Trần Nguyễn Tuấn Lộc 16038271

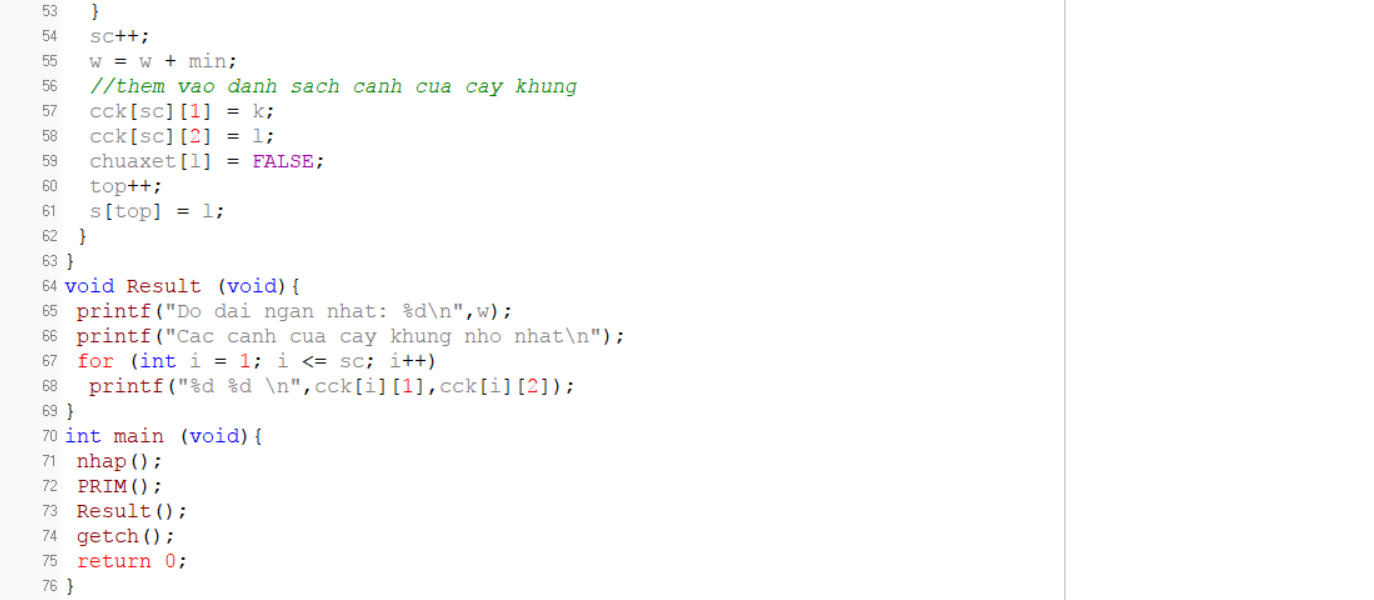
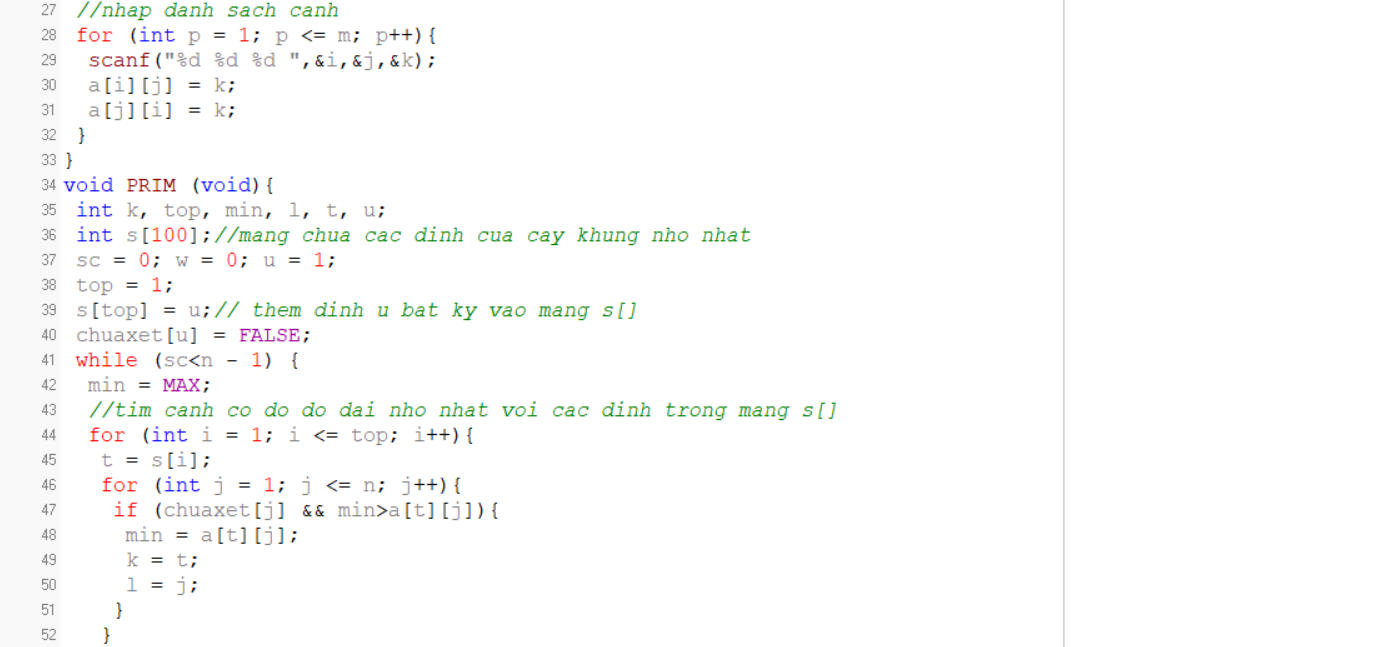
