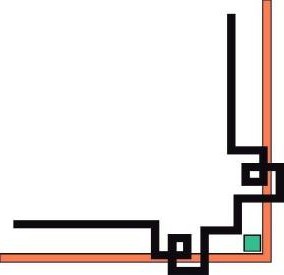
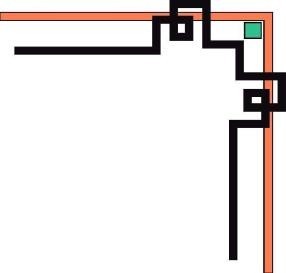
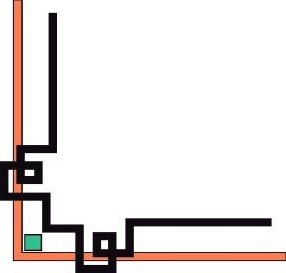
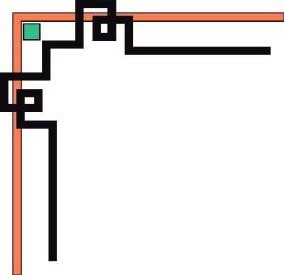
**KHOA CÔNG NGHỆ THÔNG TIN**

**MÔN HỌC: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỒ ÁN CUỐI KỲ**

**\*\*\*\*\*\***

**MÔ PHỎNG THUẬT TOÁN SẮP XẾP**





**GVHD: Huỳnh Xuân Phụng**

**Nhóm SVTH:**

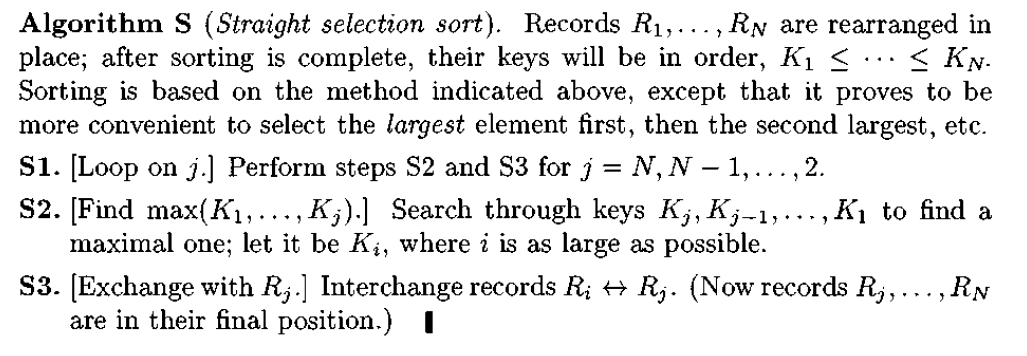
1. Đỗ Chí Hùng - 21110478
2. Nguyễn Trần Tuấn Đạt - 21110419
3. Trần Quốc Phúc -21110608

**Thành phố Hồ Chí Minh, ngày 28 tháng 12 năm 2022**

**BÁO CÁO ĐỒ ÁN**

1. **Selection sort**

**-** Yêu cầu bài toán:



- Code:

void selectionSort(int arr[], toaDo b[], int n)

{

int max;

//Di chuyen ranh gioi cua mang da sap xep va chua sap xep

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

{

int vt = j;

int temp = INT\_MIN; //am vo cuc

// Tim phan tu LON NHAT trong mang chua sap xep

for (int i= j - 1; i >= 0; i--) {

if (arr[i] > arr[j] && arr[i] > temp) {

temp = arr[i];

vt = i;

}

}

max = vt;

arr[max] = temp;

//Doi cho phan tu LON NHAT voi phan tu CUOI CUNG

MPSapXep(b, max,j, arr[max], arr[j]);

swap(arr[j], arr[max]);

}

gotoXY(50, 2);

SetColor(13);

cout << "SAP XEP HOAN TAT!";

SetColor(7);

}

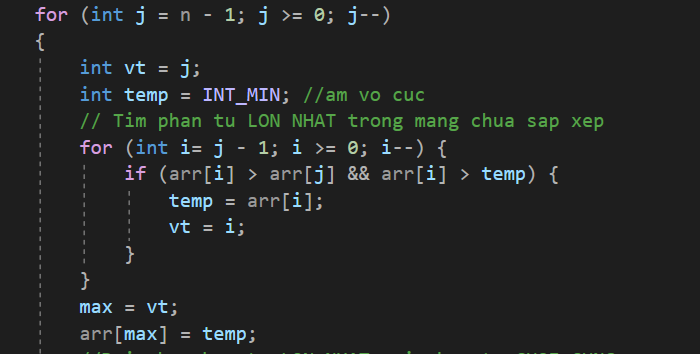
* Phân tích chương trình:



* Tạo vòng lặp j với điều kiện j = N,N-1,…2.

