Kth Smallest Element

Aim- Write a C program to find a given kth smallest element in an array of characters

Problem Statement – Given a array of characters implement select algorithm to calculate given kth smallest element in the array

INPUT - The number of elements in the array = 11

- Array Elements – E,G,I,L,N,Q,V,F,Q,S,D
- 1] k = 4
- 2] k= 9

OUTPUT – Display each step of the process and finally display the kth smallest element

ALGORITHM –

i] Algorithm Select (a,n,k)

// Selects the kth smallest element in a[1:n] and places it in the kth position of a[1:n] .

//Remaining elements are rearranged such that a[m]<= a[k] ,for i<=m<=k and a[m]>=a[k]

//for k<m<=n

{

low:= 1 ; up :=n+1 ;

a[n+1] ;=∞ ;

repeat {

j := partition (a,low,up) ;

else if (k<j) then { up := j ; }

else { low := j+1 ;}

until(false) ; }

ii] Algorithm Partition(a,m,q)

```
//Within a[m],a[m+1]…… a[p-1] the elements are rearranged in such a manner that if
// initially  t= a[m] ,then after completion a[q] = t for some q between m and p-1 ;
// a[k] <=t for some m<=k<=q , and a[k] >= t for some q<k<p , q is returned
{   v := a[m] ;
 i := m ;  j : = p ;
repeat {
repeat { i := i+1 ;     }
until (a[i] >=v) ;
repeat { j := j+1 ;
until (a[j] <=v) ;
if (i < j    ) {
temp := a[i] ;
a[i] = a[j] ;
a[j] = temp ; }
until (i>=j)  ;
a[m] = a[j] ;
a[j] : = v ;
return j ; }
```

Space and Time Complexity :

I] Algorithm Select

Time Complexity

1. **Best Case**:
    - In the best case, the pivot divides the array into two equal parts each time.
    - Time Complexity: O(n)
2. **Worst Case**:
    - In the worst case, the pivot is always the smallest or largest element, leading to one partition with n−1n-1n−1 elements.
    - Time Complexity: $O(n^2)$

3. **Average Case**:
    - On average, the pivot divides the array into two unequal parts (e.g., n4\frac{n}{4}4n and 3n4\frac{3n}{4}43n).
    - Time Complexity: O (n logn)

Space Complexity

Algorithm: Select

1. **Best Case**:
    - Space is required for recursive calls. The depth of the recursion tree is O(log n) for the best case.
    - Space Complexity : O(log n)
2. **Worst Case**:
    - The depth of the recursion tree is O(n) in the worst case (when pivot produces highly unbalanced partitions).
    - Space Complexity: O(n)
3. **Average Case**:
    - On average, the depth of recursion is O(log n).
    - Space Complexity: O(log n)

II] Algorithm Partition

Time Complexity:

i) **Best Case:**

- **O(n)**
- All elements are scanned and rearranged once relative to the pivot.

ii) **Worst Case:**

- **O(n)**
- Every element is still scanned and rearranged in a single pass regardless of their order.

iii) **Average Case:**

- **O(n)**
- The partitioning process always involves a single scan of all elements in the subarray.

Space Complexity:

i) **Best Case:**

- **O(1)**
- Partitioning requires constant auxiliary space for temporary variables.

ii) **Worst Case:**

- **O(1)**
- No additional space is required apart from the input array and temporary variables.

iii) **Average Case:**

- **O(1)**
- Space usage remains constant.

RECURSION EQUATION –

I] Select

$$T(n) = T(k) + O(n)$$

II] Partition

The Partition algorithm does not have a recurrence relation because it is not recursive. It is a single-pass operation that rearranges elements relative to a pivot. Its complexity is handled entirely within its single invocation, and no further subproblems are generated. Therefore, no recurrence equation exists for Partition.

