Sắp xếp

1. Các thuật toán sắp xếp

* Selection Sort (sắp xếp chọn) x
* Interchange Sort (sắp xếp đổi chỗ trực tiếp)x
* Bubble Sort (sắp xếp nổi bọt)x
* Shaker Sort (sắp xếp cocktail)
* Insertion Sort (sắp xếp chèn)x
* Shell Sort (sắp xếp vỏ)
* Counting Sort ( sắp xếp bằng phép đếm phân phối)
* Bucket Sort (sắp xếp cái xô)
* Radix Sort (sắp xếp cơ số)
* Heap Sort (sắp xếp vun đống)
* Merge Sort (sắp xếp trộn)x
* Quick Sort (sắp xếp nhanh)x

1. Tiêu chí:

* Cài đặt/triển khai
* Độ phức tạp
* Bộ nhớ
* Tính ổn định

1. Thuật toán Selection Sort (Sắp xếp chọn) – chọn phần tử bé nhất từ vị trí i chèn vào vị trí i

Ý tưởng: duyệt 2 vòng lặp vòng lặp thứ 2 tìm phần tử bé/lớn nhất bắt đầu từ vị trí tiếp của vị trí i

Triển khai:

void SelectionSort(int\* a, int n){

    for(int i=0; i<n-1; i++){ // duyệt từ đầu đến cuối

        int min = i;

        for(int j=i+1; j<n; j++) // tìm phần tử min

            if(a[min] > a[j])

                min = j; // lưu vị trí bé nhất vào vị trí min

        if(a[i] > a[min]){

            int t = a[i];

            a[i] = a[min];

            a[min] = t;

        }

}

Vòng lặp thứ nhất lặp 1 ở vị trí i thì vòng lặp thứ 2 lặp (n-i) lần

Vòng lặp thứ nhất lặp (n-1) lần => O(f(n)) = n + (n-1) + (n-2) + .. + 1 = n\*(n+1)/2 = O(n^2)

VD: 1 3 2 7 0

i = 0 => min = 0 => 0 3 2 7 1  
i = 1 => min = 1 => 0 1 2 7 3  
i = 2 => min = 2 => 0 1 2 7 3  
i = 3 => min = 3 => 0 1 2 3 7

VD: 10 -1 3 7 7 -5 2

1. Insertion Sort (sắp xếp chèn) – chèn vào vị trí thỏa mãn

Ý tưởng dùng 2 vòng lặp thực hiện nếu a[i] > a[i+1] ta thực hiện lưu a[i+1] vào x và dịch các phần tử sang phải cho đến khi tìm vị trí j phần tử nhỏ hơn x rồi chèn vị trí j + 1

VD: 1 3 2 7 0

i = 0 => 1 < 3 => 1 3 2 7 0  
i = 1 => 3 > 2 => x = 2, chuỗi lúc này là 1 3 3 7 0  
 Kiểm duyệt tìm đc số 1 vị trí j = 0 => chèn vào j + 1 = 1 => 1 2 3 7 0

i = 2 => 3 < 7 => 1 2 3 7 0  
i = 3 => 7 > 0 => x = 0, chuỗi thành 1 2 3 7 7  
 Duyệt rồi dịch 1 2 3 3 7   
 Tiếp 1 2 2 3 7  
 Tiếp 1 1 2 3 7  
 Không tìm thấy => 0 1 2 3 7

Triển khai:

void InsertionSort(T \*a, int n, bool tangDan = true){

   for(int i=0; i<n; i++){

        if(a[i] > a[i+1] == tangDan){

            T x = a[i+1];

            for(int j=i+1; j>=0; j--)

                if(a[j-1] > x == tangDan && j != 0)

                    a[j] = a[j-1];

                else{

                    a[j] = x;

                    break;

                }

        }

   }

}

Vòng lặp 1 thực hiện n vòng lặp => O(n)  
Vòng lặp 2 trường hợp xấu nhất thực hiện i lần

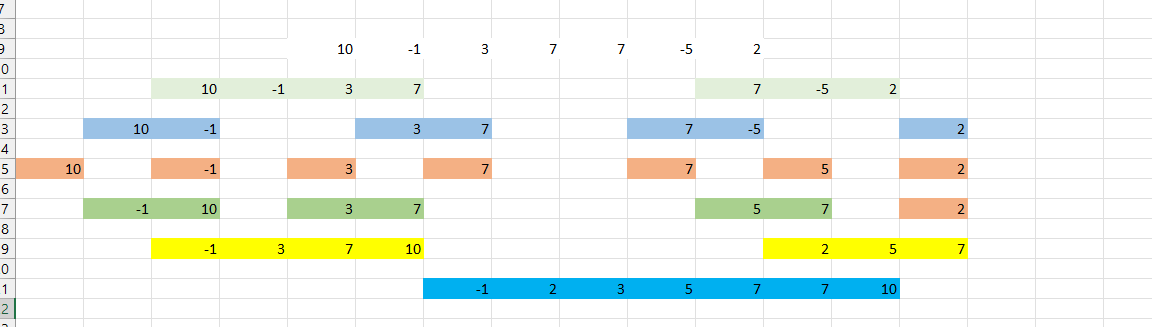
* 1 + 2 + 3 + … + n = n\*(n-1)/2 => O(n^2)

VD: 10 -1 3 7 7 -5 2

1. MegaSort:

Ý tưởng: chia dãy thành các đoạn nhỏ rồi sắp xếp từng đoạn nhỏ 1.

VD: 10 -1 3 7 7 -5 2



Code mẫu:

void Mega(int\* a, int left, int mid, int right, bool tangDan) {

int n1 = mid - left + 1;

int n2 = right - mid;

int\* a1 = new int[n1];

int\* a2 = new int[n2];

// copy day ben trai

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

a1[i] = a[left + i];

}

// copy day ben phai

for (int i = 0; i < n2; i++) {

a1[i] = a[mid + 1 + i];

}

int x1 = 0, x2 = 0;

for (int i = left; i <= right; i++) {

if ((a1[x1] < a2[x2] == tangDan && x1 < n1) || x2 == n2) {

a[i] = a1[x1];

x1++;

}

else {

a[i] = a2[x2];

x2++;

}

}

}

void MegaSort(int\* a, int left, int right, bool tangDan) {

if (left < right) {

int mid = (left + right) / 2;

MegaSort(a, left, mid, tangDan);

MegaSort(a, mid + 1, right, tangDan);

Mega(a, left, mid, right, tangDan);

}

}

1. BubbleSort (Nổi bọt)

Ý tưởng: duyệt 2 vòng for vòng for thứ 2 đảo vị trí sao cho các phần tử bé lên trên.

VD:

Ảnh có chứa bàn

Mô tả được tạo tự động

Code mẫu:

void BubbleSort(int\* a, int n, bool tangDan) {

for(int i=0; i<n-1; i++)

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

if (a[j] < a[j - 1] == tangDan) {

int t = a[j];

a[j] = a[j - 1];

a[j - 1] = t;

}

}

1. Quick – Sort (sắp xếp nhanh)

Giống vs MegaSort ta lọc thành 2 mảng sao cho 1 mảng gồm các phần tử nhỏ hơn chốt, mảng còn lại gồm các phần tử lớn hơn chốt.

Ảnh có chứa văn bản, buồng, ảnh chụp màn hình

Mô tả được tạo tự động

Code mẫu:

int Quick(int\* a, int left, int right, bool tangDan) {

int index = right;

int i = left;

for(int j=left; j<right; j++)

if (a[j] < a[index] == tangDan) {

int t = a[j];

a[j] = a[i];

a[i] = t;

i++;

}

if (a[i] > a[index] == tangDan) {

int t = a[i];

a[i] = a[index];

a[index] = t;

return i;

}

return -1;

}

void QuickSort(int\* a, int left, int right, bool tangDan) {

if (left < right) {

int index = Quick(a, left, right, tangDan);

if (index != -1) {

QuickSort(a, left, index - 1, tangDan);

QuickSort(a, index + 1, right, tangDan);

}

}

}

1. Vấn đề mở rộng

Tạo project để test

* B1: Tạo test vào file: 1000 số, 10000 số, ….
* B2: Lưu thời gian chạy

1. Các thuật toán khác

so sánh: (m+1)(n+1)

phép thông thường: 2\*m\*n