

Design and Analysis of Algorithms

Week-6

Name: Musali Reddy Manish Reddy

Roll.no :CH.SC.U4CSE24129

Quick sort:

(starting number as pivot element)

Code:

```
//CH.SC.U4CSE24129
#include <stdio.h>
int partitionFirst(int a[], int low, int high)
{
    int pivot, i, j, temp;
    pivot = a[low];
    i = low + 1;
    j = high;

    while (i <= j)
    {
        while (a[i] <= pivot && i <= high)
            i++;
        while (a[j] > pivot)
            j--;
        if (i < j)
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
}
```

```
a[j] = temp;
}
}

temp = a[low];
a[low] = a[j];
a[j] = temp;

return j;
}

void quickSortFirst(int a[], int low, int high)
{
    int p;
    if (low < high)
    {
        p = partitionFirst(a, low, high);
        quickSortFirst(a, low, p - 1);
        quickSortFirst(a, p + 1, high);
    }
}

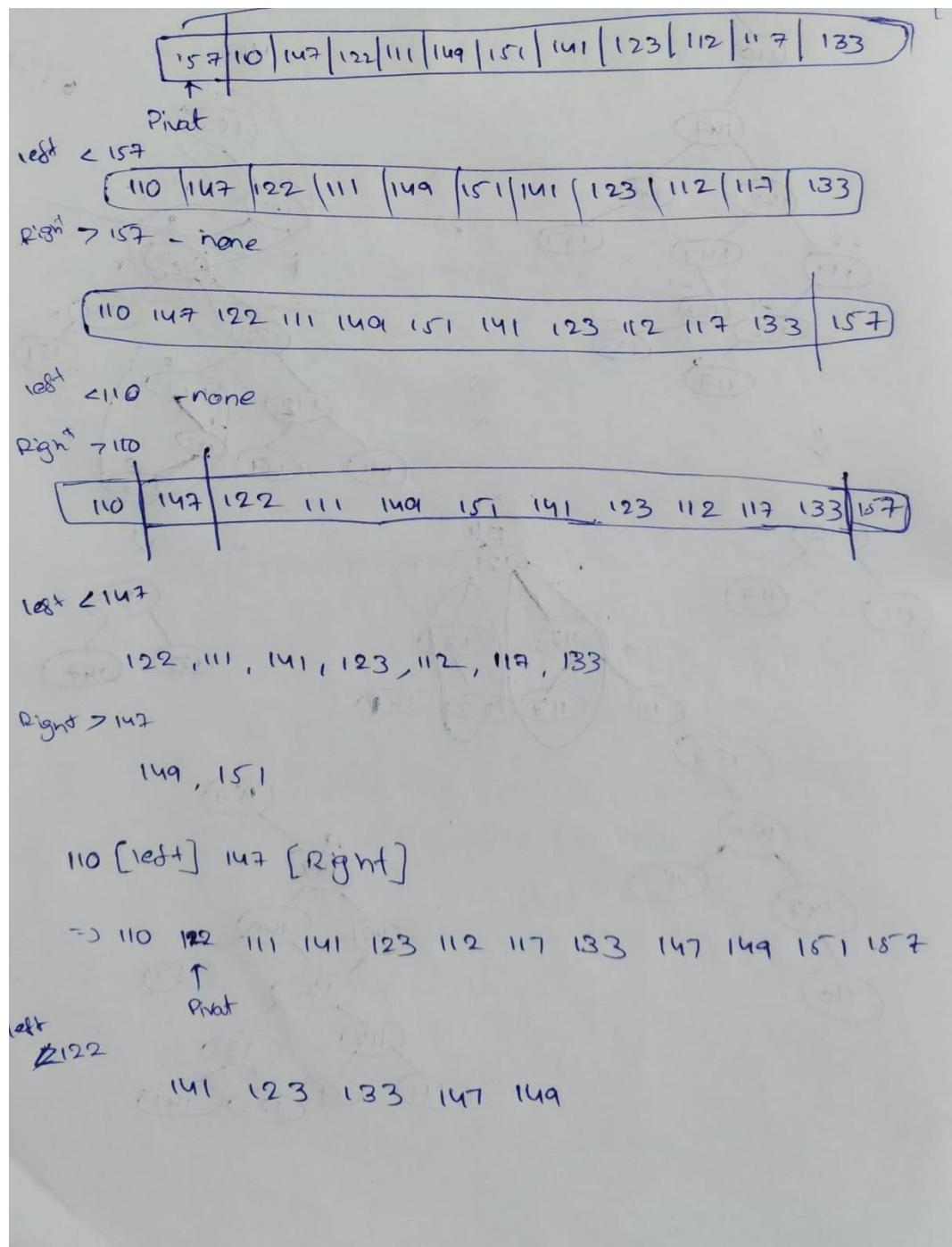
int main()
{
    int a[12] = {157,110,147,122,111,149,151,141,123,112,117,133};
    int i;
    quickSortFirst(a, 0, 11);
    printf("Sorted Array:\n");
    for (i = 0; i < 12; i++)
        printf("%d ", a[i]);
    return 0;
}
```

Output:

Sorted Array:

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

Handwritten:



no. [left] 147 [right] 157 < 4 digits, switch
 1482 < 122 } (digit tree, end bin, 0) to 2nd tree
 111 112 117 { Smart, i, j, k, l, m
 right > 122 $(\text{end})_0 = \text{good}$
~~113~~ 141 123 133 149 $i + \text{end} = i$
~~113~~ 141 123 133 149 $\text{good} = l$
 no. [left] 122 [right] } ($i = 1$) switch
 110 111 112 117 122 | 141 123 133 147 149 151 157
 ↓ (bin) < [1] 0) switch
 pivot $i = 1$
 < 141 } ($i = 1$) fe
 123 133 $(i)_0 = \text{good}$
 > 141 $(i)_0 = (i)_0$
 147 149 151 157 $\text{good} = (i)_0$
 no. 110 111 112 117 122 123 133 141 | 147 149 151 157
 ↓ $(\text{end})_0 = \text{Pivot good}$
 < 147 - none $(i)_0 = (\text{end})_0$
 > 147 $\text{good} = (i)_0$
 149 151 157 $i \text{ invalid}$
 no. 110 111 112 117 122 123 133 141 147 | 149 151 157
 } (digit tree, end bin, 0) to 3rd tree
 ↓ $i = 1$
 pivot
 left pivot right } (right > end) fe
~~113~~ $(\text{digit}, \text{end}, 0) \text{ to 3rd tree} = 0$
 $(1 - 9, \text{end}, 0) \text{ to 3rd tree} = 0$
 $(\text{pivot}, 149, 0) \text{ to 3rd tree} = 0$

```

#include <cs50.h>
int PartitionFirst (int a[], int low, int high) {
    int Pivot, i, j, tempP;
    Pivot = a[low];
    i = low + 1;
    j = high;
    while (i <= j) {
        while (a[i] <= Pivot && i <= high)
            i++;
        while (a[j] >= Pivot)
            j--;
        if (i < j) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[low];
    a[low] = a[i];
    a[i] = temp;
    return j;
}
void quickSortFirst (int a[], int low, int high) {
    int P;
    if (low < high) {
        P = PartitionFirst (a, low, high);
        quickSortFirst (a, low, P - 1);
        quickSortFirst (a, P + 1, high);
    }
}

```

```

int main () {
    int a[12] = {157, 110, 147, 122, 111, 149, 151, 141, 123, 112,
                 135, 117, 133};
    int i;
    quickSortFirst (a, 0, 11);
    printf ("Sorted Array:\n");
    for (i = 0; i < 12; i++)
        printf ("%d\n", a[i]);
    return 0;
}

```

(1) P = 9mod
(2) P = 110

Quick sort:

(last number as pivot element)

Code:

```
//CH.SC.U4CSE24129
#include <stdio.h>

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

void quickSortLast(int a[], int low, int high)
{
    int p;
```

```

if (low < high)
{
    p = partitionLast(a, low, high);
    quickSortLast(a, low, p - 1);
    quickSortLast(a, p + 1, high);
}

int main()
{
    int a[12] = {157,110,147,122,111,149,151,141,123,112,117,133};
    int i;
    quickSortLast(a, 0, 11);
    printf("Sorted Array:\n");
    for (i = 0; i < 12; i++)
        printf("%d ", a[i]);
    return 0;
}

```

Output:

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

Handwritten:

$$gmat = [l]e$$

Last Element as PIVOT

$$[l+j]P = gmat$$

$$[l+j]P = [l+j]P$$

↑

Pivot

$$122 < 133$$

$$gmat = [l+j]P$$

$j+i$ number

$$\{ 10 \ 157 \ 147 \ 122 \ 111 \ 149 \ 151 \ 141 \ 123 \ 112 \ 117 \ 133$$

$$\} (684 \text{ tai}, \text{and} \ 10) \ (70 \text{ tai}) \ \text{feel} \ \text{to} \ \text{skip} \ \text{step} \ \text{pivot}$$

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

{8 tai}

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

$$(148 \text{ tai}, \text{and} \ 10) \ (70 \text{ tai}) \ \text{feel} \ \text{to} \ \text{skip} \ 123 < 133 = 9$$

$$(1+9, \text{and} \ 10) \ \text{feel} \ \text{to} \ \text{skip}$$

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

$$112 < 133$$

$$117 < 133$$

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

{1 number}

{1 tai}

(11, 0, 0) feel to skip

(("ai" point before") 800s)