PROGRAM –

```
#include <stdio.h>
#include <time.h>
#define MAX 20
char arr[MAX];

void display_array(int n) {
  if (n == 0) {
    printf("Array is empty. Please enter the array first.\n");
    return;  }
  printf("Array elements are: ");
  for (int i = 0; i < n; i++) {
    printf("|%c ", arr[i]);
  }
  printf("|\n");
}
int partition(int low, int high) {
    clock_t start, end;
    double cpu_time_used;
    start = clock();
  int i = low + 1, j = high;
  char pivot = arr[low];
  while (i <= j) {
    while (i <= high && arr[i] <= pivot) i++;
    while (arr[j] > pivot) j--;
    if (i < j) {
      char temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
    }
  }
  arr[low] = arr[j];
  arr[j] = pivot;
  return j;
}

void select(int k, int n) {
  if (k <= 0 || k > n) {
    printf("Invalid value of k. Please enter a value between
1 and %d.\n", n);
    return;
  }
  int l = 0, h = n - 1;
  int j;
  while (l <= h) {
    j = partition(l, h);
    printf ("\n ");
    display_array(n);
    printf (" j = %d and pivot = %c\n", j, arr[j]);
    printf ("\n ");
    if (j == k - 1) {
      printf ("\n ");
      display_array(n);
      printf (" j = %d and pivot = %c\n", j, arr[j]);
      printf ("\n \n");
      printf("%dth smallest element is '%c'\n", k, arr[j]);
      return;
    } else if (j < k - 1) {
      printf ("\n ");
      display_array(n);
      printf (" j = %d and pivot = %c\n", j, arr[j]);
      printf ("\n ");
      l = j + 1;
    } else {
       printf ("\n ");
    display_array(n);
       printf (" j = %d and pivot = %c\n", j, arr[j]);
```

```
    printf ("\n ");
    h = j - 1;
   }
  }

}

int main() {
 printf
("**********************************************
*********");
   printf ("\n Roll number: 23B-CO-010\n");
   printf (" PR Number - 202311390\n");
   printf("**************************************
***************\n\n\n");
  int choice;
  int n = 0, k;
   clock_t start, end;
   double cpu_time_used;
  do {
   printf("\n\n---------- Menu --------\n");
   printf("1. Enter the elements of the array\n");
   printf("2. Find kth smallest element in the list\n");
   printf("3. Display the array\n");
   printf("4. Exit\n");
   printf("Choose your option: ");
   scanf("%d", &choice);
   switch (choice) {
    case 1:
      printf("Enter the number of elements in the array
(max %d): ", MAX);
      scanf("%d", &n);
      if (n <= 0 || n > MAX) {
       printf("Invalid number of elements. Please enter a
value between 1 and %d.\n", MAX);
        n = 0;
        break;
      }
      printf("Enter the elements of the array (as
characters):\n");
      for (int i = 0; i < n; i++) {
        while ((arr[i] = getchar()) == '\n');
      }
      break;
    case 2:   start = clock();
      if (n == 0) {
       printf("Array is empty. Please enter the array
first.\n");
        break;
      }
      printf("Enter the value of k: ");
      scanf("%d", &k);
      select(k, n);
       end = clock();
  cpu_time_used = ((double) (end - start)) /
CLOCKS_PER_SEC;
  printf("\nTime taken by partition function : %f
seconds\n", cpu_time_used);
      break;
    case 3:
      display_array(n);
      break;
    case 4:
      printf("Exiting the program.\n");
      break;
    default:
      printf("Invalid choice. Please choose a valid
option.\n");
      break;
   }
  } while (choice != 4);

  return 0;
}
```

INPUT -

```
**************************************************
 Roll number: 23B-CO-010
 PR Number - 202311390
**************************************************



--------------- Menu --------
1. Enter the elements of the array
2. Find kth smallest element in the list
3. Display the array
4. Exit
Choose your option: 1
Enter the number of elements in the array (max 20): 11
Enter the elements of the array (as characters):
E
G
I
L
N
O
V
F
Q
S
D


--------------- Menu --------
1. Enter the elements of the array
2. Find kth smallest element in the list
3. Display the array
4. Exit
Choose your option: 3
Array elements are: |E |G |I |L |N |O |V |F |Q |S |D |


--------------- Menu --------
1. Enter the elements of the array
2. Find kth smallest element in the list
3. Display the array
4. Exit
Choose your option: 2
```

OUTPUT –

I] K = 4

```
Enter the value of k: 4

 Array elements are: |D |E |I |L |N |O |V |F |Q |S |G |
 j = 1 and pivot = E


 Array elements are: |D |E |I |L |N |O |V |F |Q |S |G |
 j = 1 and pivot = E


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 4 and pivot = I


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 4 and pivot = I


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 2 and pivot = F


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 2 and pivot = F


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 3 and pivot = G


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 3 and pivot = G


4th smallest element is 'G'
```

II] K = 9

```
Enter the value of k: 9

 Array elements are: |D |E |I |L |N |O |V |F |Q |S |G |
 j = 1 and pivot = E


 Array elements are: |D |E |I |L |N |O |V |F |Q |S |G |
 j = 1 and pivot = E


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 4 and pivot = I


 Array elements are: |D |E |F |G |I |O |V |N |Q |S |L |
 j = 4 and pivot = I


 Array elements are: |D |E |F |G |I |N |L |O |Q |S |V |
 j = 7 and pivot = O


 Array elements are: |D |E |F |G |I |N |L |O |Q |S |V |
 j = 7 and pivot = O


 Array elements are: |D |E |F |G |I |N |L |O |Q |S |V |
 j = 8 and pivot = Q


 Array elements are: |D |E |F |G |I |N |L |O |Q |S |V |
 j = 8 and pivot = Q


9th smallest element is 'Q'
```

TIME TAKEN –

I] K = 4

```
Time taken by select function : 3.170000 seconds
```

II] K = 9

```
Time taken by select function : 3.327000 seconds
```

CONCLUSION – The kth smallest element was calculation sucessfully using select algorithm without any errors .