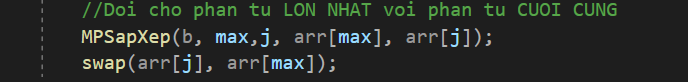


* Tạo một biến temp với giá trị ban đầu là âm vô cực làm biến chung gian để tìm được giá trị max.
* Sau khi tạo vòng lặp thì thực hiện S2 tìm giá trị max đặt là Ki



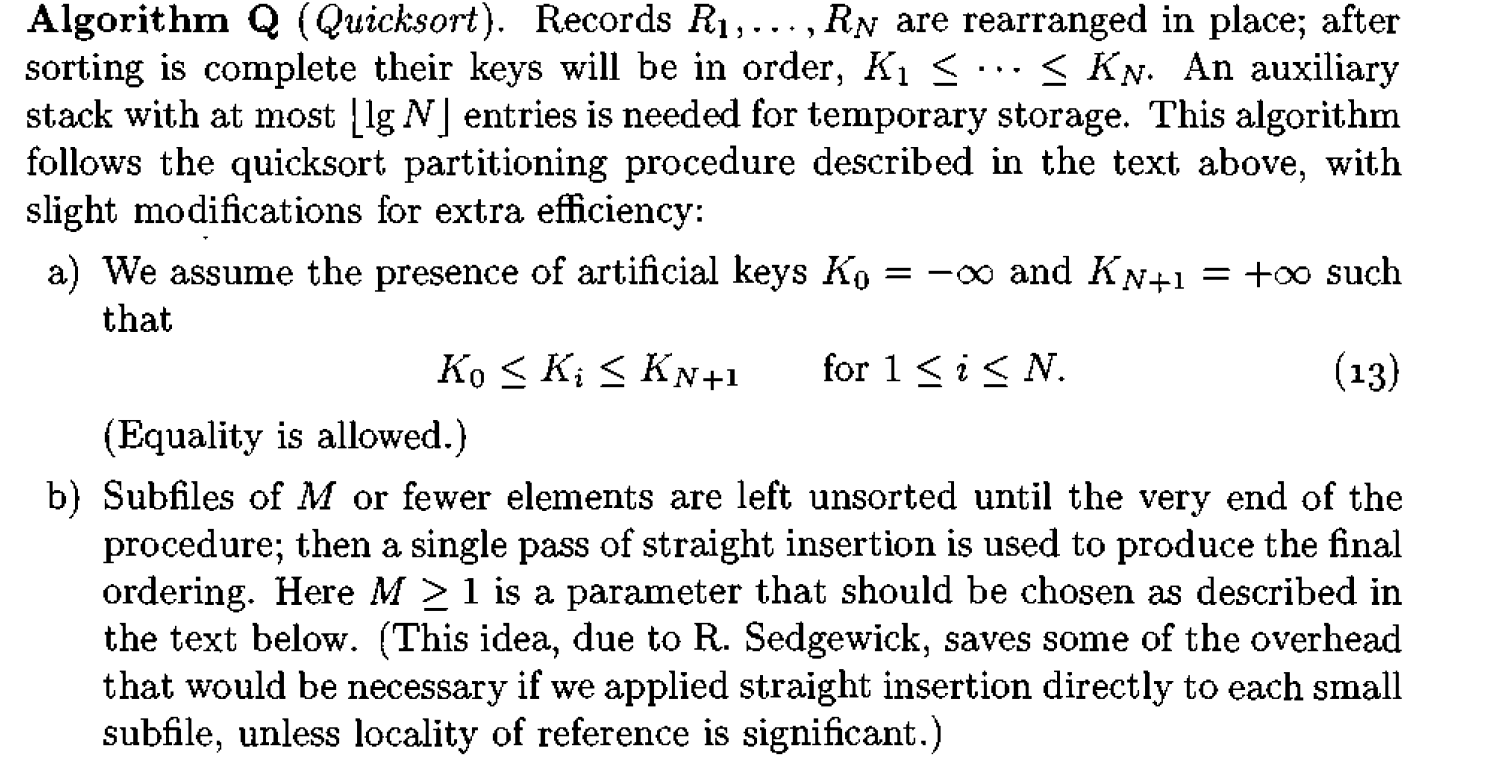


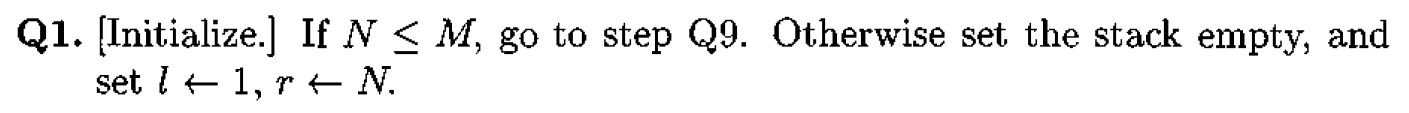
* Sau đó là thực hiện bước S3 cũng trong vòng lặp j đó là đổi chỗ Ri với Rj
* Trước khi đổi chỗ gọi hàm MPSapXep để thấy rõ được quá trình di chuyển của arr[j] và arr[max].