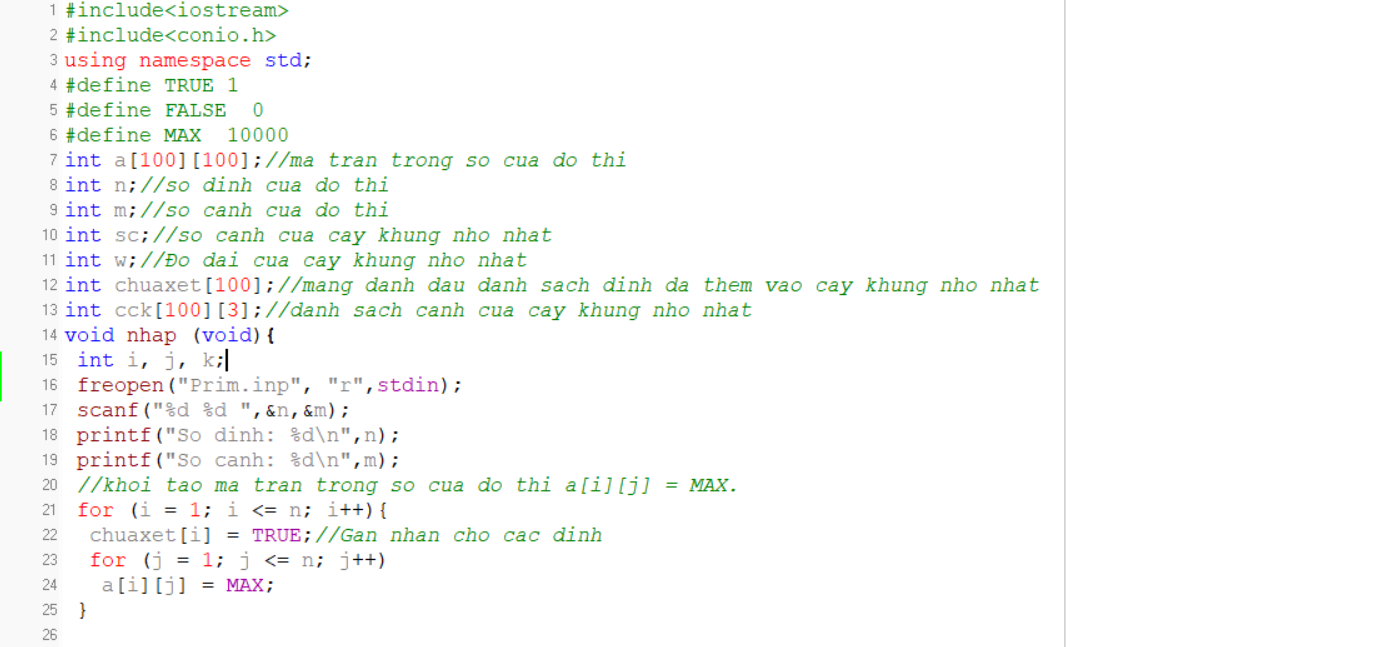
**Bài Báo Cáo Lý Thuyết Đồ Thị**