((+1) {s121 : 0=1} reb

((G) "b0") front

{0 end}

```

# include <stdio.h>
int partitionLast (int a[], int low, int high) {
    int Pivot, i, j, temp;
    Pivot = a[high];
    i = low - 1;
    for (j = low; j < high; j++) {
        if (a[j] <= Pivot) {
            i++;
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[i + 1];
    a[i + 1] = a[high];
    a[high] = temp;
    return i + 1;
}
void quickSortLast (int a[], int low, int high) {
    int P;
    if (low < high) {
        P = partitionLast (a, low, high);
        quickSortLast (a, low, P - 1);
        quickSortLast (a, P + 1, high);
    }
}
int main () {
    int a[12] = {157, 110, 147, 122, 111, 149, 151, 141, 123, 127, 119, 115};
    int i;
    quickSortLast (a, 0, 11);
    printf ("sorted Array :\n");
    for (i = 0; i < 12; i++)
        printf ("%d", a[i]);
    return 0;
}

```

Quick sort:

(Middle number as pivot element)

Code:

```
//CH.SC.U4CSE24129
#include <stdio.h>

int partitionMiddle(int a[], int low, int high)
{
    int mid, pivot, i, j, temp;
    mid = (low + high) / 2;
    pivot = a[mid];
    i = low;
    j = high;
    while (i <= j)
    {
        while (a[i] < pivot)
            i++;

        while (a[j] > pivot)
            j--;

        if (i <= j)
        {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;

            i++;
            j--;
        }
    }
}
```

```

    }

    return i;
}

void quickSortMiddle(int a[], int low, int high)
{
    int index;

    if (low < high)
    {
        index = partitionMiddle(a, low, high);

        if (low < index - 1)
            quickSortMiddle(a, low, index - 1);

        if (index < high)
            quickSortMiddle(a, index, high);
    }
}

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

    int i;
    quickSortMiddle(a, 0, 11);
    printf("Sorted Array:\n");
    for (i = 0; i < 12; i++)
        printf("%d ", a[i]);
    return 0;
}

```

Output:

<pre> Sorted Array: 110 111 112 117 122 123 133 141 147 149 151 157 -----</pre>

Handwritten:

iii) middle element as pivot \leftarrow either smallest

157 110 147 122 111 (mag) 151 141 123 112 117 133

$$\text{midst Index} = \frac{o+1}{2} (\text{avg} + \text{avg}) + \text{bias}$$

$$\text{Pivot} = \text{1uq}$$

$\begin{array}{r} \cancel{149} \\ 110 \quad 147 \quad 122 \quad 111 \quad 133 \quad 149 \quad 123 \quad 112 \quad 117 \quad 149 \quad | \quad 151 \quad 157 \end{array}$

$$M \cdot D = 133$$

133

$$\begin{array}{cccccc}
 110 & 122 & 111 & \overset{123}{112} & 117 & \\
 \downarrow & & & & & \\
 M \cdot D = 111 & & & & & \\
 \end{array}
 \quad
 \begin{array}{c}
 P = S \cdot B \cdot B \\
 | \quad | \quad | \quad | \\
 133 \quad 147 \quad 141 \\
 \downarrow \\
 \text{Gest.} = (1) \cdot P
 \end{array}$$

$$M \cdot D = 111$$

110 111 112 113 112 113

$$M.D = 123$$

Diagram illustrating the relationship between the sequence terms and the series terms:

110	111	122	112	117	123				
110	111	112	117	123					

MD = 112 + 113 + 114 + 115 + 116 + 117 + 118 + 119 + 120 + 121

```

#include <stdio.h>
int PartitionMiddle(int a[], int low, int high) {
    int mid, Pivot, i, j, temp;
    mid = (low + high) / 2;
    Pivot = a[mid];
    i = low;
    j = high;
    while (i <= j) {
        while (a[i] < Pivot) {
            i++;
        }
        if (i <= j) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
        }
        i++;
        j--;
    }
    return i;
}

void quickSortMiddle(int a[], int low, int high) {
    int index;
    if (low < high) {
        index = PartitionMiddle(a, low, high);
        if (low < index - 1)
            quickSortMiddle(a, low, index - 1);
        if (index < high)
            quickSortMiddle(a, index, high);
    }
}

```

```

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

int i;
quickSortMiddle(a, 0, 11);
printf("Sorted Array:\n");
for (i = 0; i < 12; i++)
    printf("%d ", a[i]);
return 0;
}

```