C S Deeraj

CH.SC.U4CSE24106

Week 5

Date – 22/01/2026

Design and Analysis of Algorithms (23CSE211)

Quick Sort – Selection of Pivots

quicksort - Notepad

File Edit Format View Help

```
//CH.SC.U4CSE24106
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int *a,int *b){
        int t=*a;
        *a=*b;
        *b=t;
}

int partition_last(int a[],int low,int high){
        int pivot=a[high];
        int i=low-1;

        for(int j=low;j<high;j++){
                if(a[j]<pivot){
                        i++;
                        swap(&a[i],&a[j]);
                }
        }
        swap(&a[i+1],&a[high]);
        return i+1;
}

int partition_first(int a[],int low,int high){
        swap(&a[low],&a[high]);
        return partition_last(a,low,high);
}

int partition_random(int a[],int low,int high){
        int r=low+rand()%(high-low+1);
        swap(&a[r],&a[high]);
        return partition_last(a,low,high);
}

void quicksort(int a[],int low,int high,int ch){
        if(low<high){
                int p;
                if(ch==1)
                        p=partition_first(a,low,high);
                else if(ch==2)
```

quicksort - Notepad

File Edit Format View Help

```
                if(ch==1)
                        p=partition_first(a,low,high);
                else if(ch==2)
                        p=partition_last(a,low,high);
                else
                        p=partition_random(a,low,high);

                quicksort(a,low,p-1,ch);
                quicksort(a,p+1,high,ch);
        }
}

int main(){
        int a[]={157,110,147,122,111,149,151,141,123,112,117,133};
        int n=12;
        int ch;

        srand(time(NULL));

        printf("Original list:\n");
        for(int i=0;i<n;i++)
                printf("%d ",a[i]);

        printf("\n\nChoose pivot type:\n");
        printf("1. First element as pivot\n");
        printf("2. Last element as pivot\n");
        printf("3. Random element as pivot\n");
        printf("Enter your choice: ");
        scanf("%d",&ch);

        quicksort(a,0,n-1,ch);

        printf("\nSorted list:\n");
        for(int i=0;i<n;i++)
                printf("%d ",a[i]);

        printf("\nRoll number: CH.SC.U4CSE24106");

        return 0;
}
```

Output:

```
C:\Users\savit\OneDrive\Desktop\Sem IV CSE\Ws1 haskell>gcc quicksort.c -o quicksort

C:\Users\savit\OneDrive\Desktop\Sem IV CSE\Ws1 haskell>./quicksort
'.' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\savit\OneDrive\Desktop\Sem IV CSE\Ws1 haskell>quicksort
Original list:
157 110 147 122 111 149 151 141 123 112 117 133

Choose pivot type:
1. First element as pivot
2. Last element as pivot
3. Random element as pivot
Enter your choice: 1

Sorted list:
110 111 112 117 122 123 133 141 147 149 151 157
Roll number: CH.SC.U4CSE24106
```

Time Complexity: O(nlogn) – Average, O(n^2) – Worst Case

The time complexity depends on how we choose the pivot. In the best and average cases, when the pivot roughly splits the array in half each time, the algorithm runs in O(n log n) time. This happens because we divide the array log n times and do about n work at each division level. In the worst case, when the pivot is always the smallest or largest element (like using first or last pivot on an already sorted array), it degrades to $O(n^2)$ time because we only place one element per recursion. The random pivot option helps us avoid this worst case most of the time.

Space complexity: The space complexity is determined by the recursion stack depth. In the best case with balanced partitions, the stack depth is O(log n). However, in the worst case with unbalanced partitions, the recursion can go n levels deep, using O(n) stack space. The algorithm itself uses only O(1) extra memory for swapping elements and storing indices, making it an in-place sorting method.