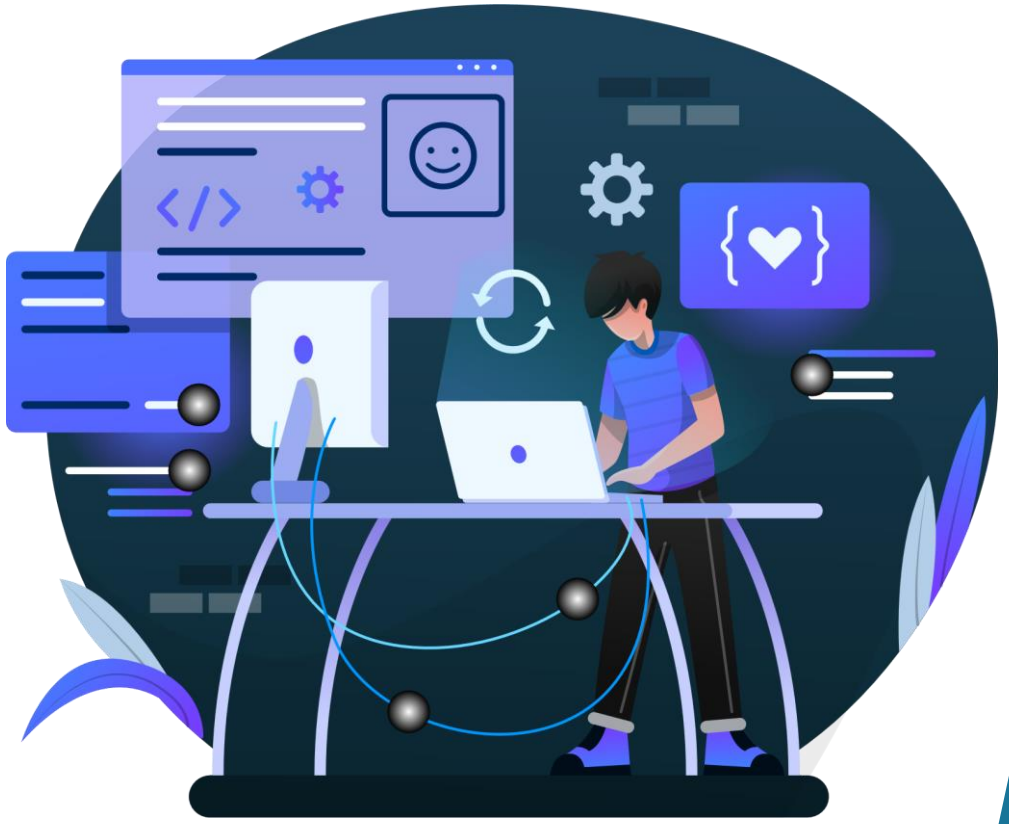


THUẬT TOÁN SẮP XẾP

Thuật toán sắp xếp:

Sắp xếp là một thuật toán quan trọng, được sử dụng cực kì nhiều trong các ứng dụng thực tế. Các bạn có thể không biết các thuật toán như quy hoạch động, chia và trị nhưng bắt buộc phải nắm được sắp xếp và tìm kiếm.

1.Thuật toán sắp xếp chọn (Selection sort):

```
void selectionSort(int a[], int n){
    for(int i = 0; i < n; i++){
        int min_pos = i;
        for (int j = i + 1; j < n; j++){
            if (a[j] < a[min_pos]){
                min_pos = j;
            }
        }
        swap(a[min_pos], a[i]);
    }
}
```

Độ phức tạp: O(N^2)

2.Thuật toán sắp xếp nổi bọt (Bubble sort):

```
void bubbleSort(int a[], int n){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n - i - 1; j++){
            if (a[j] > a[j + 1]){
                swap(a[j], a[j + 1]);
            }
        }
    }
}
```

Độ phức tạp: O(N^2)

3.Thuật toán sắp xếp chèn (Insertsion sort):

```
void insertionSort(int a[], int n){
    for(int i = 1; i < n; i++){
        int pos = i - 1, x = a[i];
        while(pos >= 0 && a[pos] > x){
            a[pos + 1] = a[pos];
            --pos;
        }
        a[pos + 1] = x;
    }
}
```

Độ phức tạp: O(N^2)

4. Thuật toán sắp xếp đếm phân phối (Counting sort):

Điều kiện áp dụng: Có thể khai báo được mảng đếm có số lượng phần tử lớn hơn giá trị lớn nhất của phần tử trong mảng

```
int dem[1000001]; // 0 <= a[i] <= 10^6
void countingSort(int a[], int n){
    int K = -1e9;
    for(int i = 0; i < n; i++){
        dem[a[i]]++;
        K = max(K, a[i]);
    }
    for(int i = 0; i <= K; i++){
        if(dem[i]){
            for(int j = 0; j < dem[i]; j++){
                cout << i << ' ';
            }
        }
    }
}
```

Độ phức tạp: O(N^2)

5. Thuật toán sắp xếp trộn (Merge sort):

Độ phức tạp: O(NLogN)

Thao tác trộn:

```
void merge(int a[], int l, int m, int r){
    int n1 = m - l + 1, n2 = r - m;
    int x[n1], y[n2];
    for(int j = l; j <= m; j++)
        x[j - l] = a[j];
    for(int j = m + 1; j <= r; j++)
        y[j - m - 1] = a[j];
    int i = 0, j = 0, cnt = l;
    while(i < n1 && j < n2){
        if(x[i] <= y[j])
            a[cnt++] = x[i++];
        else
            a[cnt++] = y[j++];
    }
    while(i < n1) a[cnt++] = x[i++];
    while(j < n2) a[cnt++] = y[j++];
}
```

Hàm merge soft và main:

```
void mergeSort(int a[], int l, int r){
    if(l < r){
        int m = (l + r) / 2;
        mergeSort(a, l, m);
        mergeSort(a, m + 1, r);
        merge(a, l, m , r);
    }
}
int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    mergeSort(a, 0, n - 1);
    for(int x : a) cout << x << ' ';
}
```

6. Thuật toán quick sort:

Độ phức tạp: O(NLogN)

Thao tác phân hoạch bằng Lomuto partition:

```
int partition(int a[], int l, int r){
    int pivot = a[r];
    int i = l - 1;
    for(int j = l; j < r; j++){
        if(a[j] <= pivot){
            ++i;
            swap(a[i], a[j]);
        }
    }
    ++i;
    swap(a[i], a[r]);
    return i;
}
```

Hàm quick soft và main:

```
void quicksort(int a[], int l, int r){
    if(l < r){
        int m = (l + r) / 2;
        int p = partition(a, l, r);
        quicksort(a, l, p - 1);
        quicksort(a, p + 1, r);
    }
}

int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    quicksort(a, 0, n - 1);
    for(int x : a) cout << x << ' ';
}
```

7. Hàm soft trong thư viện STL:

Thư viện STL cung cấp 2 hàm sort

sort

Hàm sort được cài đặt bằng Intro sort (kết hợp của quick sort và heap sort)

stable_sort

Hàm stable_sort được cài đặt bằng thuật toán merge sort nên có thêm tính chất stable

STL

7. Hàm soft trong thư viện STL:

CÚ PHÁP

Cú pháp áp dụng hàm sort với toàn bộ mảng, mặc định sẽ được sắp xếp theo thứ tự tăng dần về giá trị số và tăng dần về thứ tự từ điển về giá trị kí tự

```
//Sort mảng a có n phần tử
sort (a, a + n);
```

Sort theo thứ tự giảm dần, ta thêm tham số greater vào hàm sort, chú ý tới kiểu dữ liệu của mảng, trong trường hợp này mảng a là mảng int

```
sort(a, a + n, greater<int>());
```

7. Hàm soft trong thư viện STL:

Sort mảng a trong đoạn từ chỉ số x tới chỉ số y

```
sort(a + x, a + y + 1);
sort(a + x, a + y + 1, greater<int>());
```

Sort toàn bộ vector v

```
sort(v.begin(), v.end());
sort(v.begin(), v.end(), greater<int>());
```

Sort vector từ chỉ số x tới chỉ số y

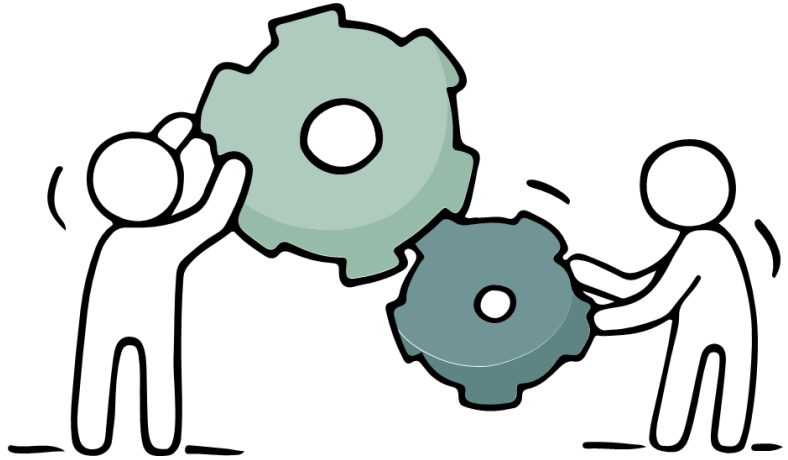
```
sort(v.begin() + x, v.begin() + y);
sort(v.begin() + x, v.begin() + y, greater<int>());
```

Đối với stable_sort các bạn áp dụng tương tự

8. Xây dựng hàm comparator cho hàm sort:

Nếu các bạn chỉ có thể sử dụng hàm sort để sắp xếp tăng dần và giảm dần thì sẽ không áp dụng nó vào các bài toán thực tế được.

Hàm comparator do người dùng tự xây dựng để thực hiện yêu cầu sắp xếp riêng theo từng bài toán thực tế.



Sắp xếp mảng tăng dần

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

bool cmp(int a, int b){
    if (a < b)
        return true;
    else
        return false;
    //return a < b;
}
```

Hàm main:

```
int main(){
    int n; cin >> n;
    int a[1000];
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    sort(a, a + n, cmp);
    for(int i = 0; i < n; i++){
        cout << a[i] << ' ';
    }
}
```

Sắp xếp mảng giảm dần

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

bool cmp(int a, int b){
    if (a > b)
        return true;
    else
        return false;
    //return a > b;
}
```

Hàm main:

```
int main(){
    int n; cin >> n;
    int a[1000];
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    sort(a, a + n, cmp);
    for(int i = 0; i < n; i++){
        cout << a[i] << ' ';
    }
}
```

Sắp xếp theo trị tuyệt đối tăng dần

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

bool cmp(int a, int b){
    if (abs(a) < abs(b))
        return true;
    else
        return false;
    //return abs(a) < abs(b);
}
```

Hàm main:

```
int main(){
    int n = 5;
    int a[5] = {-9, 2, 4, -3, 1};
    sort(a, a + n, cmp);
    for(int x : a)
        cout << x << ' ';
}
```

OUTPUT: 1 2 -3 4 -9

Sắp xếp theo tổng chữ số tăng dần

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

int sum(int n){
    int res = 0;
    while(n){
        res += n % 10; n /= 10;
    }
    return res;
}
bool cmp(int a, int b){
    if(sum(a) < sum(b))
        return true;
    else return false;
}
```

Hàm main:

```
int main(){
    int n = 5;
    int a[5] = {22, 1, 4, 9, 33};
    sort(a, a + n, cmp);
    for(int x : a)
        cout << x << ' ';
}
```

OUTPUT: 1 22 4 33 9

Sắp xếp theo tổng chữ số tăng dần, nếu 2 số có cùng tổng chữ số thì số nhỏ hơn sẽ xếp trước

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

int sum(int n){
    int res = 0;
    while(n){
        res += n % 10; n /= 10;
    }
    return res;
}
bool cmp(int a, int b){
    if(sum(a) != sum(b))
        return sum(a) < sum(b);
    else return a < b;
}
```

Hàm main:

```
int main(){
    int n = 5;
    int a[5] = {22, 6, 4, 9, 33};
    sort(a, a + n, cmp);
    for(int x : a)
        cout << x << ' ';
}
```

OUTPUT: 4 22 6 33 9

Sắp xếp theo tổng chữ số tăng dần, nếu 2 số có cùng tổng chữ số thì số nào được nhập trước sẽ xếp trước (Dùng stable_sort)

Hàm comparator

```
#include <bits/stdc++.h>
using namespace std;

int sum(int n){
    int res = 0;
    while(n){
        res += n % 10; n /= 10;
    }
    return res;
}
bool cmp(int a, int b){
    return sum(a) < sum(b);
}
```

Hàm main:

```
int main(){
    int n = 5;
    int a[5] = {22, 6, 4, 9, 33};
    stable_sort(a, a + n, cmp);
    for(int x : a)
        cout << x << ' ';
}
```

OUTPUT: 22 4 6 33 9