

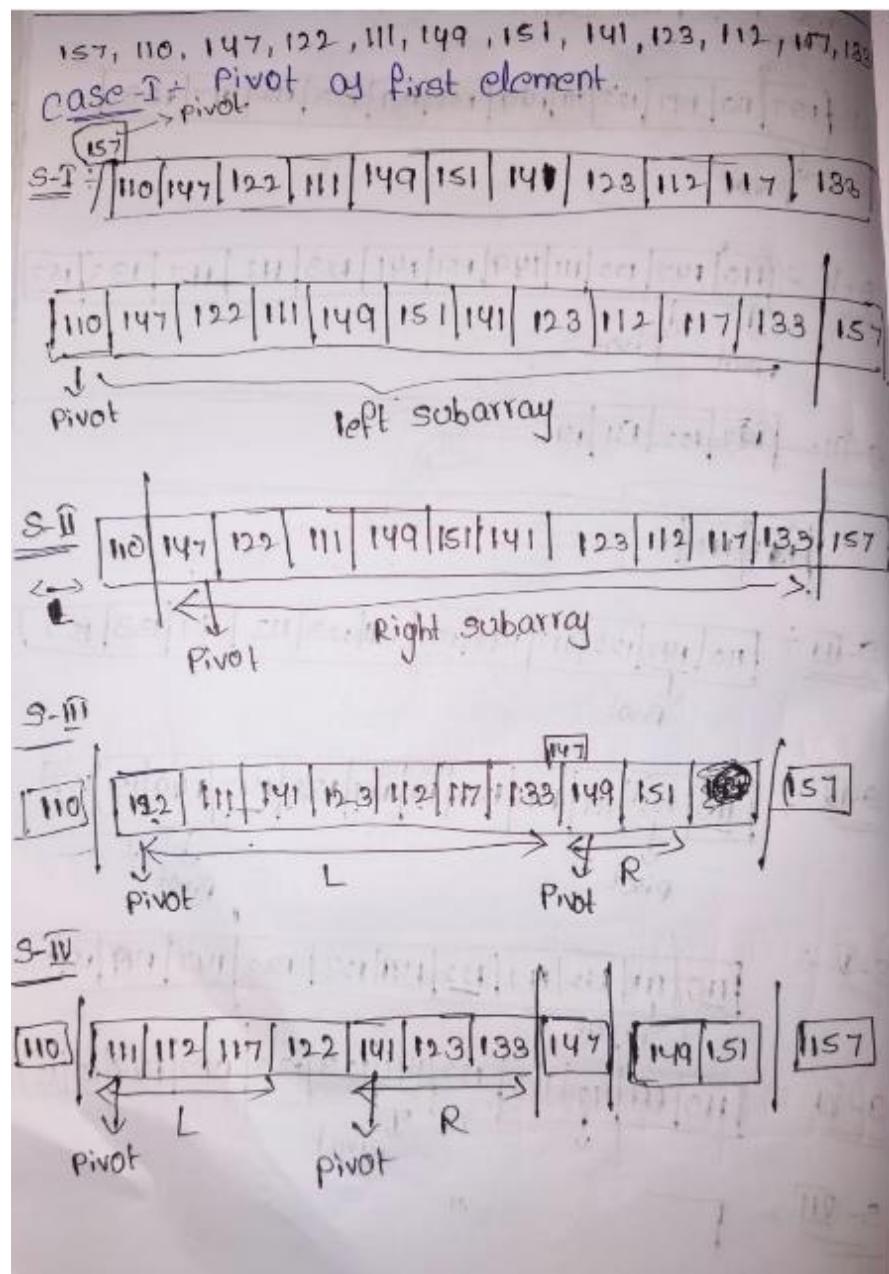
NAME : T. VASANTHA

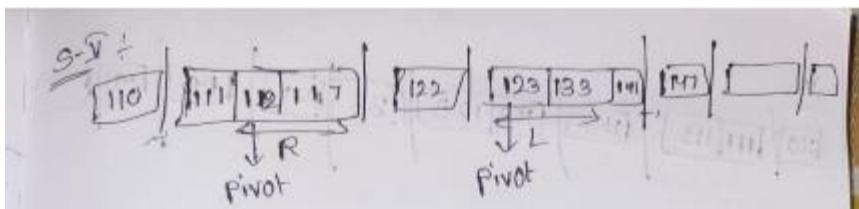
ROLL NO : CH.SC.U4CSE24147

WEEK - 6

QUICK SORT:

Logic:





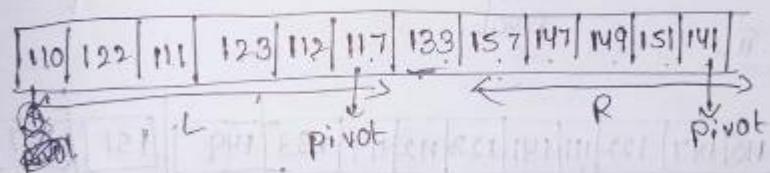
S-V :

| | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| [110] | [111] | [112] | [117] | [122] | [123] | [133] | [141] | [147] | [149] | [151] | [157] |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|