**Đề bài:** Viết chương trình hiện thực hóa giải thuật tìm cây khung nhỏ nhất nhất.

* ***Prim***

Input: Ma trận trọng số, số đỉnh, số cạnh của đồ thị; Số cạnh, độ dài, mảng đánh dấu danh sách đỉnh đã thêm vào, danh sách cạnh của cây khung nhỏ nhất.

Output: In ra màn hình console số đỉnh và số cạnh của đồ thị, độ dài của cây khung nhỏ nhất và các cạnh của cây.



- int a[100][100]: Ma trận trọng số của đồ thị.

- int n: Số đỉnh của đồ thị.

- int m: Số cạnh của đồ thị.

- int sc: Số cạnh của cây khung nhỏ nhất.

- int w: Độ dài của cây khung nhỏ nhất.

- int chuaxet[100]: Mảng đánh dấu danh sách đỉnh đã thêm vào cây khung nhỏ nhất.

- int cck[100][3]: Danh sách cạnh của cây khung nhỏ nhất.

- for (i = 1; i <= n; i++){

chuaxet[i] = TRUE; Gán nhãn cho các đỉnh

for (j = 1; j <= n; j++)

a[i][j] = MAX;

}

=> Khởi tạo ma trận trọng số của đồ thị a[i][j] = MAX.

- for (int p = 1; p <= m; p++){

scanf("%d %d %d ",&i,&j,&k);

a[i][j] = k;

a[j][i] = k;

}

=> Nhập danh sách cạnh.

- int s[100]: Mảng chứa các đỉnh của cây khung nhỏ nhất.

- s[top] = u: Thêm đỉnh u bất kì vào mảng s[].

- for (int i = 1; i <= top; i++){

t = s[i];

for (int j = 1; j <= n; j++){

if (chuaxet[j] && min>a[t][j]){

min = a[t][j];

k = t;

l = j;

}}}

=> Tìm cạnh có số độ dài nhỏ nhất với các đỉnh trong mảng s[].

- cck[sc][1] = k;

cck[sc][2] = l;

