Nguyễn Trần Ánh Linh-47.01.104.123

**Bài thực hành 4: Cài đặt quicksort**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| arr[] = { 9,4,8,3,7,1,6,2,5 } | 1 2 3 4 5 6 7 8 9 |
| arr[] = { 1,2,6,10,15,20,1,5 } | 1 1 2 5 6 10 15 20 |
|  |  |

Cách xử lý:

Bước 1: Một mảng các số không có thứ tự.

Bước 2: Một số được chọn làm trục.

Bước 3: Các số nhỏ hơn trục di chuyển sang bên trái của trục.

Bước 4: Các số lớn hơn trục di chuyển sang bên phải của trục.

Bước 5: Mảng được chia thành hai – nữa đầu tiên sẽ chứa các phần tử ở bên trái của trục trong khi nữa thứ hai sẽ chứa các phần tử ở bên phải.

Bước 6: lặp lại b2->b5 cho từng bên cho đến khi mọi phần tử được sắp xếp theo thứ tự tăng dần.

**Source code:**

#include <iostream>

using namespace std;

// Swap function to swap two elements

void Myswap(int& x, int& y)

{

int temp = x;

x = y;

y = temp;

}

// Partition function to divide the array into two parts

int partition(int arr[], int start, int end)

{

int pivot = arr[end]; // Choose the last element as the pivot

int x = start - 1; // Choose x as the index of the smaller element

// Iterate through the array from start to end-1

for (int y = start; y <= end - 1; y++) {

// Check if the current element is smaller than the pivot

if (arr[y] < pivot) {

x++; // Increment x

Myswap(arr[x], arr[y]); // Swap arr[x] and arr[y]

}

}

// Swap the pivot with arr[x+1]

Myswap(arr[x + 1], arr[end]);

return (x + 1); // Return the index of the pivot after placing it in the correct position

}

// Quicksort function

void quickSort(int arr[], int start, int end)

{

if (start < end) {

int array\_partition = partition(arr, start, end); // Partition the array and store the pivot index

quickSort(arr, start, array\_partition - 1); // Sort the elements smaller than the pivot

quickSort(arr, array\_partition + 1, end); // Sort the elements greater than the pivot

}

}

// Print the array

void printArray(int arr[], int size)

{

for (int i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

int main()

{

int arr[] = { 1, 2, 6, 10, 15, 20, 1, 5 };

int size = sizeof(arr) / sizeof(arr[0]); // Get the length of the array

quickSort(arr, 0, size - 1); // Sort the array

cout << "Mang sau khi sap xep: ";

printArray(arr, size); // Print the sorted array

return 0;

}

**Bài thực hành 6: Tháp Hà Nội**

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| N = 4 | A -> B  A -> C  B -> C  A -> B  C -> A  C -> B  A -> B  A -> C  B -> C  B -> A  C -> A  B -> C  A -> B  A -> C  B -> C |
| N = 1 | A -> C |
| N = 3 | A -> C  A -> B  C -> B  A -> C  B -> A  B -> C  A -> C |

Cách xử lý: xử dụng đệ quy

Nếu N = 1 thì chuyển A- > C

Nếu N > 1 thì:

Dịch chuyển n-1 đĩa từ A -> B

Dịch chuyển đĩa thứ n từ A -> C

Dịch chuyển n-1 đĩa từ B -> C

Đến khi còn 1 đĩa thì hết đệ quy và chuyển trực tiếp từ A -> C (Kết thúc thuật toán)

**Source code:**

#include <iostream>

using namespace std;

void towerHN(int n, char source, char auxiliary, char destination)

{

if (n == 1) {

cout << "Move disk 1 from " << source << " to " << destination << endl;

return;

}

towerHN(n - 1, source, destination, auxiliary);

cout << "Move disk " << n << " from " << source << " to " << destination << endl;

towerHN(n - 1, auxiliary, source, destination);

}

int main() {

int n;

do {

cout << "Nhap vao so dia: ";

cin >> n;

if (n < 0) {

cout << "Nhap so dia > 0: ";

cin >> n;

}

} while (n < 0);

cout << "Thu tu dich chuyen la: " << endl;

towerHN(n, 'A', 'B', 'C');

return 0;

}