



AMRITA
VISHWA VIDYAPEETHAM

DESIGN AND ANALYSIS OF
ALGORITHMS
LAB WORKBOOK
WEEK – 6

NAME : ROHITH S

ROLL NUMBER : CH.SC.U4CSE24140

CLASS : CSE-B

Question 1: Write a program to perform Quick Sort by taking First Element, Last Element and a Random Element as Pivot Element for the given numbers:

157, 110, 147, 122, 149, 151, 111, 141, 112, 123, 133, 117

CODE:

```
C quicksort.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  void swap(int *a,int *b)
6  {
7      int t=*a;
8      *a=*b;
9      *b=t;
10 }
11
12 void printArray(int a[],int n)
13 {
14     for(int i=0;i<n;i++)
15         printf("%d ",a[i]);
16     printf("\n");
17 }
18
19 int partitionFirst(int a[],int low,int high)
20 {
21     int pivot=a[low];
22     int i=low+1;
23     int j=high;
24     while(i<=j)
25     {
26         while(i<=high&& a[i]<=pivot)
27             i++;
28         while(a[j]>pivot)
29             j--;
30         if(i<j)
31             swap(&a[i],&a[j]);
32     }
33     swap(&a[low],&a[j]);
34     return j;
35 }
```

```
37 void quickSortFirst(int a[],int low,int high)
38 {
39     if(low<high)
40     {
41         int p=partitionFirst(a,low,high);
42         quickSortFirst(a,low,p-1);
43         quickSortFirst(a,p+1,high);
44     }
45 }
46
47 int partitionLast(int a[],int low,int high)
48 {
49     int pivot=a[high];
50     int i=low-1;
51     for(int j=low;j<high;j++)
52     {
53         if(a[j]<=pivot)
54         {
55             i++;
56             swap(&a[i],&a[j]);
57         }
58     }
59     swap(&a[i+1],&a[high]);
60     return i+1;
61 }
62
63 void quickSortLast(int a[],int low,int high)
64 {
65     if(low<high)
66     {
67         int p=partitionLast(a,low,high);
68         quickSortLast(a,low,p-1);
69         quickSortLast(a,p+1,high);
70     }
71 }
```

```

73  int partitionRandom(int a[],int low,int high)
74  {
75      int r=low+rand()%(high-low+1);
76      swap(&a[r],&a[high]);
77      return partitionLast(a,low,high);
78  }
79
80  void quickSortRandom(int a[],int low,int high)
81  {
82      if(low<high)
83      {
84          int p=partitionRandom(a,low,high);
85          quickSortRandom(a,low,p-1);
86          quickSortRandom(a,p+1,high);
87      }
88  }
89

```

```

90  int main()
91  {
92      int a[]={157,110,147,122,149,151,111,141,112,123,133,117};
93      int n=sizeof(a)/sizeof(a[0]);
94      srand(time(0));
95      int a1[n],a2[n],a3[n];
96      for(int i=0;i<n;i++)
97      {
98          a1[i]=a[i];
99          a2[i]=a[i];
100         a3[i]=a[i];
101     }
102     quickSortFirst(a1,0,n-1);
103     quickSortLast(a2,0,n-1);
104     quickSortRandom(a3,0,n-1);
105     printArray(a1,n);
106     printArray(a2,n);
107     printArray(a3,n);
108     return 0;
109 }

```

OUTPUT:

110 111 112 117 122 123 133 141 147 149 151 157

110 111 112 117 122 123 133 141 147 149 151 157

110 111 112 117 122 123 133 141 147 149 151 157

PS C:\Users\123sr\OneDrive\Desktop\COLLEGE STUFF\SEM IV\DAA LAB\Week 6> □

WORKING:

Quick Sort
Working:-

1) First element as Pivot

Pass 1

1) 157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133

Pivot L → R

$A[R] = 157 < 157$

$R = 11$

$L \geq R$

∴ Swap

Pass 2)

133, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117

Pivot = 33

$L = 1 \rightarrow 110$

$R = 10 \rightarrow 117$

$10 \leq 33 \rightarrow L++$

$A[L] = 147 > 33$

$L = 2$

Move R

$117 < 33$

Swap

133, 110, 117, 122, 111, 149, 151, 141, 123, 112, 147

Swap (149, 112)

133, 110, 117, 122, 111, 112, 151, 141, 123, 149, 147

Swap (151, 123)

133, 110, 117, 122, 111, 112, 123, 141, 151, 149, 147

Swap with pivot

123, 110, 117, 122, 111, 112, 133, 141, 151, 149, 147

Pass 3: Pivot = 123

Left = [123, 110, 117, 122, 111, 125]

Right = [141, 151, 149, 147]

Array:

[123, 110, 117, 122, 111, 125]

7 1 2 3 4 5 6

Swap (02, 123)

→ [12, 110, 117, 122, 111, 125]

Pass 4: Pivot = 112

[112, 110, 117, 122, 111, 125]

1 2 3 4 5 6

Swap (117, 111)

→ [112, 110, 111, 122, 117, 125]

1 2 3 4 5 6

Swap (111, 122) (Pivot Swap)

[111, 110, 112, 122, 117, 125]

LP → [111, 110] → Sorted → [110, 111]

RP → [122, 117] → Swap → [117, 122]

Left side:-

→ [110, 111, 112, 117, 122, 125]

Right Side Sorting:-

[141, 151, 149, 147]

Pivot = 141

[141, 151, 149, 147]

1 2 3 4

Swap (Pivot with L)

[151, 141, 149, 147]

Pivot = [151]

[151, 141, 149, 147]

1 2 3 4

Swap (P, R)

[151]

→ [147, 141, 149, 151]

Pivot = 147

[147, 141, 149, 151]

Swap (P, L)

→ [141, 147, 149, 151]

Merge two array

[110, 111, 112, 117, 122, 125, 141, 147, 149, 151]

ii) Last element as Pivot

Phase 1
 $[157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 155]$
 Pivot = 155

Swap (L, R)

$[117, 110, 147, 122, 111, 149, 151, 141, 123, 155, 157]$
 L₁ L₂ L₃ L₄ L₅ L₆ L₇ L₈ L₉ L₁₀

Swap (L, R)

$[117, 110, 112, 122, 111, 149, 151, 141, 123, 157, 155]$
 L₁ L₂ L₃ L₄ L₅ L₆ L₇ L₈ L₉ L₁₀

Swap (L, R)

$[117, 110, 112, 122, 111, 123, 151, 141, 149, 157, 155]$
 L₁ L₂ L₃ L₄ L₅ L₆ L₇ L₈ L₉ L₁₀

Swap (Pivot, L)

$[117, 110, 112, 122, 111, 123, 155, 141, 149, 157, 151]$
 L₁ L₂ L₃ L₄ L₅ L₆ L₇ L₈ L₉ L₁₀

Left Array :-

$[117, 110, 112, 122, 111, 123]$

Right Array :-

$[141, 149, 157, 151]$

Phase 2 (Left Array):

$[117, 110, 112, 122, 111, 123]$
 L₁ L₂ L₃ L₄ L₅ L₆

Swap (111, 123)

$[117, 110, 112, 122, 123, 111]$
 L₁ L₂ L₃ L₄ L₅ L₆

Phase 3:

$[117, 110, 112, 122, 123, 111]$
 L₁ L₂ L₃ L₄ L₅ L₆

Swap (L, R)

$[110, 117, 112, 122, 123, 111]$
 L₁ L₂ L₃ L₄ L₅ L₆

Swap (Pivot, R)

$[110, 111, 112, 122, 123, 117]$

Sorted:

Right Array :-

$[141, 149, 157, 151]$
 L₁ L₂ L₃ L₄

Swap (P, Pivot)

$[141, 149, 157, 151]$
 L₁ L₂ L₃ L₄

Swap (L, R)

$[141, 149, 147, 151, 157]$
 L₁ L₂ L₃ L₄ L₅

Swap (Pivot, L)

$[141, 147, 149] \rightarrow [151, 157]$

L₁ + R₁

$[110, 111, 112, 122, 123, 117, 141, 149, 151, 157]$

iii) Random element as pivot:-

A [157, 110, 147, 122, 111, 149, 151, 141, 123, 117, 133]
 \downarrow
 L₁ Pivot R

Swap(L, R)

i) [157, 110, 147, 122, 111, 149, 151, 141, 123, 117, 133]
 \downarrow
 L₁ P R

Swap(P, R)

ii) [157, 110, 147, 122, 111, 149, 117, 141, 123, 151, 133]
 \downarrow
 L₁ P R

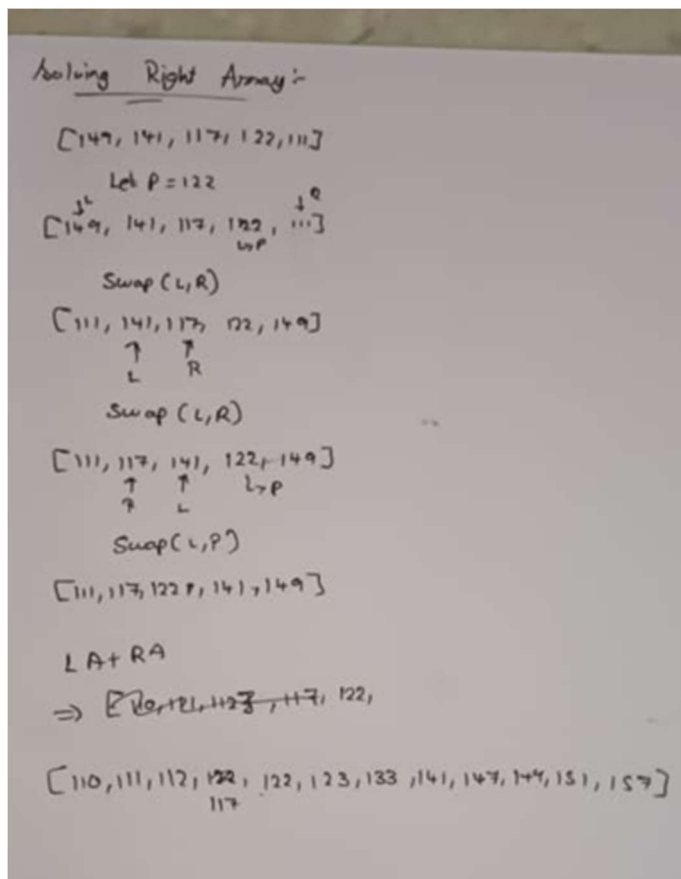
Left Array

[157, 110, 147, 122, 111, 149, 117, 141, 123, 151, 133]
 \downarrow
 we take 157 as pivot L₁ P

iii) [157, 110, 147, 122, 111, 149, 117, 141, 123, 151, 133]
 \downarrow
 L₁ R L

Swap(L, R)

iv) [117, 110, 147, 122, 111, 149, 151, 141, 123, 157, 133]
 \downarrow
 L₁ L₂ L₃ L₄ L₅ L₆ L₇ L₈ L₉ L₁₀ L₁₁ L₁₂ L₁₃ L₁₄ L₁₅ L₁₆ L₁₇ L₁₈ L₁₉ L₂₀ L₂₁ L₂₂ L₂₃ L₂₄ L₂₅ L₂₆ L₂₇ L₂₈ L₂₉ L₃₀ L₃₁ L₃₂ L₃₃ L₃₄ L₃₅ L₃₆ L₃₇ L₃₈ L₃₉ L₄₀ L₄₁ L₄₂ L₄₃ L₄₄ L₄₅ L₄₆ L₄₇ L₄₈ L₄₉ L₅₀ L₅₁ L₅₂ L₅₃ L₅₄ L₅₅ L₅₆ L₅₇ L₅₈ L₅₉ L₆₀ L₆₁ L₆₂ L₆₃ L₆₄ L₆₅ L₆₆ L₆₇ L₆₈ L₆₉ L₇₀ L₇₁ L₇₂ L₇₃ L₇₄ L₇₅ L₇₆ L₇₇ L₇₈ L₇₉ L₈₀ L₈₁ L₈₂ L₈₃ L₈₄ L₈₅ L₈₆ L₈₇ L₈₈ L₈₉ L₉₀ L₉₁ L₉₂ L₉₃ L₉₄ L₉₅ L₉₆ L₉₇ L₉₈ L₉₉ L₁₀₀ L₁₀₁ L₁₀₂ L₁₀₃ L₁₀₄ L₁₀₅ L₁₀₆ L₁₀₇ L₁₀₈ L₁₀₉ L₁₁₀ L₁₁₁ L₁₁₂ L₁₁₃ L₁₁₄ L₁₁₅ L₁₁₆ L₁₁₇ L₁₁₈ L₁₁₉ L₁₂₀ L₁₂₁ L₁₂₂ L₁₂₃ L₁₂₄ L₁₂₅ L₁₂₆ L₁₂₇ L₁₂₈ L₁₂₉ L₁₃₀ L₁₃₁ L₁₃₂ L₁₃₃ L₁₃₄ L₁₃₅ L₁₃₆ L₁₃₇ L₁₃₈ L₁₃₉ L₁₄₀ L₁₄₁ L₁₄₂ L₁₄₃ L₁₄₄ L₁₄₅ L₁₄₆ L₁₄₇ L₁₄₈ L₁₄₉ L₁₅₀ L₁₅₁ L₁₅₂ L₁₅₃ L₁₅₄ L₁₅₅ L₁₅₆ L₁₅₇ L₁₅₈ L₁₅₉ L₁₆₀ L₁₆₁ L₁₆₂ L₁₆₃ L₁₆₄ L₁₆₅ L₁₆₆ L₁₆₇ L₁₆₈ L₁₆₉ L₁₇₀ L₁₇₁ L₁₇₂ L₁₇₃ L₁₇₄ L₁₇₅ L₁₇₆ L₁₇₇ L₁₇₈ L₁₇₉ L₁₈₀ L₁₈₁ L₁₈₂ L₁₈₃ L₁₈₄ L₁₈₅ L₁₈₆ L₁₈₇ L₁₈₈ L₁₈₉ L₁₉₀ L₁₉₁ L₁₉₂ L₁₉₃ L₁₉₄ L₁₉₅ L₁₉₆ L₁₉₇ L₁₉₈ L₁₉₉ L₂₀₀ L₂₀₁ L₂₀₂ L₂₀₃ L₂₀₄ L₂₀₅ L₂₀₆ L₂₀₇ L₂₀₈ L₂₀₉ L₂₁₀ L₂₁₁ L₂₁₂ L₂₁₃ L₂₁₄ L₂₁₅ L₂₁₆ L₂₁₇ L₂₁₈ L₂₁₉ L₂₂₀ L₂₂₁ L₂₂₂ L₂₂₃ L₂₂₄ L₂₂₅ L₂₂₆ L₂₂₇ L₂₂₈ L₂₂₉ L₂₃₀ L₂₃₁ L₂₃₂ L₂₃₃ L₂₃₄ L₂₃₅ L₂₃₆ L₂₃₇ L₂₃₈ L₂₃₉ L₂₄₀ L₂₄₁ L₂₄₂ L₂₄₃ L₂₄₄ L₂₄₅ L₂₄₆ L₂₄₇ L₂₄₈ L₂₄₉ L₂₅₀ L₂₅₁ L₂₅₂ L₂₅₃ L₂₅₄ L₂₅₅ L₂₅₆ L₂₅₇ L₂₅₈ L₂₅₉ L₂₆₀ L₂₆₁ L₂₆₂ L₂₆₃ L₂₆₄ L₂₆₅ L₂₆₆ L₂₆₇ L₂₆₈ L₂₆₉ L₂₇₀ L₂₇₁ L₂₇₂ L₂₇₃ L₂₇₄ L₂₇₅ L₂₇₆ L₂₇₇ L₂₇₈ L₂₇₉ L₂₈₀ L₂₈₁ L₂₈₂ L₂₈₃ L₂₈₄ L₂₈₅ L₂₈₆ L₂₈₇ L₂₈₈ L₂₈₉ L₂₉₀ L₂₉₁ L₂₉₂ L₂₉₃ L₂₉₄ L₂₉₅ L₂₉₆ L₂₉₇ L₂₉₈ L₂₉₉ L₃₀₀ L₃₀₁ L₃₀₂ L₃₀₃ L₃₀₄ L₃₀₅ L₃₀₆ L₃₀₇ L₃₀₈ L₃₀₉ L₃₁₀ L₃₁₁ L₃₁₂ L₃₁₃ L₃₁₄ L₃₁₅ L₃₁₆ L₃₁₇ L₃₁₈ L₃₁₉ L₃₂₀ L₃₂₁ L₃₂₂ L₃₂₃ L₃₂₄ L₃₂₅ L₃₂₆ L₃₂₇ L₃₂₈ L₃₂₉ L₃₃₀ L₃₃₁ L₃₃₂ L₃₃₃ L₃₃₄ L₃₃₅ L₃₃₆ L₃₃₇ L₃₃₈ L₃₃₉ L₃₄₀ L₃₄₁ L₃₄₂ L₃₄₃ L₃₄₄ L₃₄₅ L₃₄₆ L₃₄₇ L₃₄₈ L₃₄₉ L₃₅₀ L₃₅₁ L₃₅₂ L₃₅₃ L₃₅₄ L₃₅₅ L₃₅₆ L₃₅₇ L₃₅₈ L₃₅₉ L₃₆₀ L₃₆₁ L₃₆₂ L₃₆₃ L₃₆₄ L₃₆₅ L₃₆₆ L₃₆₇ L₃₆₈ L₃₆₉ L₃₇₀ L₃₇₁ L₃₇₂ L₃₇₃ L₃₇₄ L₃₇₅ L₃₇₆ L₃₇₇ L₃₇₈ L₃₇₉ L₃₈₀ L₃₈₁ L₃₈₂ L₃₈₃ L₃₈₄ L₃₈₅ L₃₈₆ L₃₈₇ L₃₈₈ L₃₈₉ L₃₉₀ L₃₉₁ L₃₉₂ L₃₉₃ L₃₉₄ L₃₉₅ L₃₉₆ L₃₉₇ L₃₉₈ L₃₉₉ L₄₀₀ L₄₀₁ L₄₀₂ L₄₀₃ L₄₀₄ L₄₀₅ L₄₀₆ L₄₀₇ L₄₀₈ L₄₀₉ L₄₁₀ L₄₁₁ L₄₁₂ L₄₁₃ L₄₁₄ L₄₁₅ L₄₁₆ L₄₁₇ L₄₁₈ L₄₁₉ L₄₂₀ L₄₂₁ L₄₂₂ L₄₂₃ L₄₂₄ L₄₂₅ L₄₂₆ L₄₂₇ L₄₂₈ L₄₂₉ L₄₃₀ L₄₃₁ L₄₃₂ L₄₃₃ L₄₃₄ L₄₃₅ L₄₃₆ L₄₃₇ L₄₃₈ L₄₃₉ L₄₄₀ L₄₄₁ L₄₄₂ L₄₄₃ L₄₄₄ L₄₄₅ L₄₄₆ L₄₄₇ L₄₄₈ L₄₄₉ L₄₅₀ L₄₅₁ L₄₅₂ L₄₅₃ L₄₅₄ L₄₅₅ L₄₅₆ L₄₅₇ L₄₅₈ L₄₅₉ L₄₆₀ L₄₆₁ L₄₆₂ L₄₆₃ L₄₆₄ L₄₆₅ L₄₆₆ L₄₆₇ L₄₆₈ L₄₆₉ L₄₇₀ L₄₇₁ L₄₇₂ L₄₇₃ L₄₇₄ L₄₇₅ L₄₇₆ L₄₇₇ L₄₇₈ L₄₇₉ L₄₈₀ L₄₈₁ L₄₈₂ L₄₈₃ L₄₈₄ L₄₈₅ L₄₈₆ L₄₈₇ L₄₈₈ L₄₈₉ L₄₉₀ L₄₉₁ L₄₉₂ L₄₉₃ L₄₉₄ L₄₉₅ L₄₉₆ L₄₉₇ L₄₉₈ L₄₉₉ L₅₀₀ L₅₀₁ L₅₀₂ L₅₀₃ L₅₀₄ L₅₀₅ L₅₀₆ L₅₀₇ L₅₀₈ L₅₀₉ L₅₁₀ L₅₁₁ L₅₁₂ L₅₁₃ L₅₁₄ L₅₁₅ L₅₁₆ L₅₁₇ L₅₁₈ L₅₁₉ L₅₂₀ L₅₂₁ L₅₂₂ L₅₂₃ L₅₂₄ L₅₂₅ L₅₂₆ L₅₂₇ L₅₂₈ L₅₂₉ L₅₃₀ L₅₃₁ L₅₃₂ L₅₃₃ L₅₃₄ L₅₃₅ L₅₃₆ L₅₃₇ L₅₃₈ L₅₃₉ L₅₄₀ L₅₄₁ L₅₄₂ L₅₄₃ L₅₄₄ L₅₄₅ L₅₄₆ L₅₄₇ L₅₄₈ L₅₄₉ L₅₅₀ L₅₅₁ L₅₅₂ L₅₅₃ L₅₅₄ L₅₅₅ L₅₅₆ L₅₅₇ L₅₅₈ L₅₅₉ L₅₆₀ L₅₆₁ L₅₆₂ L₅₆₃ L₅₆₄ L₅₆₅ L₅₆₆ L₅₆₇ L₅₆₈ L₅₆₉ L₅₇₀ L₅₇₁ L₅₇₂ L₅₇₃ L₅₇₄ L₅₇₅ L₅₇₆ L₅₇₇ L₅₇₈ L₅₇₉ L₅₈₀ L₅₈₁ L₅₈₂ L₅₈₃ L₅₈₄ L₅₈₅ L₅₈₆ L₅₈₇ L₅₈₈ L₅₈₉ L₅₉₀ L₅₉₁ L₅₉₂ L₅₉₃ L₅₉₄ L₅₉₅ L₅₉₆ L₅₉₇ L₅₉₈ L₅₉₉ L₆₀₀ L₆₀₁ L₆₀₂ L₆₀₃ L₆₀₄ L₆₀₅ L₆₀₆ L₆₀₇ L₆₀₈ L₆₀₉ L₆₁₀ L₆₁₁ L₆₁₂ L₆₁₃ L₆₁₄ L₆₁₅ L₆₁₆ L₆₁₇ L₆₁₈ L₆₁₉ L₆₂₀ L₆₂₁ L₆₂₂ L₆₂₃ L₆₂₄ L₆₂₅ L₆₂₆ L₆₂₇ L₆₂₈ L₆₂₉ L₆₃₀ L₆₃₁ L₆₃₂ L₆₃₃ L₆₃₄ L₆₃₅ L₆₃₆ L₆₃₇ L₆₃₈ L₆₃₉ L₆₄₀ L₆₄₁ L₆₄₂ L₆₄₃ L₆₄₄ L₆₄₅ L₆₄₆ L₆₄₇ L₆₄₈ L₆₄₉ L₆₅₀ L₆₅₁ L₆₅₂ L₆₅₃ L₆₅₄ L₆₅₅ L₆₅₆ L₆₅₇ L₆₅₈ L₆₅₉ L₆₆₀ L₆₆₁ L₆₆₂ L₆₆₃ L₆₆₄ L₆₆₅ L₆₆₆ L₆₆₇ L₆₆₈ L₆₆₉ L₆₇₀ L₆₇₁ L₆₇₂ L₆₇₃ L₆₇₄ L₆₇₅ L₆₇₆ L₆₇₇ L₆₇₈ L₆₇₉ L₆₈₀ L₆₈₁ L₆₈₂ L₆₈₃ L₆₈₄ L₆₈₅ L₆₈₆ L₆₈₇ L₆₈₈ L₆₈₉ L₆₉₀ L₆₉₁ L₆₉₂ L₆₉₃ L₆₉₄ L₆₉₅ L₆₉₆ L₆₉₇ L₆₉₈ L₆₉₉ L₇₀₀ L₇₀₁ L₇₀₂ L₇₀₃ L₇₀₄ L₇₀₅ L₇₀₆ L₇₀₇ L₇₀₈ L₇₀₉ L₇₁₀ L₇₁₁ L₇₁₂ L₇₁₃ L₇₁₄ L₇₁₅ L₇₁₆ L₇₁₇ L₇₁₈ L₇₁₉ L₇₂₀ L₇₂₁ L₇₂₂ L₇₂₃ L₇₂₄ L₇₂₅ L₇₂₆ L₇₂₇ L₇₂₈ L₇₂₉ L₇₃₀ L₇₃₁ L₇₃₂ L₇₃₃ L₇₃₄ L₇₃₅ L₇₃₆ L₇₃₇ L₇₃₈ L₇₃₉ L₇₄₀ L₇₄₁ L₇₄₂ L₇₄₃ L₇₄₄ L₇₄₅ L₇₄₆ L₇₄₇ L₇₄₈ L₇₄₉ L₇₅₀ L₇₅₁ L₇₅₂ L₇₅₃ L₇₅₄ L₇₅₅ L₇₅₆ L₇₅₇ L₇₅₈ L₇₅₉ L₇₆₀ L₇₆₁ L₇₆₂ L₇₆₃ L₇₆₄ L₇₆₅ L₇₆₆ L₇₆₇ L₇₆₈ L₇₆₉ L₇₇₀ L₇₇₁ L₇₇₂ L₇₇₃ L₇₇₄ L₇₇₅ L₇₇₆ L₇₇₇ L₇₇₈ L₇₇₉ L₇₈₀ L₇₈₁ L₇₈₂ L₇₈₃ L₇₈₄ L₇₈₅ L₇₈₆ L₇₈₇ L₇₈₈ L₇₈₉ L₇₉₀ L₇₉₁ L₇₉₂ L₇₉₃ L₇₉₄ L₇₉₅ L₇₉₆ L₇₉₇ L₇₉₈ L₇₉₉ L₈₀₀ L₈₀₁ L₈₀₂ L₈₀₃ L₈₀₄ L₈₀₅ L₈₀₆ L₈₀₇ L₈₀₈ L₈₀₉ L₈₁₀ L₈₁₁ L₈₁₂ L₈₁₃ L₈₁₄ L₈₁₅ L₈₁₆ L₈₁₇ L₈₁₈ L₈₁₉ L₈₂₀ L₈₂₁ L₈₂₂ L₈₂₃ L₈₂₄ L₈₂₅ L₈₂₆ L₈₂₇ L₈₂₈ L₈₂₉ L₈₃₀ L₈₃₁ L₈₃₂ L₈₃₃ L₈₃₄ L₈₃₅ L₈₃₆ L₈₃₇ L₈₃₈ L₈₃₉ L₈₄₀ L₈₄₁ L₈₄₂ L₈₄₃ L₈₄₄ L₈₄₅ L₈₄₆ L₈₄₇ L₈₄₈ L₈₄₉ L₈₅₀ L₈₅₁ L₈₅₂ L₈₅₃ L₈₅₄ L₈₅₅ L₈₅₆ L₈₅₇ L₈₅₈ L₈₅₉ L₈₆₀ L₈₆₁ L₈₆₂ L₈₆₃ L₈₆₄ L₈₆₅ L₈₆₆ L₈₆₇ L₈₆₈ L₈₆₉ L₈₇₀ L₈₇₁ L₈₇₂ L₈₇₃ L₈₇₄ L₈₇₅ L₈₇₆ L₈₇₇ L₈₇₈ L₈₇₉ L₈₈₀ L₈₈₁ L₈₈₂ L₈₈₃ L₈₈₄ L₈₈₅ L₈₈₆ L₈₈₇ L₈₈₈ L₈₈₉ L₈₉₀ L₈₉₁ L₈₉₂ L₈₉₃ L₈₉₄ L₈₉₅ L₈₉₆ L₈₉₇ L₈₉₈ L₈₉₉ L₉₀₀ L₉₀₁ L₉₀₂ L₉₀₃ L₉₀₄ L₉₀₅ L₉₀₆ L₉₀₇ L₉₀₈ L₉₀₉ L₉₁₀ L₉₁₁ L₉₁₂ L₉₁₃ L₉₁₄ L₉₁₅ L₉₁₆ L₉₁₇ L₉₁₈ L₉₁₉ L₉₂₀ L₉₂₁ L₉₂₂ L₉₂₃ L₉₂₄ L₉₂₅ L₉₂₆ L₉₂₇ L₉₂₈ L₉₂₉ L₉₃₀ L₉₃₁ L₉₃₂ L₉₃₃ L₉₃₄ L₉₃₅ L₉₃₆ L₉₃₇ L₉₃₈ L₉₃₉ L₉₄₀ L₉₄₁ L₉₄₂ L₉₄₃ L₉₄₄ L₉₄₅ L₉₄₆ L₉₄₇ L₉₄₈ L₉₄₉ L₉₅₀ L₉₅₁ L₉₅₂ L₉₅₃ L₉₅₄ L₉₅₅ L₉₅₆ L₉₅₇ L₉₅₈ L₉₅₉ L₉₆₀ L₉₆₁ L₉₆₂ L₉₆₃ L₉₆₄ L₉₆₅ L₉₆₆ L₉₆₇ L₉₆₈ L₉₆₉ L₉₇₀ L₉₇₁ L₉₇₂ L₉₇₃ L₉₇₄ L₉₇₅ L₉₇₆ L₉₇₇ L₉₇₈ L₉₇₉ L₉₈₀ L₉₈₁ L₉₈₂ L₉₈₃ L₉₈₄ L₉₈₅ L₉₈₆ L₉₈₇ L₉₈₈ L₉₈₉ L₉₉₀ L₉₉₁ L₉₉₂ L₉₉₃ L₉₉₄ L₉₉₅ L₉₉₆ L₉₉₇ L₉₉₈ L₉₉₉ L₁₀₀₀ L₁₀₀₁ L₁₀₀₂ L₁₀₀₃ L₁₀₀₄ L₁₀₀₅ L₁₀₀₆ L₁₀₀₇ L₁₀₀₈ L₁₀₀₉ L₁₀₁₀ L₁₀₁₁ L₁₀₁₂ L₁₀₁₃ L₁₀₁₄ L₁₀₁₅ L₁₀₁₆ L₁₀₁₇ L₁₀₁₈ L₁₀₁₉ L₁₀₂₀ L₁₀₂₁ L₁₀₂₂ L₁₀₂₃ L₁₀₂₄ L₁₀₂₅ L₁₀₂₆ L₁₀₂₇ L₁₀₂₈ L₁₀₂₉ L₁₀₃₀ L₁₀₃₁ L₁₀₃₂ L₁₀₃₃ L₁₀₃₄ L₁₀₃₅ L₁₀₃₆ L₁₀₃₇ L₁₀₃₈ L₁₀₃₉ L₁₀₄₀ L₁₀₄₁ L₁₀₄₂ L₁₀₄₃ L₁₀₄₄ L₁₀₄₅ L₁₀₄₆ L₁₀₄₇ L₁₀₄₈ L₁₀₄₉ L₁₀₅₀ L₁₀₅₁ L₁₀₅₂ L₁₀₅₃ L₁₀₅₄ L₁₀₅₅ L₁₀₅₆ L₁₀₅₇ L₁₀₅₈ L₁₀₅₉ L₁₀₆₀ L₁₀₆₁ L₁₀₆₂ L₁₀₆₃ L₁₀₆₄ L₁₀₆₅ L₁₀₆₆ L₁₀₆₇ L₁₀₆₈ L₁₀₆₉ L₁₀₇₀ L₁₀₇₁ L₁₀₇₂ L₁₀₇₃ L₁₀₇₄ L₁₀₇₅ L₁₀₇₆ L₁₀₇₇ L₁₀₇₈ L₁₀₇₉ L₁₀₈₀ L₁₀₈₁ L₁₀₈₂ L₁₀₈₃ L₁₀₈₄ L₁₀₈₅ L₁₀₈₆ L₁₀₈₇ L₁₀₈₈ L₁₀₈₉ L₁₀₉₀ L₁₀₉₁ L₁₀₉₂ L₁₀₉₃ L₁₀₉₄ L₁₀₉₅ L₁₀₉₆ L₁₀₉₇ L₁₀₉₈ L₁₀₉₉ L₁₁₀₀ L₁₁₀₁ L₁₁₀₂ L₁₁₀₃ L₁₁₀₄ L₁₁₀₅ L₁₁₀₆ L₁₁₀₇ L₁₁₀₈ L₁₁₀₉ L₁₁₁₀ L₁₁₁₁ L₁₁₁₂ L₁₁₁₃ L₁₁₁₄ L₁₁₁₅ L₁₁₁₆ L₁₁₁₇ L₁₁₁₈ L₁₁₁₉ L₁₁₂₀ L₁₁₂₁ L₁₁₂₂ L₁₁₂₃ L₁₁₂₄ L₁₁₂₅ L₁₁₂₆ L₁₁₂₇ L₁₁₂₈ L₁₁₂₉ L₁₁₃₀ L₁₁₃₁ L₁₁₃₂ L₁₁₃₃ L₁₁₃₄ L₁₁₃₅ L₁₁₃₆ L₁₁₃₇ L₁₁₃₈ L₁₁₃₉ L₁₁₄₀ L₁₁₄₁ L₁₁₄₂ L₁₁₄₃ L₁₁₄₄ L



ANALYSIS:

Time Complexity: $O(N^2)$: It takes $O(N^2)$ time in the worst case because each partition processes all elements while reducing the problem size by only one element at a time.

Space Complexity: $O(N)$: $O(N)$ space in the worst case because recursive calls can go as deep as the number of elements when partitions are unbalanced