:
            p<- A[left]  // pivot
            i<-left,  j<-r+1
            while i < j:
                while A[i] < p:  i+=1
                while A[j] > p: j-=1
                swap(A[i], A[j])
```

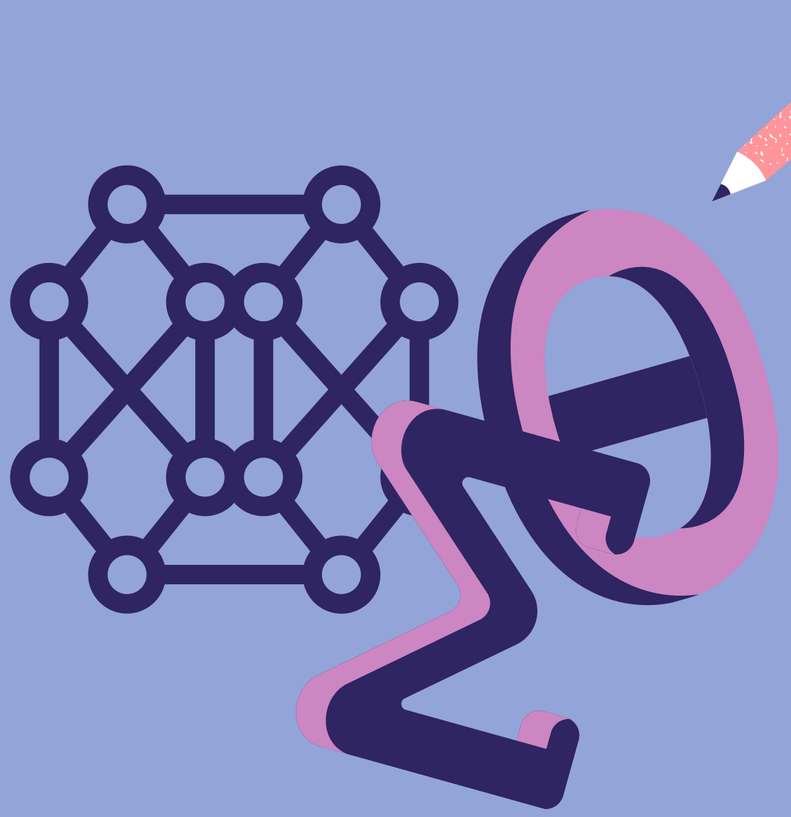
basic operation is the comparison

The partition loop finds elements on the left (i) greater than the pivot and elements on the right (j) smaller than the pivot, then swaps them.

swap for partition then check the conditions

```
function quickselect(A[0..n-1], k):
        left<-0,  right<-n-1
        while left <= right:
            p<- A[left]  // pivot
            i<-left,  j<-r+1
            while i < j:
                while A[i] < p:  i+=1
                while A[j] > p: j-=1
                swap(A[i], A[j])
            swap(A[l], A[j])
            if j > k-1: r <- j-1 // left part
            else if j < k-1: l<- j+1 // right part
            else return A[j] // the k-th element
```

The comparison is to k-1 because the array is zero based.
The pivot is swapped with the element at index j, placing the pivot in its final sorted position.



The average case

$$T_{avg}(n) = O(n) + O(\frac{n}{2}) + O(\frac{n}{4}) + \ldots + O(1)$$

$$T_{avg}(n) = \sum_{i=0}^{log_2 n} O(\frac{n}{2^i})$$

The worst case

$$T_{worst}(n) = O(n) + O(n-1) + O(n-2) + \ldots + O(1)$$

$$T_{worst}(n) = \sum_{i=0}^{n-1} O(n-i)$$

What is the analysis?

File Edit Selection View Go Run Terminal Help

Search

EXPLORER

OPEN EDITORS

algo.py C:\Users\Windows 10...

NO FOLDER OPENED

You have not yet opened a folder.

Open Folder

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

Clone Repository

To learn more about how to use Git and source control in VS Code read our docs.

OUTLINE

TIMELINE

algo.py

C: > Users > Windows 10 Pro > Desktop > algo.py > quickselect

Click here to ask Blackbox to help you code faster

```python
def quickselect(arr, k):
    n = len(arr)
    left, right = 0, n - 1
    arr.append(float('inf'))  # Sentinel value for simplicity
    while left <= right:
        p = arr[left]  # Pivot
        i, j = left, right + 1

        while i < j:
            while arr[i] < p:
                i += 1
            while arr[j] > p:
                j -= 1
            arr[i], arr[j] = arr[j], arr[i]
        # Swap pivot into its final position
        arr[left], arr[j] = arr[j], arr[left]
        if j > k - 1:
            right = j - 1
        elif j < k - 1:
            left = j + 1
        else:
            return arr[left]  # Return the k-th smallest element
# Example usage:
arr = [4, 1, 10, 8, 7, 12, 9, 2, 15]
k = 5
result = quickselect(arr, k)
print(f"The {k}-th smallest element is: {result}")

```

PROBLEMS DEBUG CONSOLE TERMINAL OUTPUT PORTS SEARCH ERROR

```
The 5-th smallest element is: 8
PS C:\Users\Windows 10 Pro\Desktop>
```

powersh...

Python

Python Deb...

Activate Windows
Go to Settings to activate Windows.

0 0 0 Share Code Link Explain Code Comment Code Code Chat Blackbox Search Error

Ln 4, Col 62 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Blackbox

Type here to search

23°C غائم غالبًا ENG 5:11 PM 12/30/2023

we finished

I hope you understand this topic well

for resources

GitHub Link :
https://github.com/maiiiiiii234/QuickSelect_Resources.git