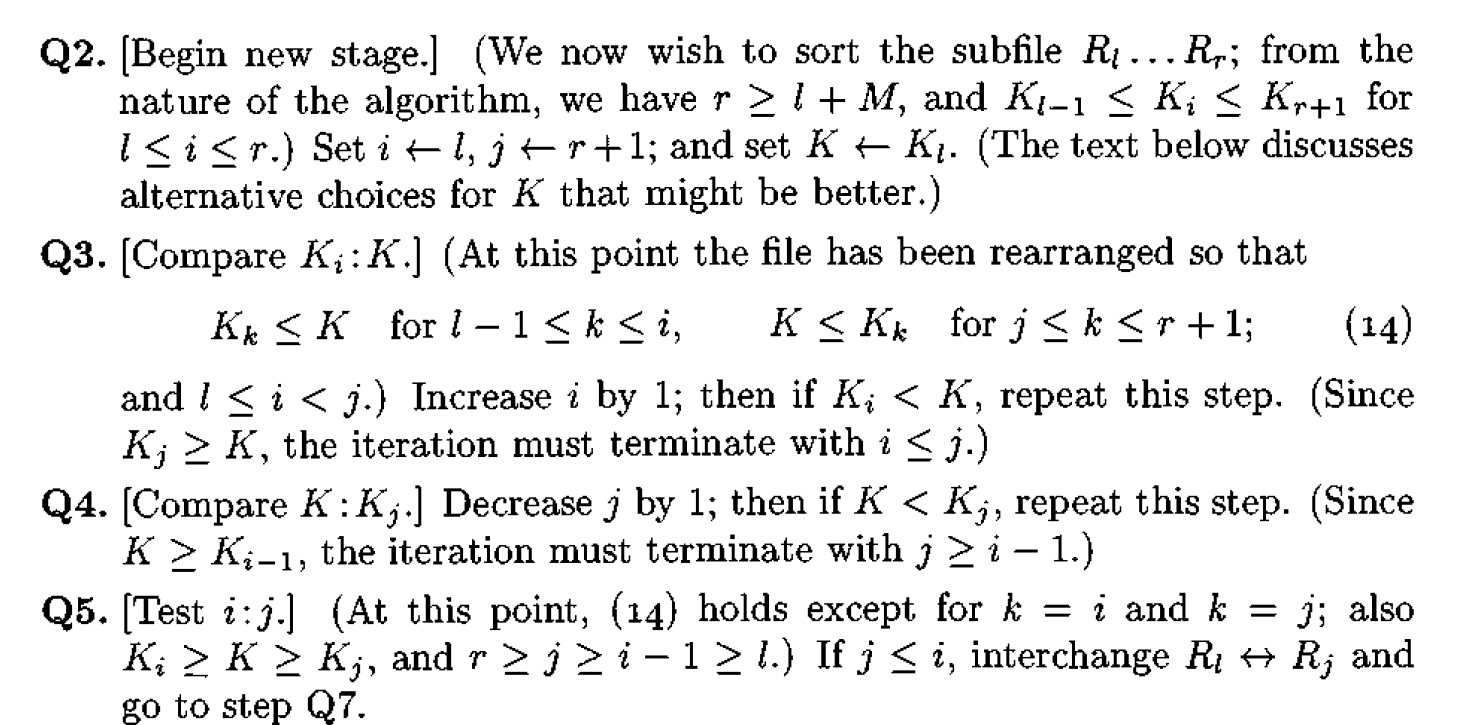


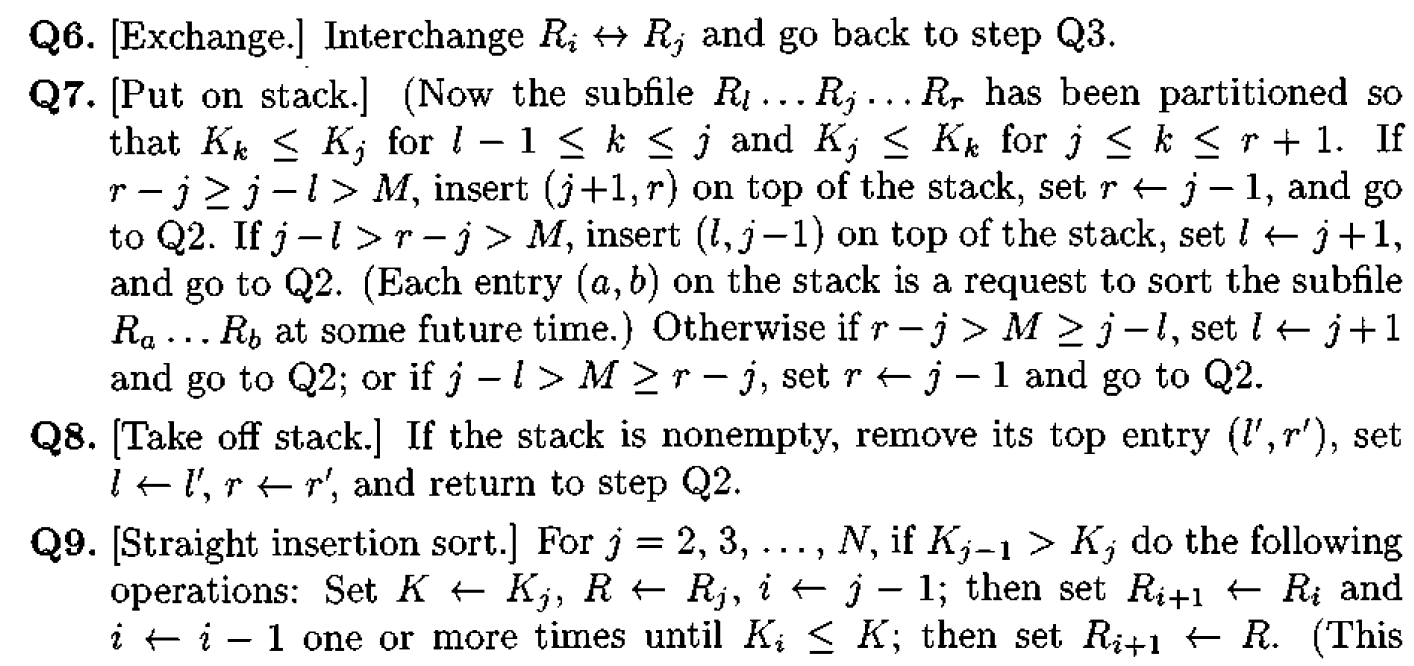
1. **Quick sort**

**-** Yêu cầu bài toán:









- Code:

void quickSort(int a[], toaDo b[], int l, int r) {

int p = a[(l + r) / 2];

int i = l, j = r;

while (i < j) {

while (a[i] < p) {

i++;

}

while (a[j] > p) {

j--;

}

if (i <= j) {

MPSapXep(b, i, j, a[i], a[j]);

int temp = a[i];

a[i] = a[j];

a[j] = temp;

i++;

j--;

}

}

if (i < r) {

quickSort(a, b, i, r);

}

if (l < j) {

quickSort(a, b, l, j);

}

gotoXY(50, 2);

SetColor(13);

cout << "SAP XEP HOAN TAT!";

}

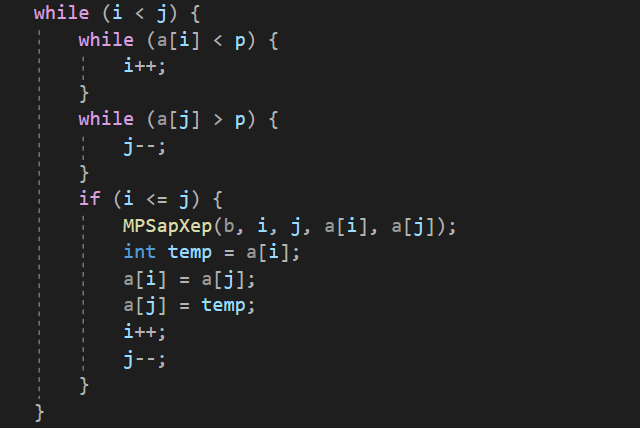
* Phân tích chương trình:
* Ý tưởng : chia nhỏ các đoạn chương trình mỗi đoạn thực hiện việc so sánh ở 2 đầu.
* Bước 1 : tạo giá trị p là giá trị ở giữa = (l+r)/2 với l và r lần lượt là 0 và n-1 tức là giá trị đầu và cuối của mảng. Đặt i=l và j=r cho 2 biến này chạy đối nhau.



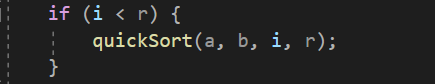
* Bước 2 dùng while với điều kiện là i < j để khi i=j thì tiếp tục chia nhỏ thành 2 phần .

+ Trong trường hợp i<j, so sánh 2 đầu với phần tử p tăng 1 cho tới khi tìm được phần tử i lớn hơn phần tử p tương tự giảm đi 1 cho tới khi tìm được phần tử j bé hơn phần tử p.

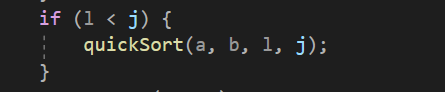
+ Sau đó sẽ hoán đổi phần tử i và j theo tính chất bắc cầu



* Bước 3 : Khi gặp trường hợp i=j nhưng mảng vẫn chưa được sắp xếp xong thì sẽ tiếp tục chia nhỏ thành 2 phần passion A có left =i và right = r tiếp tục thực hiện các bước ở bước 2.

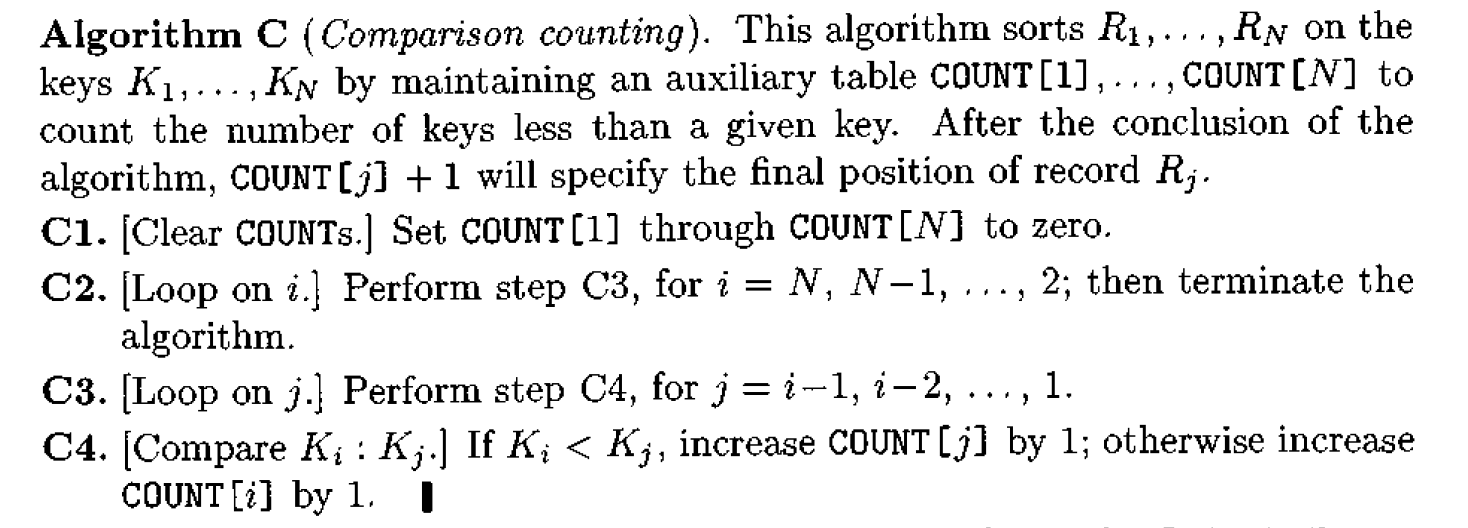


Passion B có left = l và right = j tiếp tục thực hiện các bước ở bước 2.



1. **Comparision couting sort**

**-** Yêu cầu bài toán:



- Code:

void Comparison(int arr[], toaDo b[], int n)

{

int\* count = (int\*)malloc(sizeof(int));

int i, j;

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

{

count[i] = 0;

}

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

{

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

{

if (arr[i] <= arr[j])

{

count[j] += 1;

MPSapXep(b, j, i, arr[j], arr[i]);

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

}

else

count[i] += 1;

}

}

gotoXY(50, 2);

SetColor(13);

cout << "SAP XEP HOAN TAT!";

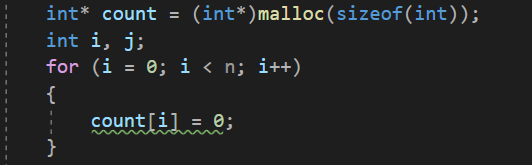
SetColor(7);

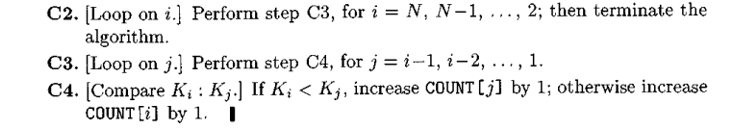
}

* Phân tích thuật toán:

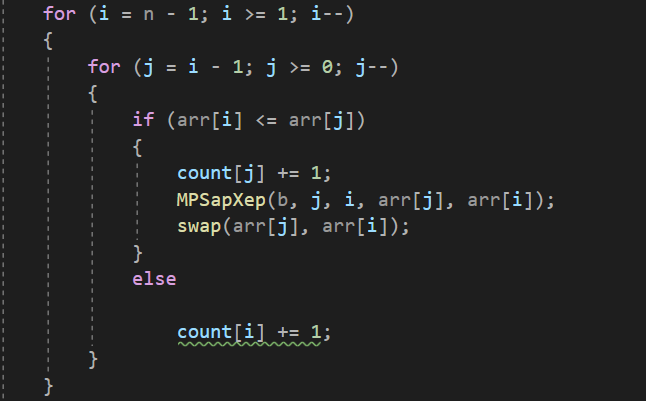


* Cấp phát một bộ nhớ COUNT xong set giá trị bằng 0 cho COUNT[1] đến COUNT[N].



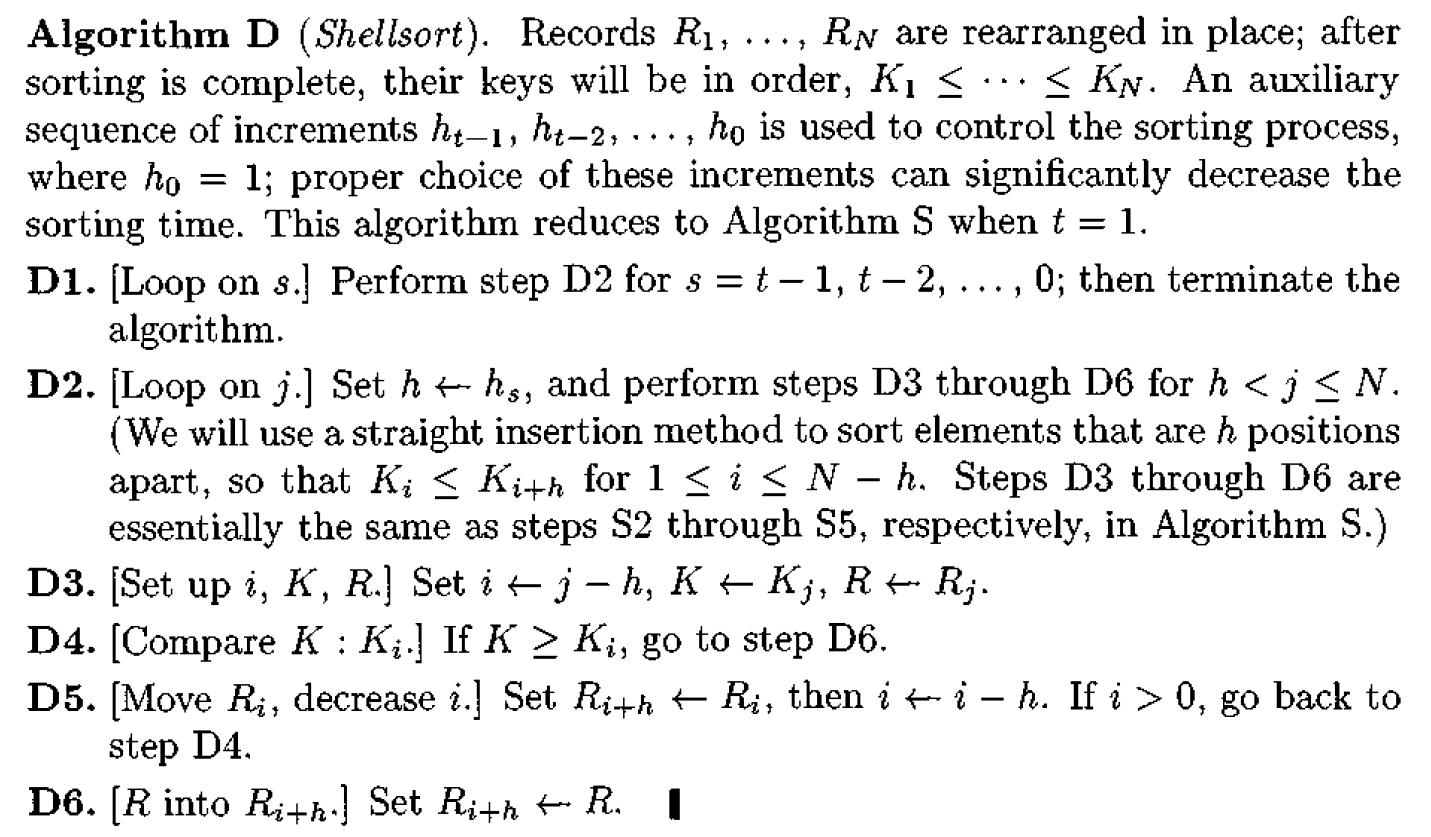


* Sử dụng vòng lặp i với i = N,N-1,…2 để thực hiện bước 3
* Tiếp tục sử dụng vòng lặp j = i – 1 , i – 2 ,…1 trong vòng lặp i trên
* Xét điều kiện nếu Ki < Kj thì tăng COUNT[j] thêm 1 đơn vị đồng thời đổi chỗ Ki và Kj  để thấy rõ quá trình di chuyển các số và ngược lại tăng COUNT[i] thêm 1 đơn vị.



1. **Shell sort**

**-** Yêu cầu bài toán:



- Code:

void shellSort(int a[], toaDo b[], int n) {

int interval, i, j, temp;

for (interval = n / 2; interval > 0; interval /= 2) {

for (i = interval; i < n; i++) {

temp = a[i];

for (j = i; j >= interval && a[j - interval] > temp; j -= interval)

{

a[j] = a[j - interval];

}

MPSapXep(b, j, i, a[j], temp);

a[j] = temp;

}

}

gotoXY(50, 2);

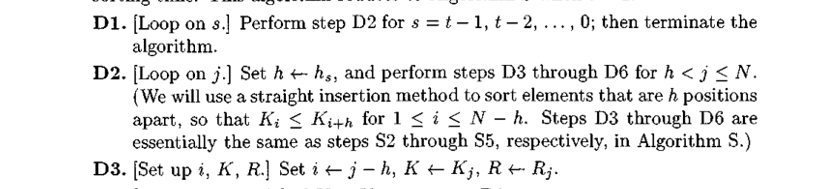
SetColor(13);

cout << "SAP XEP HOAN TAT!";

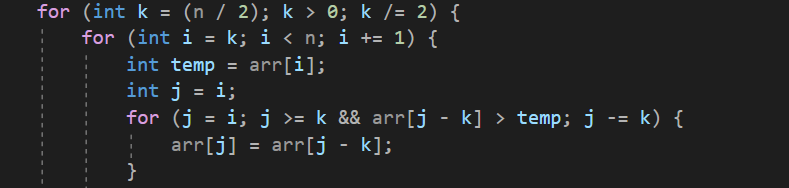
SetColor(7);

}

* Phân tích thuật toán:



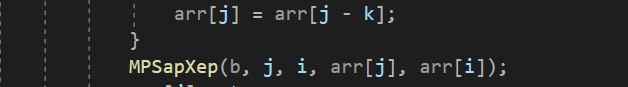
* Bước đầu tiên ta chia đôi mảng và tạo vòng lặp cho mảng giảm dần ½ theo từng vòng.
* Bước 2 : Sử dụng phương pháp chèn thẳng để sắp xếp các phần tử cách nhau h vị trí.
* Bước 3 : đặt i = j-h, K=Kj , R=Rj

- Bước 4 : so sánh nếu K ≥ Ki thì di chuyển tới bước 6





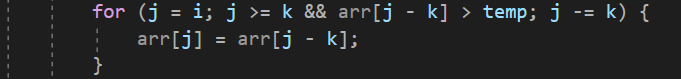
* Bước 6 : đổi chỗ giá trị Ri+h và R. Sau khi đổi chỗ thì gọi hàm MPSapXep để minh họa rõ được quá trình chuyển đổi số.





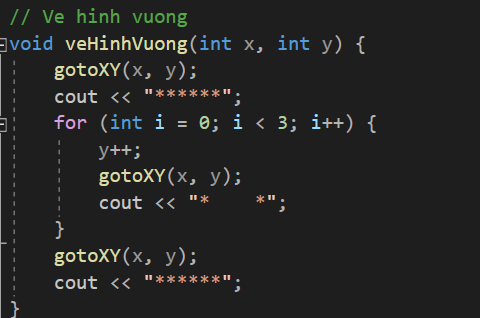
* Bước 5 : Trong trường hợp bước 4 so sánh không thỏa mãn thì nhảy sang bước 5 đặt Ri+h = Ri sau đó giảm i đi h đơn vị và tiếp tục so sánh quay về bước 4 với i>0.

+ Ta sử dụng vòng lặp với điều kiện so sánh nếu không thỏa thì sẽ giảm j đi k đơn vị để tiếp tục vòng lặp mới và so sánh.

