

DIVIDE AND CONQUER

Theory –

General Method

Divide-and-conquer is a problem-solving strategy that breaks a large problem into smaller, more manageable subproblems. The process involves:

1. **Splitting the Problem:** Divide the input into distinct subsets, where , resulting in subproblems.
2. **Solving Subproblems:** Solve each subproblem independently, often recursively.
3. **Combining Solutions:** Combine the solutions of the subproblems to form the solution to the original problem.

If the subproblems are still large, the process of division can be reapplied until the subproblems become small enough to solve directly. Typically, the subproblems are of the same type as the original problem. This approach is naturally expressed as a recursive algorithm.

Control Abstraction

To better understand the strategy, consider the general form of a divide-and-conquer algorithm:

1. If the input size is small enough, solve the problem directly.
2. Otherwise, divide the problem into smaller instances .
3. Recursively solve these smaller problems.
4. Combine the solutions of the smaller problems to obtain the solution to .

Algorithm : DandC

Algorithm DAndC(P):

if Small(P) then

return S(P);

else

divide P into smaller instances P_1, P_2, \dots, P_k ;

Apply DAndC to each of these subproblems;

return Combine(DAndC(P_1), DAndC(P_2), ..., DAndC(P_k));

- **Small(P):** A Boolean function that determines whether the input size is small enough to solve directly.
- **S(P):** The solution for small problems.
- **Combine:** A function that merges the solutions of subproblems.

Time Complexity

The time complexity of a divide-and-conquer algorithm is generally expressed by a recurrence relation. For instance:

- : Total time for an input of size .
- : Time for directly solving small inputs.
- : Time for dividing and combining solutions.
- : Time for solving each subproblem.

For many divide-and-conquer algorithms, the recurrence takes the form:

Here:

- : Number of subproblems.
- : Factor by which the input size is divided.
- : Time for dividing and combining.

Solving Recurrences

One common method for solving such recurrence relations is the **substitution method**, which involves:

1. Repeatedly substituting the recurrence formula into itself.
2. Simplifying until all instances of are eliminated.
3. Summing up the resulting terms to obtain a closed-form solution.

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 : 0.025000 seconds

II] K = 9

 Time taken by select function : 0.045000 seconds

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

INPUT –

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

Menu:
1. Enter elements of the array
2. Display elements of the array
3. Search for an element
4. Exit
Enter your choice: 1
Enter the number of elements in the array: 9
Enter element 1: DOG
Enter element 2: CAT
Enter element 3: PIG
Enter element 4: ANT
Enter element 5: ELEPHANT
Enter element 6: ZEBRA
Enter element 7: LION
Enter element 8: TIGER
Enter element 9: OX

Menu:
1. Enter elements of the array
2. Display elements of the array
3. Search for an element
4. Exit
Enter your choice: 2
The elements of the array are: |      ANT |      CAT |      DOG |  ELEPHANT |      LION |      OX |      PIG |      TIGER |      ZEBRA |

Menu:
1. Enter elements of the array
2. Display elements of the array
3. Search for an element
4. Exit
Enter your choice: 3
Enter the element to be searched: OX
```

OUTPUT –

```
Element found at index 5
```

TIME TAKEN –

```
Time taken by Binary Search: 0.000000 seconds
```

CONCLUSION - Binary search on array of strings was successfully executed without errors