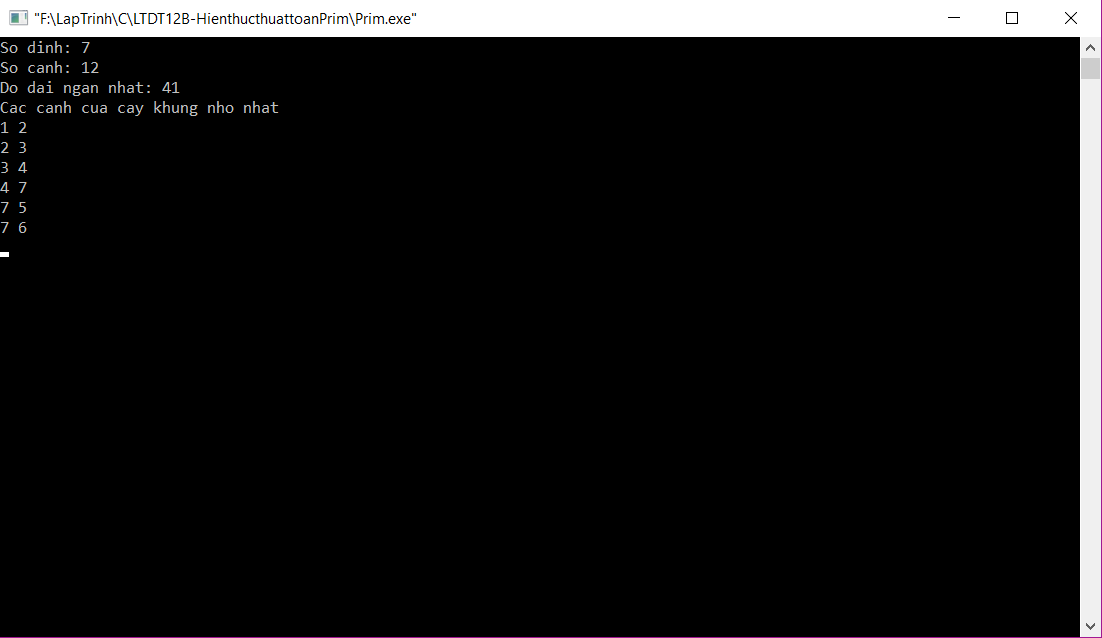
chuaxet[l] = FALSE;

top++;

s[top] = l;

=> Thêm vào danh sách cạnh của cây khung.

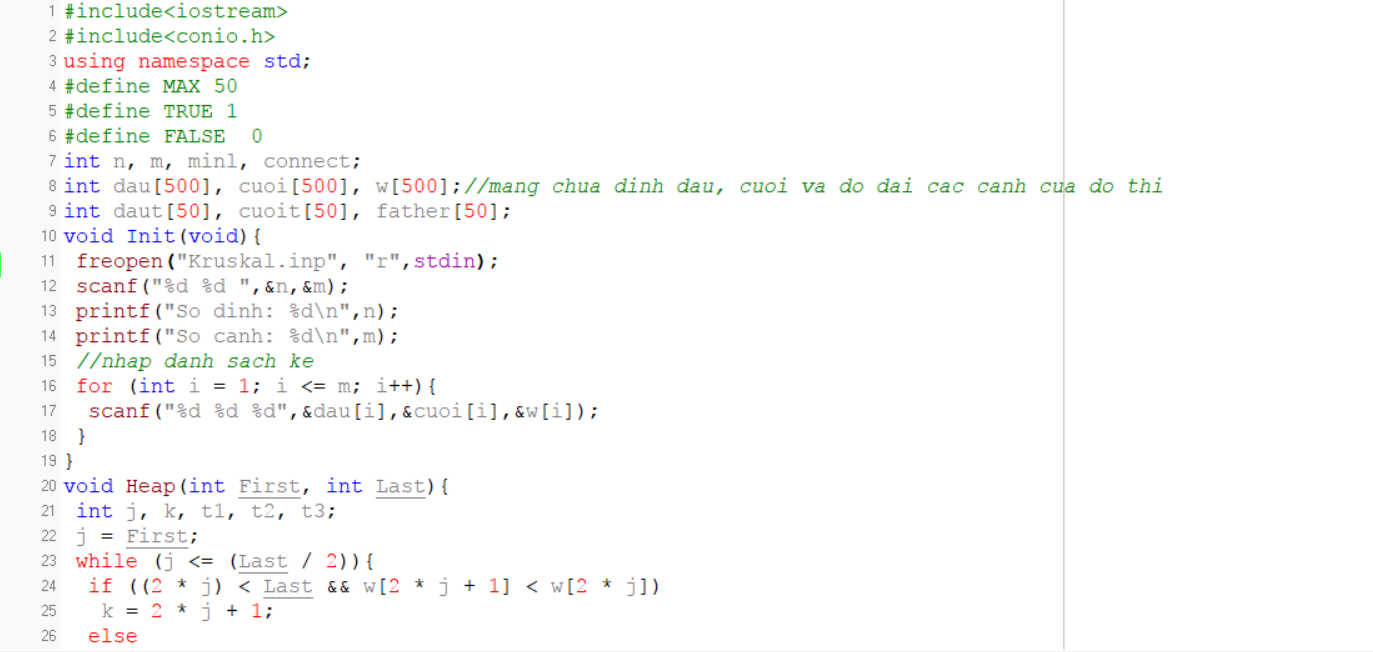
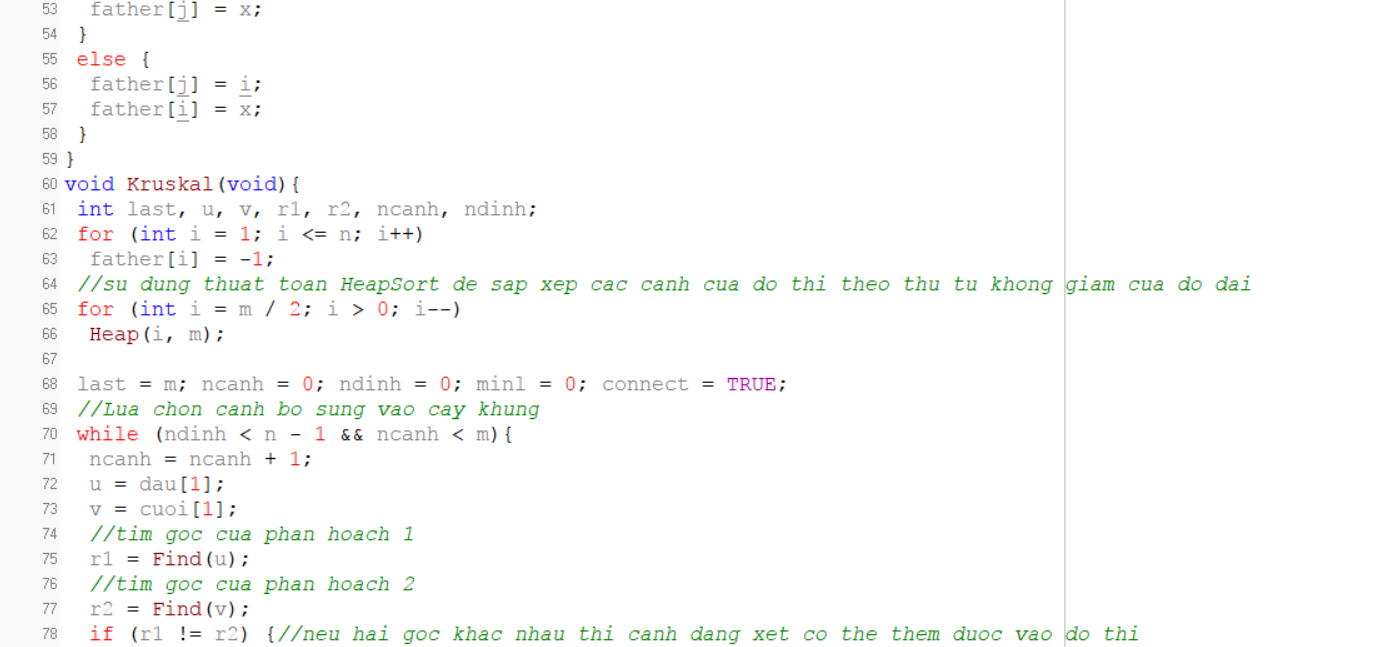
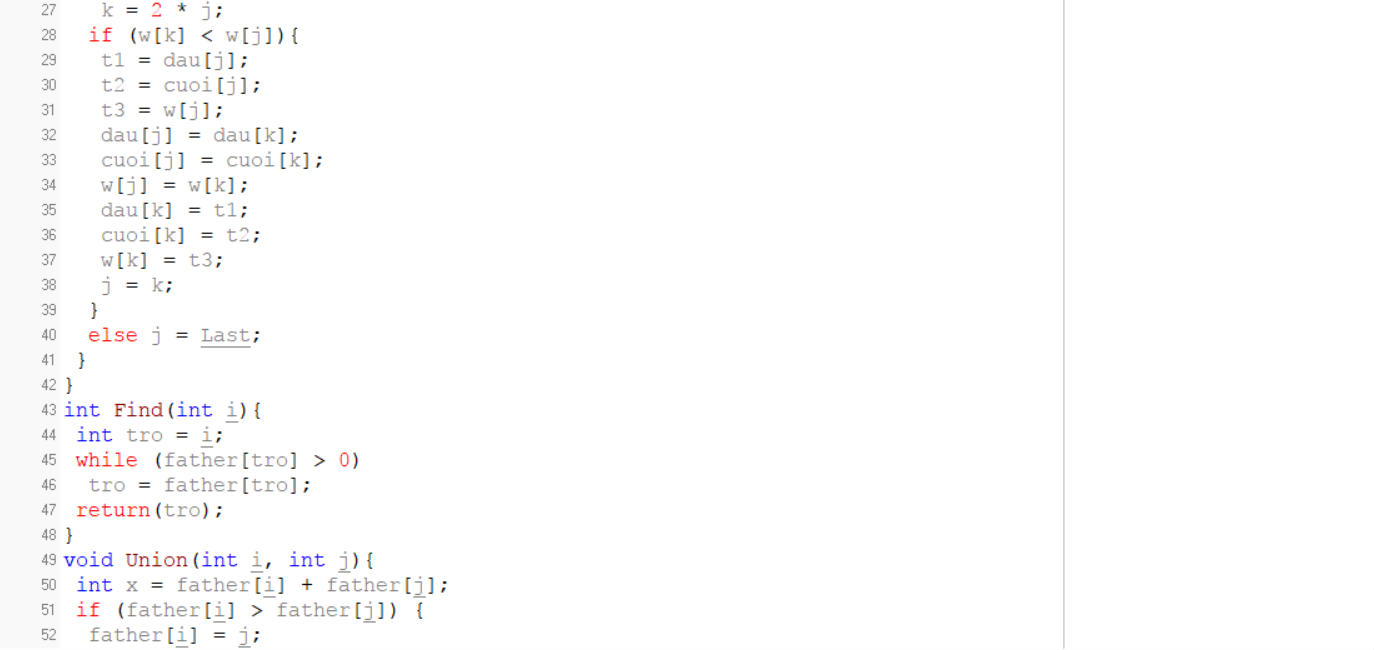
Kết quả thực nghiệm:



* ***Kruskal***

Input: Mảng chứa đỉnh đầu đỉnh, cuối và độ dài các cạnh của đồ thị.

Output: In ra màn hình console số đỉnh và số cạnh của đồ thị, độ dài của cây khung nhỏ nhất và các cạnh của cây.

- int dau[500], cuoi[500], w[500]:Mảng chứa đỉnh đầu đỉnh, cuối và độ dài các cạnh của đồ thị.

- for (int i = 1; i <= m; i++){

scanf("%d %d %d",&dau[i],&cuoi[i],&w[i]);

}

=> Nhập danh sách kế

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

Heap(i, m);

=> Sử dụng thuật toán HeapSort để sắp xếp các cạnh của đồ thị theo thứ tự không gian của đồ thị.

- while (ndinh < n - 1 && ncanh < m){

ncanh = ncanh + 1;

u = dau[1];

v = cuoi[1];

=> Lựa chọn cạnh bổ sung vào cây khung.

- r1 = Find(u): Tìm gốc của phân hạch 1.

- r2 = Find(v): Tìm gốc của phân hạch 2.

- if (r1 != r2): Nếu 2 gốc khác nhau thì cạnh đang xét có thể thêm được vào đồ thị

Kết quả thực nghiệm:

