**LAB 2**

**Task 1**

**Quick Sorting Algorithm**

**1. #include <iostream>**

**2. #include <vector>**

**3. using namespace std;**

**4.**

**5. int partition(vector<int>& arr,int low , int high,int& count,int& loop){**

**6.     int pivot = arr[high];**

**7.     int i=low-1;**

**8.**

**9.     for (int j=low ; j <= high-1;j++){**

**10.         if(arr[j]<pivot){**

**11.             i++;**

**12.             swap(arr[i],arr[j]);**

**13.             count ++;**

**14.         }**

**15.         loop++;**

**16.     }**

**17.**

**18.     swap(arr[i+1],arr[high]);**

**19.     return i+1;**

**20. }**

**21.**

**22. void quicksort(vector<int>& arr,int low , int high,int& count,int& loop){**

**23.     if (low<high){**

**24.         int pi = partition(arr,low,high,count,loop);**

**25.         quicksort(arr,low,pi-1,count,loop);**

**26.         quicksort(arr,pi+1,high,count,loop);**

**27.     }**

**28. }**

**29.**

**30. int main (){**

**31.     vector<int> arr = {10, 7, 8, 9, 1, 5, 2, 6, 3, 4};**

**32.     int n = arr.size();**

**33.     int count = 0;**

**34.     int loop = 0;**

**35.     quicksort(arr,0,n-1,count,loop);**

**36.     cout<<"Number of Swaps: "<<count<<endl;**

**37.     cout<<"Number of loops: "<<loop<<endl;**

**38.     cout<<"Sorted array: ";**

**39.     for (int i=0;i<n;i++){**

**40.         cout<<arr[i]<<" ";**

**41.     }**

**42.**

|  |  |  |
| --- | --- | --- |
|  | **LOOPS** | **SWAPS** |
| **BEST** | **45** | **45** |
| **WORSE** | **45** | **20** |
| **AVG** | **22** | **14** |

A computer screen shot of a black background

Description automatically generated

**PIVOT ELEMENT AS FIRST**

**1. #include <iostream>**

**2. #include <vector>**

**3. using namespace std;**

**4.**

**5. int partition(vector<int>& arr, int low, int high, int& count, int& loop) {**

**6.     int pivot = arr[low];**

**7.     int i = low + 1;**

**8.**

**9.     for (int j = low + 1; j <= high; j++) {**

**10.         if (arr[j] < pivot) {**

**11.             swap(arr[i], arr[j]);**

**12.             i++;**

**13.             count++;**

**14.         }**

**15.         loop++;**

**16.     }**

**17.     cout<<pivot<<endl;**

**18.     swap(arr[low], arr[i - 1]);**

**19.     return i - 1;**

**20. }**

**21.**

**22. void quicksort(vector<int>& arr, int low, int high, int& count, int& loop) {**

**23.     if (low < high) {**

**24.         int pi = partition(arr, low, high, count, loop);**

**25.         quicksort(arr, low, pi - 1, count, loop);**

**26.         quicksort(arr, pi + 1, high, count, loop);**

**27.     }**

**28. }**

**29.**

**30. int main() {**

**31.     vector<int> arr = {10,40,80,70,50};**

**32.     int n = arr.size();**

**33.     int count = 0;**

**34.     int loop = 0;**

**35.     quicksort(arr, 0, n - 1, count, loop);**

**36.**

**37.     cout << "Number of Swaps: " << count << endl;**

**38.     cout << "Number of loops: " << loop << endl;**

**39.     cout << "Sorted array: ";**

**40.     for (int i = 0; i < n; i++) {**

**41.         cout << arr[i] << " ";**

**42.     }**

**43. }**

**44.**

|  |  |  |
| --- | --- | --- |
|  | **LOOPS** | **SWAPS** |
| **BEST** | **45** | **45** |
| **WORSE** | **45** | **20** |
| **AVG** | **22** | **14** |

**A screen shot of a computer

Description automatically generated**

**TASK 2**

**Decimal to Binary (Iterative)**

**1. #include <iostream>**

**2. using namespace std;**

**3.**

**4. int main() {**

**5.     int decimal, binary = 0, place = 1;**

**6.**

**7.     cout << "Enter a decimal number: ";**

**8.     cin >> decimal;**

**9.**

**10.     int originalDecimal = decimal;**

**11.**

**12.     while (decimal > 0) {**

**13.         int remainder = decimal % 2;**

**14.         binary += remainder \* place;**

**15.         place \*= 10;**

**16.         decimal /= 2;**

**17.     }**

**18.**

**19.     cout << "The binary of " << originalDecimal << " is: " << binary << endl;**

**20.**

**21.     return 0;**

**22. }**

**23.**

**A screenshot of a computer

Description automatically generated**

**Decimal to Binary Recursive**

**1. #include <iostream>**

**2. using namespace std;**

**3.**

**4. void decimalToBinary(int decimal) {**

**5.     if (decimal == 0) {**

**6.         return;**

**7.     }**

**8.     decimalToBinary(decimal / 2);**

**9.     cout << decimal % 2;**

**10. }**

**11.**

**12. int main() {**

**13.     int decimal;**

**14.     cout << "Enter a decimal number: ";**

**15.     cin >> decimal;**

**16.**

**17.     if (decimal == 0) {**

**18.         cout << "The binary of 0 is: 0" << endl;**

**19.     } else {**

**20.         cout << "The binary of " << decimal << " is: ";**

**21.         decimalToBinary(decimal);**

**22.         cout << endl;**

**23.     }**

**24.**

**25.     return 0;**

**LEETCODE PROBLEM**

**Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return *the number of islands*.**

**An island is surrounded by water and is formed by connecting adjacent lands horizontally or v  class Solution {**

1**.  public:**

**2.   int numIslands(vector<vector<char>>& grid) {**

**3.     constexpr int dirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};**

**4.     const int m = grid.size();**

**5.     const int n = grid[0].size();**

**6.     int ans = 0;**

**7.**

**8.     auto bfs = [&](int r, int c) {**

**9.       queue<pair<int, int>> q{{{r, c}}};**

**10.       grid[r][c] = '2';  // Mark '2' as visited.**

**11.       while (!q.empty()) {**

**12.         const auto [i, j] = q.front();**

**13.         q.pop();**

**14.         for (const auto& [dx, dy] : dirs) {**

**15.           const int x = i + dx;**

**16.           const int y = j + dy;**

**17.           if (x < 0 || x == m || y < 0 || y == n)**

**18.             continue;**

**19.           if (grid[x][y] != '1')**

**20.             continue;**

**21.           q.emplace(x, y);**

**22.           grid[x][y] = '2';  // Mark '2' as visited.**

**23.         }**

**24.       }**

**25.     };**

**26.**

**27.     for (int i = 0; i < m; ++i)**

**28.       for (int j = 0; j < n; ++j)**

**29.         if (grid[i][j] == '1') {**

**30.           bfs(i, j);**

**31.           ++ans;**

**32.         }**

**33.**

**34.     return ans;**

**35.   }**

**36. };**

**37.**

**38.**

**39**

40.