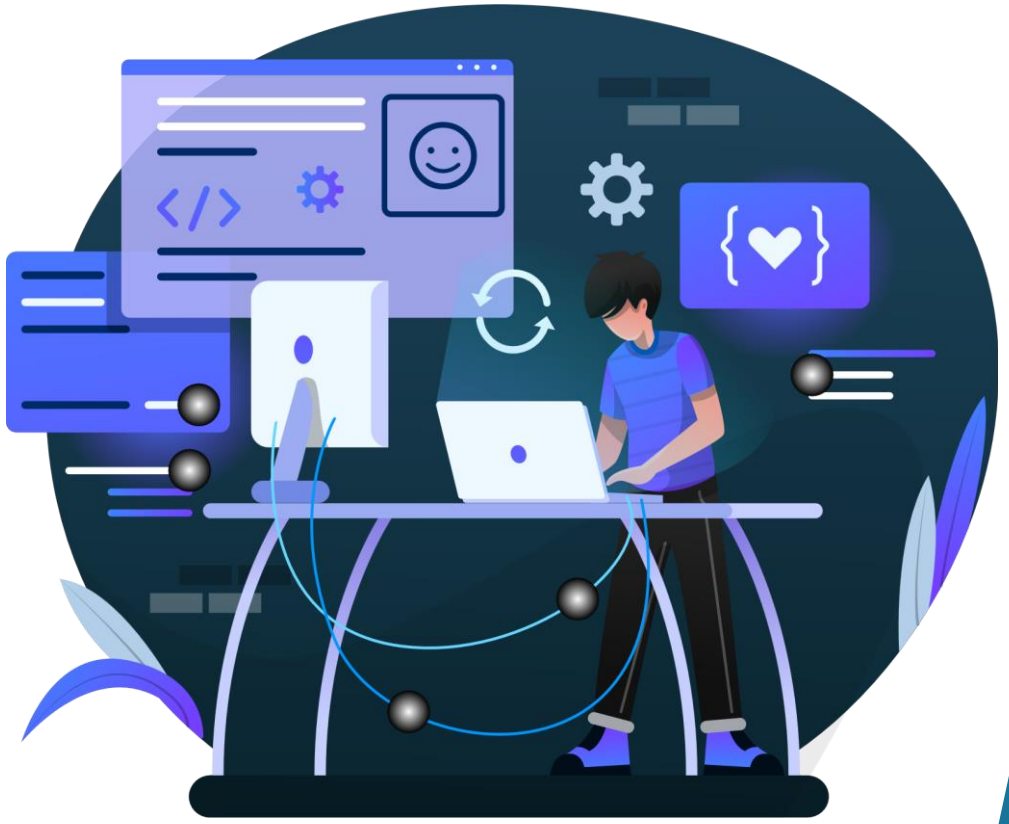

THUẬT TOÁN SẮP XẾP

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 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 << ' ';
}
```