**Thực hành: Cấu trúc dữ liệu và giải thuật buổi 2**

**Họ tên**: Nguyễn Hoàng Thắng

**Mssv**: N21DCVT097

**Bài 1**: Viết chương trình thực hiện thuật toán Quick-Sort  
để sắp xếp một mảng số nguyên nhập từ người dùng. Sau đó, in ra mảng đã sắp xếp.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

void xuatMang(int a[], int n){

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

cout<<a[i]<<" ";

}

}

void swap(int \*a, int \*b)

{

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int arr[], int l, int h)

{

int x = arr[h];

int i = (l - 1);

for (int j = l; j <= h - 1; j++)

{

if (arr[j] <= x)

{

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[h]);

return (i + 1);

}

void quickSort(int arr[], int l, int h)

{

if (l < h)

{

int p = partition(arr, l, h); // tìm vị trí của chốt

quickSort(arr, l, p - 1); // trị nửa bên trái

quickSort(arr, p + 1, h); // trị nửa bên phải

}

}

int main()

{

int a[100], n;

nhapMang(a, n);

quickSort(a, 0, n-1);

xuatMang(a, n);

}

**Bài 2**:  Xây dựng hàm thực hiện thuật toán Merge-Sort để sắp  
xếp một mảng số nguyên nhập từ người dùng. Sau đó, in ra mảng đã sắp xếp.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

void xuatMang(int a[], int n)

{

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

{

cout << a[i] << " ";

}

}

void merge(int arr[], int l, int m, int r)

{

int i, j, k, n1 = m - l + 1, n2 = r - m;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

L[i] = arr[l + i];

for (j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

i = 0;

j = 0;

k = l; // bắt đầu hợp nhất

while (i < n1 && j < n2)

{ // quá trình hợp nhất

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int l, int r)

{

if (l < r)

{

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

int main()

{

int a[100], n;

nhapMang(a, n);

mergeSort(a, 0, n - 1);

xuatMang(a, n);

}

**Bài 3**:  Tạo một cấu trúc heap từ một mảng số nguyên nhập  
từ người dùng bằng cách sử dụng thuật toán Heapify. Sau đó, sắp xếp mảng đó bằng cách lấy phần tử lớn nhất ra khỏi heap và in ra mảng đã sắp xếp.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

void xuatMang(int a[], int n)

{

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

{

cout << a[i] << " ";

}

}

void heapify(int arr[], int n, int i)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i)

{

swap(arr[i], arr[largest]);

heapify(arr, n, largest);

}

}

void heapSort(int arr[], int n)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

}

int main()

{

int a[100], n;

nhapMang(a, n);

heapSort(a, n);

xuatMang(a, n);

}

**Bài 4**:  Viết một chương trình sử dụng thuật toán Shell-Sort để sắp xếp một mảng số nguyên nhập từ người dùng. In ra mảng đã sắp xếp sau mỗi bước lặp của thuật toán để hiểu rõ cách hoạt động của Shell-Sort.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

void xuatMang(int a[], int n)

{

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

{

cout << a[i] << " ";

}

}

int shellSort(int arr[], int n)

{

for (int gap = n / 2; gap > 0; gap /= 2)

{

for (int i = gap; i < n; i += 1)

{

int temp = arr[i];

int j;

for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)

arr[j] = arr[j - gap];

arr[j] = temp;

}

}

return 0;

}

int main()

{

int a[100], n;

nhapMang(a, n);

shellSort(a, n);

xuatMang(a, n);

}

**Bài 5**:  Viết một chương trình thực hiện thuật toán Radix-Sort để sắp xếp một mảng số nguyên nhập từ người dùng. Hãy sử dụng phương pháp radix sort theo cơ số 10 và in ra mảng đã sắp xếp.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

void xuatMang(int a[], int n)

{

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

{

cout << a[i] << " ";

}

}

int maxArr(int arr[], int n)

{

int mx = arr[0];

for (int i = 1; i < n; i++)

if (arr[i] > mx)

mx = arr[i];

return mx;

}

void countSort(int arr[], int n, int exp)

{

int output[n];

int i, count[10] = {0};

for (i = 0; i < n; i++)

count[(arr[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = n - 1; i >= 0; i--)

{

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

for (i = 0; i < n; i++)

arr[i] = output[i];

}

void radixsort(int a[], int n)

{

int m = maxArr(a, n);

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(a, n, exp);

}

int main()

{

int a[100], n;

nhapMang(a, n);

radixsort(a, n);

xuatMang(a, n);

}

**Bài 6**:  Viết một chương trình thực hiện tìm kiếm tuyến tính để tìm một số nguyên cụ thể trong một mảng số nguyên nhập từ người dùng. In ra vị trí của số đó trong mảng nếu nó tồn tại, nếu không, thông báo rằng số đó không tồn tại trong mảng.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

int linearSearch(int a[], int n, int x)

{

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

{

if (a[i] == x)

return i;

}

return -1;

}

int main()

{

int a[100], n;

nhapMang(a, n);

int result = linearSearch(a, n, 5);

if (result != -1)

{

cout << "da tim ra o vi tri " << result << endl;

}

else

{

cout << "khong tim thay" << endl;

}

}

**Bài 7**:  Viết một chương trình thực hiện tìm kiếm nhị phân để tìm một số nguyên cụ thể trong một mảng số nguyên đã được sắp xếp. In ra vị trí của số đó trong mảng nếu nó tồn tại, nếu không, thông báo rằng số đó không tồn tại trong mảng.

#include <bits/stdc++.h>

using namespace std;

void nhapMang(int a[], int &n)

{

cout << "nhap n: ";

cin >> n;

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

{

cin >> a[i];

}

}

int binarySearch(int a[], int n, int x){

int left = 0, right=n-1, mid=(left+right)/2;

while (left<=right)

{

if(x<a[mid]) right=mid-1;

else if(x>a[mid]) left=mid+1;

else return mid;

mid = (left+right)/2;

}

return -1;

}

int main(){

int a[100], n;

nhapMang(a, n);

int result = binarySearch(a, n, 5);

if (result != -1)

{

cout << "da tim ra o vi tri " << result << endl;

}

else

{

cout << "khong tim thay" << endl;

}

}