Case-II : take last element as pivot element

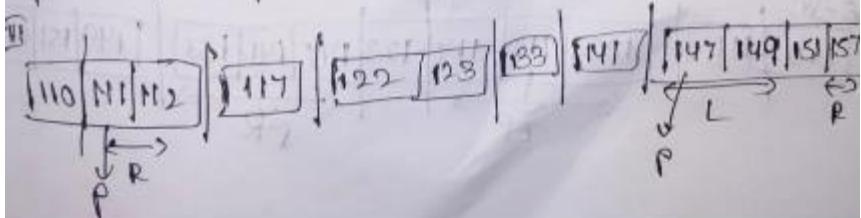
S-I :

| | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| [157] | [110] | [147] | [122] | [111] | [149] | [151] | [141] | [123] | [112] | [117] | [133] |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|



S-II :

| | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| [110] | [111] | [112] | [117] | [122] | [123] | [133] | [141] | [157] | [147] | [149] | [151] |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|



| | |
|-----------------|--|
| <u>S-I</u> | $\frac{110}{110} \boxed{111} 112 117 122 123 133 141 147 149 151 157$ |
| <u>Type-III</u> | take a random element as pivot |
| <u>S-II</u> | $\frac{\begin{matrix} 157 \\ 110 147 122 111 149 151 141 123 112 117 133 \end{matrix}}{\downarrow \text{pivot}}$ |
| <u>S-II</u> | $\frac{110 147 122 111 149 141 123 112 117 133 151 157}{\leftarrow \text{pivot} \quad \rightarrow \quad \leftarrow R}$ |
| <u>S-III</u> | $\frac{110 147 122 \boxed{111} 141 123 112 117 133 149 151 157}{\leftarrow \text{pivot L} \quad \rightarrow \quad \leftarrow \text{pivot R}}$ |
| <u>S-IV</u> | $\frac{110 110 147 122 141 123 112 117 133 149 151 157}{\leftarrow \quad \leftarrow \text{pivot R} \quad \rightarrow \quad \leftarrow \quad \leftarrow \text{pivot L} \quad \rightarrow \quad \leftarrow \quad \leftarrow \text{pivot R}}$ |

| | | | | | | | | | | | | |
|--|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| <u>S-V</u> : | | | | | | | | | | | | |
| <table border="1"> <tr> <td>110</td> <td>111</td> <td>112</td> <td>122</td> <td>117</td> <td>123</td> <td>133</td> <td>141</td> <td>147</td> <td>149</td> <td>151</td> <td>157</td> </tr> </table> | 110 | 111 | 112 | 122 | 117 | 123 | 133 | 141 | 147 | 149 | 151 | 157 |
| 110 | 111 | 112 | 122 | 117 | 123 | 133 | 141 | 147 | 149 | 151 | 157 | |

| | | | | | | | | | | | | | |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| <u>S-VI</u> : | | | | | | | | | | | | | |
| <table border="1"> <tr> <td>110</td> <td>111</td> <td>112</td> <td>147</td> <td>122</td> <td>123</td> <td>123</td> <td>133</td> <td>141</td> <td>147</td> <td>149</td> <td>151</td> <td>157</td> </tr> </table> | 110 | 111 | 112 | 147 | 122 | 123 | 123 | 133 | 141 | 147 | 149 | 151 | 157 |
| 110 | 111 | 112 | 147 | 122 | 123 | 123 | 133 | 141 | 147 | 149 | 151 | 157 | |

First Element as Pivot

Code :

```

#include <stdio.h>
int partition(int arr[], int low, int high) {
    int pivot = arr[low];
    int i = low + 1;
    int j = high;
    int temp;
    while (i <= j) {
        while (i <= j && arr[i] <= pivot)
            i++;
        while (arr[j] > pivot)
            j--;
        if (i < j) {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    arr[low] = arr[j];
    arr[j] = pivot;
    return j;
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int p = partition(arr, low, high);
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}
int main() {
    int arr[] = {157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133};
    int n = 12;
    int i;
    quickSort(arr, 0, n - 1);
    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}

```

Output :

```

Sorted array:
110 111 112 117 122 123 133 141 147 149 151 157
-----
Process exited after 0.5229 seconds with return value 0
Press any key to continue . . .

```

Last Element as Pivot

Code :

```
#include <stdio.h>
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    int temp;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return i + 1;
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int p = partition(arr, low, high);
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}
int main() {
    int arr[] = {157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133};
    int n = 12;
    int i;
    quickSort(arr, 0, n - 1);
    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Output :

```
Sorted array:
110 111 112 117 122 123 133 141 147 149 151 157
-----
Process exited after 0.543 seconds with return value 0
Press any key to continue . . .
```

Random Element as Pivot

Code :

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    int temp;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return i + 1;
}
int randomPartition(int arr[], int low, int high) {
    int r = low + rand() % (high - low + 1);
    int temp = arr[r];
    arr[r] = arr[high];
    arr[high] = temp;
    return partition(arr, low, high);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int p = randomPartition(arr, low, high);
        quickSort(arr, low, p - 1);
        quickSort(arr, p + 1, high);
    }
}
int main() {
    int arr[] = {157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133};
    int n = 12;
    int i;
    srand(time(0));
    quickSort(arr, 0, n - 1);
    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Output :

```
Sorted array:  
110 111 112 117 122 123 133 141 147 149 151 157  
-----  
Process exited after 2.224 seconds with return value 0  
Press any key to continue . . .
```

Conclusion:

Based on the array order, position of the pivot element changes.

So, the efficiency doesn't depend on the position of the pivot element changes