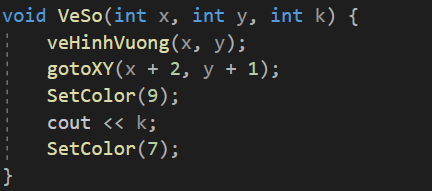


1. **Hàm vẽ mô phỏng**

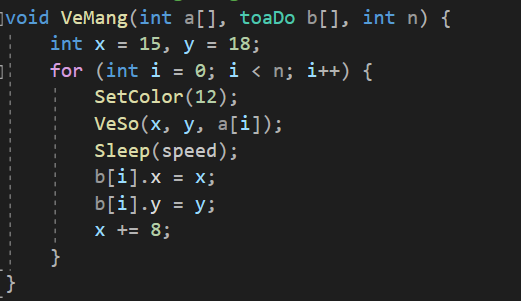
**5.1. Vẽ khung :**

****

**5.2. Điền số vào khung:**

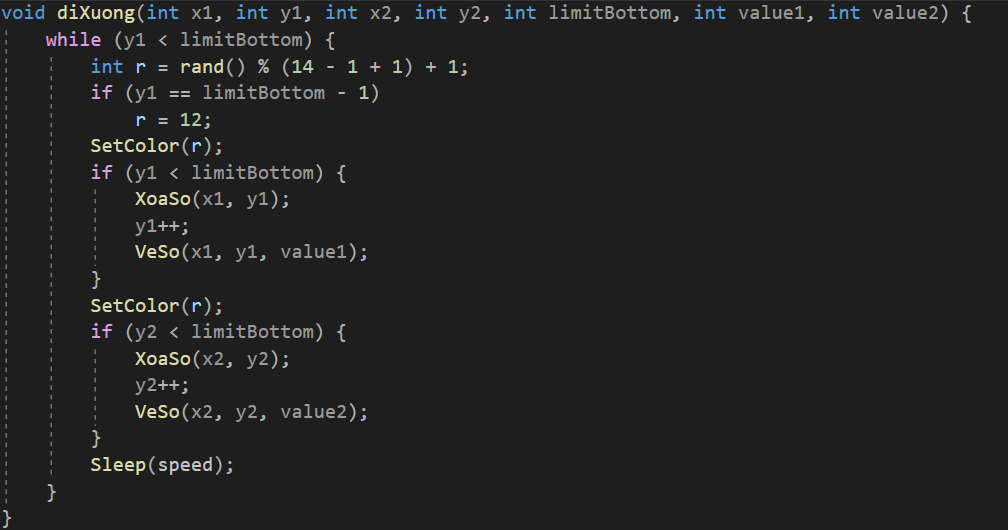


**5.3 Vẽ mảng :**



x = 15 và y = 18 là vị trí đặt phần tử đầu tiên cứ sau mỗi phần tử thì khoảng cách sẽ tăng lên 8 đơn vị ( khoảng cách giữa các phần tử là 8)

**5.4 Hàm đi xuống :**



**5.5 Hàm sắp xếp di chuyển các phần tử:**

void MPSapXep(toaDo b[], int pos1, int pos2, int value1, int value2) {

if (value1 == value2)

return;

int x1, x2, y1, y2;

x1 = b[pos1].x;

y1 = b[pos1].y;

x2 = b[pos2].x;

y2 = b[pos2].y;

// di chuyen len

int limitBottom = y1;

int limitUp = y1 - 5;

int limitLeft = x1;

int limitRight = x2;

while (y1 >= limitUp) {

int r = rand() % (14 - 1 + 1) + 1;

SetColor(r);

XoaSo(x1, y1);

y1--;

VeSo(x1, y1, value1);

SetColor(r);

XoaSo(x2, y2);

y2--;

VeSo(x2, y2, value2);

Sleep(speed);

}

int start = x1 + ((x2 - x1) / 2);

int end = y1 - 2;

gotoXY(start, end);

SetColor(1);

if (value1 < value2) {

cout << value1 << " < " << value2 << endl;

Sleep(1000);

XoaChu(start, end);

diXuong(x1, y1, x2, y2, limitBottom, value1, value2);

return;

}

else {

cout << value1 << " >= " << value2 << endl;

Sleep(1000);

XoaChu(start, end);

}

while (x1 < limitRight) {

int r = rand() % (14 - 1 + 1) + 1;

SetColor(r);

if (x1 < limitRight) {

XoaSo(x1, y1);

x1++;

VeSo(x1, y1, value1);

XoaSo(x2, y2);

}

SetColor(r);

if (x2 > limitLeft) {

XoaSo(x2, y2);

x2--;

VeSo(x2, y2, value2);

}

Sleep(speed);

}

diXuong(x1, y1, x2, y2, limitBottom, value1, value2);

}