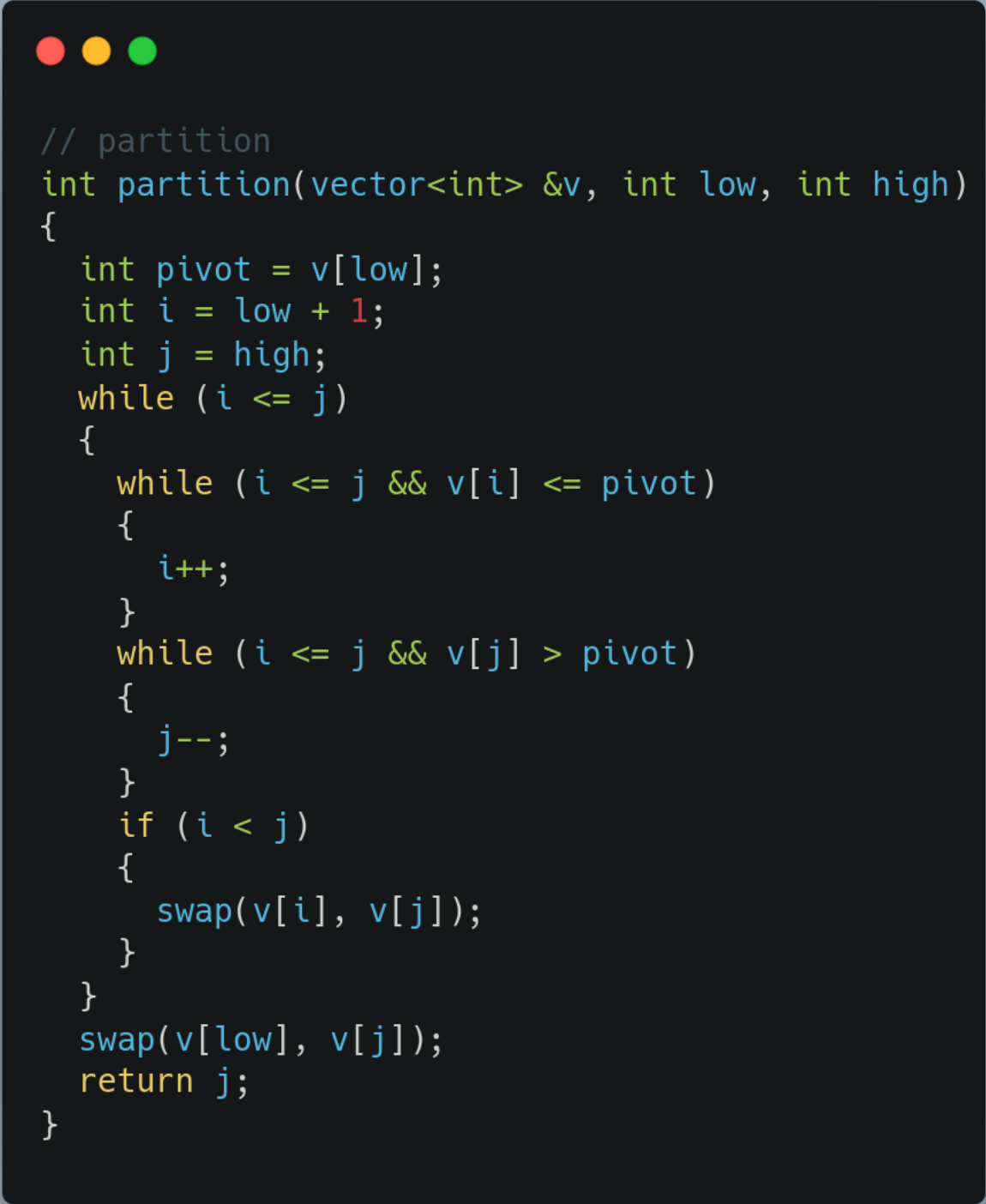


Lab 5 Report

Implementation and Analysis of Quick Sort
Algorithm

Quick Sort

Code:



```
// partition
int partition(vector<int> &v, int low, int high)
{
    int pivot = v[low];
    int i = low + 1;
    int j = high;
    while (i <= j)
    {
        while (i <= j && v[i] <= pivot)
        {
            i++;
        }
        while (i <= j && v[j] > pivot)
        {
            j--;
        }
        if (i < j)
        {
            swap(v[i], v[j]);
        }
    }
    swap(v[low], v[j]);
    return j;
}
```

This function above finds the pivot element and partitions the array.



```
// quicksort
void quicksort(vector<int> &v, int low, int high)
{
    if (low < high)
    {
        int pivot = partition(v, low, high);
        quicksort(v, low, pivot - 1);
        quicksort(v, pivot + 1, high);
    }
}
```

This above code is recursive and is responsible to sort the array with divide and conquer method.



```
// print array
void printArray(vector<int> &v)
{
    for (int i = 0; i < v.size(); i++)
    {
        cout << v[i] << " ";
    }
    cout << endl;
}
```

The above code prints the array.

```
int main()
{
    // generate random numbers
    // log the time to run the algorithm for 10 times
    vector<int> v;
    srand(time(0));
    for (int i = 0; i < 20; i++)
    {
        v.push_back(rand() % 1000);
    }
    cout << "Before sorting: ";
    printArray(v);
    double time1 = clock();
    quicksort(v, 0, v.size() - 1);
    double time2 = clock();
    cout << "After sorting: ";
    printArray(v);
    cout << "Length of array: " << v.size() << endl;
    cout << "Time taken: " << (double)((time2 - time1) / (CLOCKS_PER_SEC)) * 1000 << " ms" << endl;
    return 0;
}
```

The above code is the main code to generate the array and track the time of execution by the algorithm.

Result:

```
beyond@Infiverse:~/Desktop/Assignments/CE2020_Lab5_63_64$ ./a.out
Before sorting: 478 210 56 840 429 901 41 471 46 622 324 556 767 640 848 110 268 165 480 598
After sorting: 41 46 56 110 165 210 268 324 429 471 478 480 556 598 622 640 767 840 848 901
Length of array: 20
Time taken: 0.006 ms
```

Analysis of Quick Sort Algorithm

```
beyond@Infiverse:~/Desktop/Assignments/CE2020_Lab5_63_64$ ./a.out
*****
Iteration 1
Length of array: 100
Time taken: 0.028 ms
*****
Iteration 2
Length of array: 200
Time taken: 0.054 ms
*****
Iteration 3
Length of array: 300
Time taken: 0.084 ms
*****
Iteration 4
Length of array: 400
Time taken: 0.114 ms
*****
Iteration 5
Length of array: 500
Time taken: 0.17 ms
*****
Iteration 6
Length of array: 600
Time taken: 0.185 ms
*****
Iteration 7
Length of array: 700
Time taken: 0.211 ms
*****
Iteration 8
Length of array: 800
Time taken: 0.251 ms
*****
Iteration 9
Length of array: 900
Time taken: 0.3 ms
*****
Iteration 10
Length of array: 1000
Time taken: 0.392 ms
```

Graph:

