BÀI TẬP VÀ THỰC HÀNH

MÔN HỌC

Lý thuyết đồ thị

2009

MỤC LỤC

CHƯƠNG 1: ĐẠI CƯƠNG VỀ ĐỒ THỊ

1 Xét ví dụ thực tế

Bài 1.1. Với mỗi trường hợp sau, vẽ các mô hình đồ thị biểu diễn các đường bay và nói rõ về loại đồ thị được dùng. Trong đó lịch bay mỗi ngày như sau:

- Từ TP.HCM: có một chuyến đến Hà Nội, một chuyến đến Đà Nẵng, một chuyến đến Phú Quốc, một chuyến đến Nghệ An, một chuyến đến Hải Phòng;

- Từ Hà Nội: có hai chuyến đến TP.HCM, một chuyến đến Đà Nẵng, một chuyến đến Nghệ An, một chuyến đến Hải Phòng;

- Từ Đà Nẵng: có một chuyến đến Hải Phòng, hai chuyến bay đến TP.HCM; một chuyến đến Hà Nội;

- Từ Nghệ An: có một chuyến đến Hà Nội, một chuyến đến TP.HCM;

- Từ Hải Phòng: có một chuyến đến Hà Nội, một chuyến đến TP.HCM, và một chuyến đến Đà Nẵng;

- Từ Phú Quốc: có một chuyến đến TP.HCM.

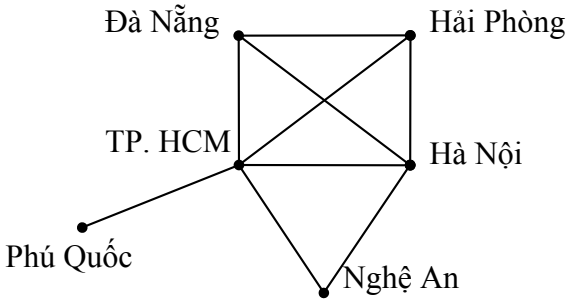
a) Đồ thị biểu diễn các thành phố có chuyến bay giữa chúng.

b) Đồ thị biểu diễn số chuyến bay hoạt động giữa các thành phố, cộng với một khuyên biểu thị chuyến du lịch đặc biệt ngắm cảnh thành phố, cất và hạ cánh tại Phú Quốc.

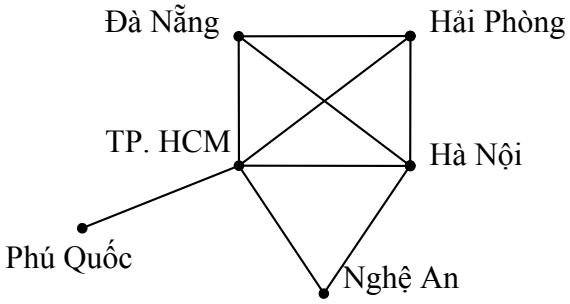
c) Đồ thị biểu diễn đầy đủ thông tin về hướng bay và số chuyến bay giữa các thành phố.

Phần hướng dẫn

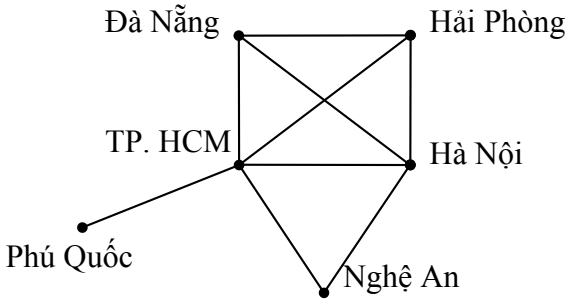
a) Đồ thị vô hướng



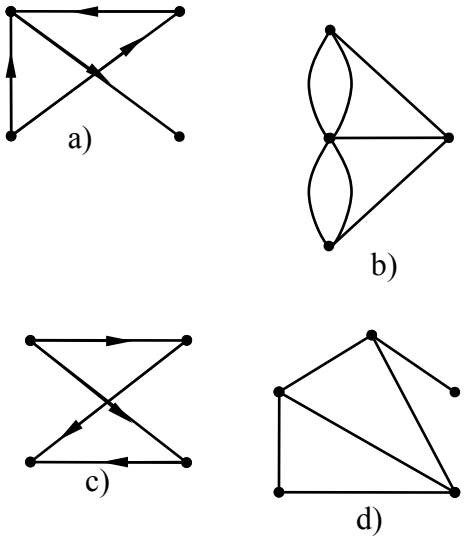
b) Đa đồ thị vô hướng



c) Đa đồ thị có hướng



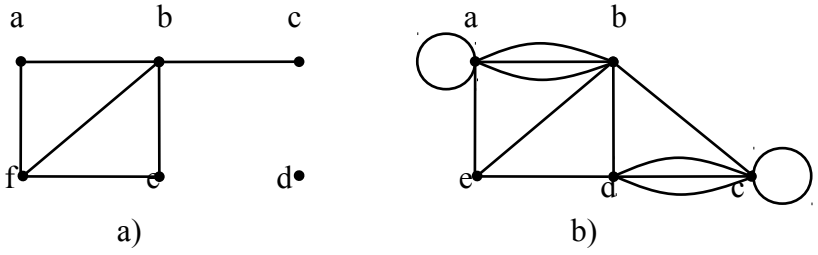
Bài 1.2. Xác định xem đồ thị nào sau đây là đồ thị đơn, đa đồ thị, đồ thị có hướng.

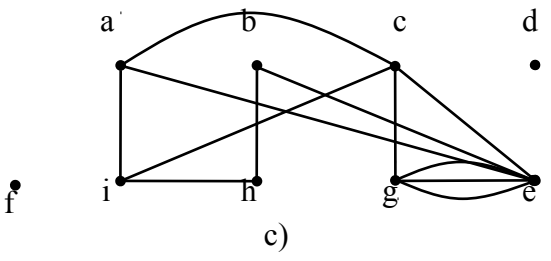


Bài 1.3. Trong trận đấu vòng tròn, đội H thắng đội G, đội C, và đội A; đội G thắng đội A và đội C; đội C thắng đội A. Hãy mô hình hóa kết quả này bằng một đồ thị có hướng…

2 Các thuật ngữ đồ thị

Bài 2.1. Xác định số lượng các đỉnh, số lượng các cạnh, và bậc của các đỉnh trong các đồ thị sau. Cho biết đỉnh nào là đỉnh cô lập, đỉnh nào là đỉnh treo.



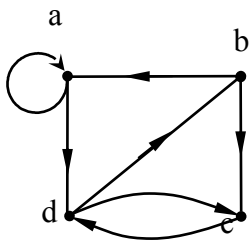


Bài 2.2. Tìm tổng các bậc của các đỉnh trong các đồ thị ở các Bài 2.1, và kiểm chứng rằng nó bằng hai lần số các cạnh trong đồ thị.

Bài 2.3. Có thể tồn tại một đồ thị đơn có 15 đỉnh, mỗi đỉnh có bậc bằng 5 không? Tại sao?

Bài 2.4. Trong một buổi chiêu đãi, mọi người đều bắt tay nhau. Chứng tỏ rằng tổng số người được bắt tay là một số chẵn. Giả sử không ai tự bắt tay mình.

Bài 2.5. Xác định số đỉnh, số cạnh, số bậc vào và số bậc ra của mỗi đỉnh đối với đồ thị có hướng sau.



Bài 2.6. Hãy xác định tổng các bậc vào và tổng các bậc ra các đỉnh của đồ thị trong bài 2.5 một cách trực tiếp. Chứng tỏ rằng chúng đều bằng tổng các cạnh của đồ thị.

Bài 2.7. Đồ thị sẽ có bao nhiêu cạnh nếu nó có các đỉnh bậc 4, 3, 3, 2, 2. Vẽ một đồ thị như vậy.

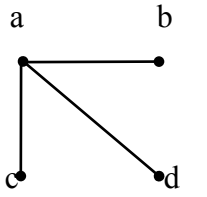
Bài 2.8. Có tồn tại đồ thị đơn chứa năm đỉnh với các bậc sau đây? Nếu có hãy vẽ đồ thị đó.

a) 3, 3, 3, 3, 2.

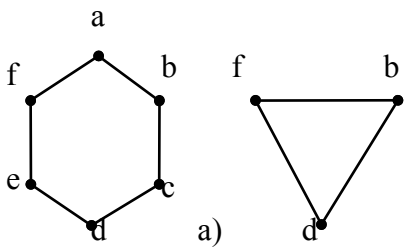
b) 1, 2, 3, 4, 5.

c) 1, 2, 3, 4, 4.

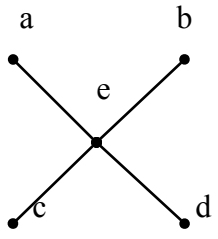
Bài 2.9. Vẽ tất cả các đồ thị con của đồ thị sau.

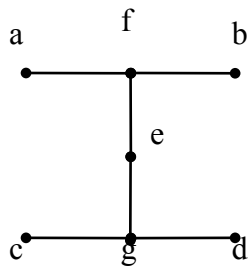


Bài



a)



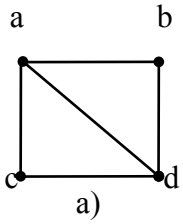


b)

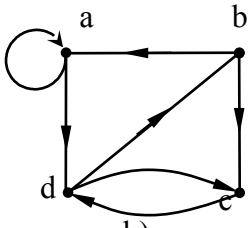
2.10. Tìm hợp của các cặp đồ thị đơn sau

3 Biểu diễn các đồ thị và sự đẳng cấu đồ thị

Bài 3.1. Dùng danh sách kề biểu diễn các đồ thị sau.



a)



b)

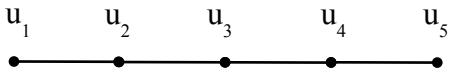
Bài 3.2. Biểu diễn các đồ thị trong bài 3.1 bằng ma trận kề.

Bài 3.3. Vẽ các đồ thị ứng với ma trận kề được cho như sau.

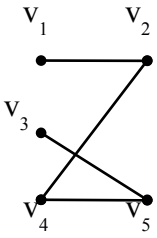
a) $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ b) $\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$ c) $\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$

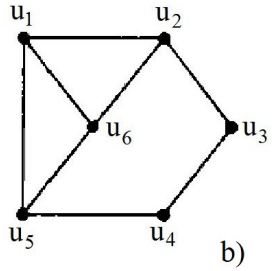
Bài 3.4. Dùng ma trận liên kết để biểu diễn các đồ thị trong Bài 3.1.

Bài 3.5. Xác định xem các cặp đồ thị đã cho có là đẳng cấu không.

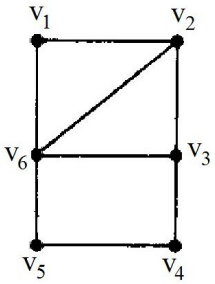


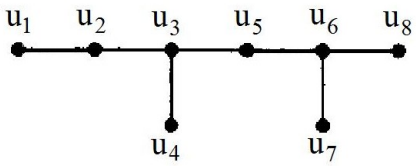
a)

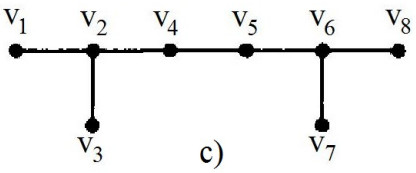




b)







c)

4 Tính liên thông

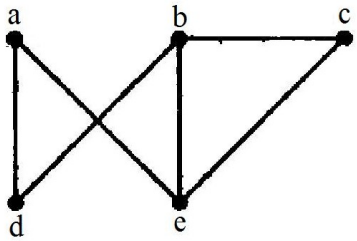
Bài 4.1. Các danh sách đỉnh sau đây có tạo nên đường đi trong đồ thị bên dưới hay không? Đường đi nào là đơn? Đường đi nào là chu trình? Độ dài của các đường đi này là bao nhiêu?

a) (a, e, b, c, b)

b) (a, e, a, d, b, c, a)

c) (e, b, a, d, b, e)

d) (c, b, d, a, e, c)



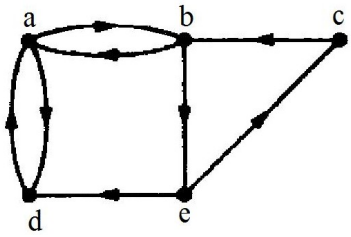
Bài 4.2. Các danh sách đỉnh sau đây có tạo nên đường đi trong đồ thị bên dưới hay không? Đường đi nào là đơn? Đường đi nào là chu trình? Độ dài của các đường đi này là bao nhiêu?

a) (a, b, e, c, b)

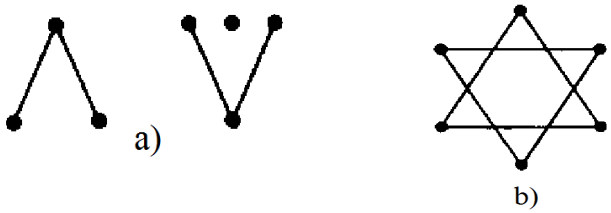
b) (a, d, a, d, a)

c) (a, d, b, e, a)

d) (a, b, e, c, b, d, a)

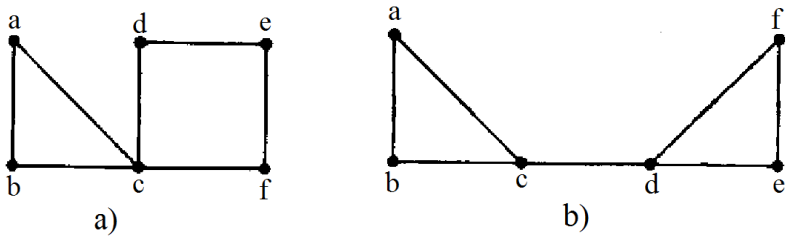


Bài 4.3. Xác định xem các đồ thị đã cho có liên thông không.



Bài 4.4. Có bao nhiêu thành phần liên thông trong các đồ thị ở các Bài tập 4.3? Tìm các thành phần liên thông đó.

Bài 4.5. Tìm tất cả các đỉnh cắt và cạnh cắt của đồ thị.



Bài thực hành số 1: Biểu diễn đồ thị

Bài tập 1:

Nhập vào ma trận kề của một đơn đồ thị (từ bàn phím và đọc từ tập tin).

a. Kiểm tra tính hợp lệ của đồ thị (giá trị trên đường chéo chính đều bằng 0).
b. Kiểm tra xem đồ thị là vô hướng hay hữu hướng?
c. Nếu ma trận kề được nhập từ bàn phím thì xuất ra thành tập tin *matranke.txt*
d. Nếu ma trận được đọc từ tập tin thì xuất kết quả ma trận ra màn hình hiển thị.
e. Xuất ra bậc của tất cả các đỉnh của đồ thị (số cạnh nối tới đỉnh).
f. Kiểm tra tính liên thông của đồ thị? Xuất ra tất cả các thành phần liên thông nếu có.

Bài tập 2:

Nhập vào ma trận trọng số của một đơn đồ thị (từ bàn phím và đọc từ tập tin).

a. Kiểm tra tính hợp lệ của đồ thị (giá trị trên đường chéo chính đều bằng 0).
b. Kiểm tra xem đồ thị là vô hướng hay hữu hướng?
c. Nếu ma trận kề được nhập từ bàn phím thì xuất ra thành tập tin *trongso.txt*
d. Nếu ma trận được đọc từ tập tin thì xuất kết quả ma trận ra màn hình hiển thị.
e. Xuất ra cạnh có trọng số nhỏ nhất và lớn nhất.

Hướng dẫn:

Chương trình nhập vào ma trận kề của đồ thị từ bàn phím

```
#include <stdio.h>
#include <conio.h>
```

```
main()
{
	int n,m,i,j;
	int a[100][100];
	// Doc n, m tu ban phim
	printf(" Nhap n, m ");
	scanf("%d %d",&n,&m);

	// Doc mang a kich thuoc n*m
	for (i=0;i<n;i++)
		for (j=0;j<m;j++)
		{
			printf(" Nhap a[%d][%d]: " ,i,j);
			scanf("%d",&a[i][j]);
		}
	// Xuat mang a ra man hinh
	for (i=0;i<n;i++)
	{
		for (j=0;j<m;j++)
		{
			fprintf(f,"%3d",a[i][j]);
		}
		fprintf(f,"\n");
	}
	getch();
}
```

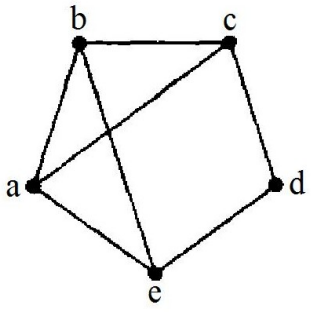
Chương trình đọc vào ma trận kề của đồ thị từ tập tin.

```
#include <stdio.h>
#include <conio.h>
main()
{
	int n,m,i,j;
	int b[100][100];
	FILE* f;
	// Doc file D:\matranke.txt
	f = fopen("D:\\matranke.txt","rt");
	fscanf(f,"%d%d",&n,&m);
	for (i=0;i<n;i++)
	{
```

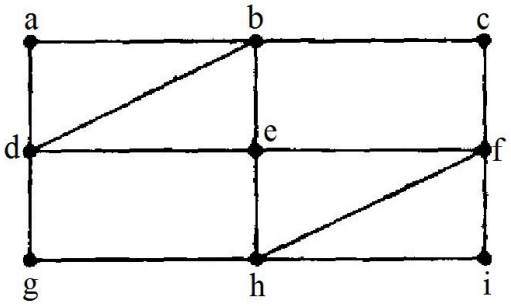
```
                for (j=0;j<m;j++)
                {
                        fscanf(f,"%d",&b[i][j]);
                }
        }
        fclose(f);

        // In mang b
        printf(" Mang B co n:%d    m:%d\n",n,m);
        for (i=0;i<n;i++)
        {
                for (j=0;j<m;j++)
                {
                        printf("%3d",b[i][j]);
                }
                printf("\n");
        }
        getch();
}
```

CHƯƠNG 2: ĐỒ THỊ EULER VÀ ĐỒ THỊ HAMILTON

Bài 1. Xác định xem có tồn tại chu trình Euler trong các đồ thị sau hay không. Vẽ chu trình đó khi nó tồn tại.



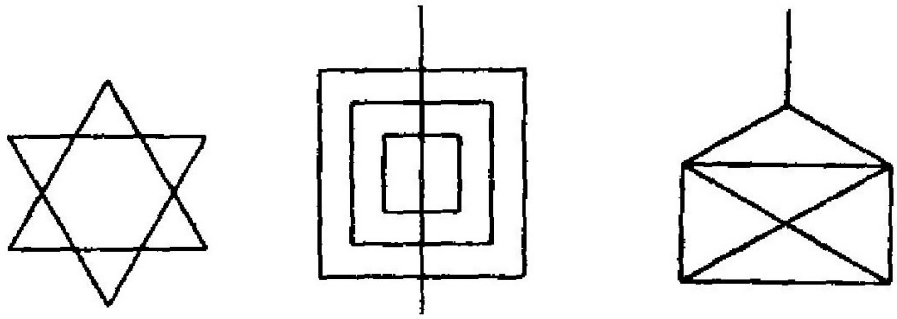
a)



b)

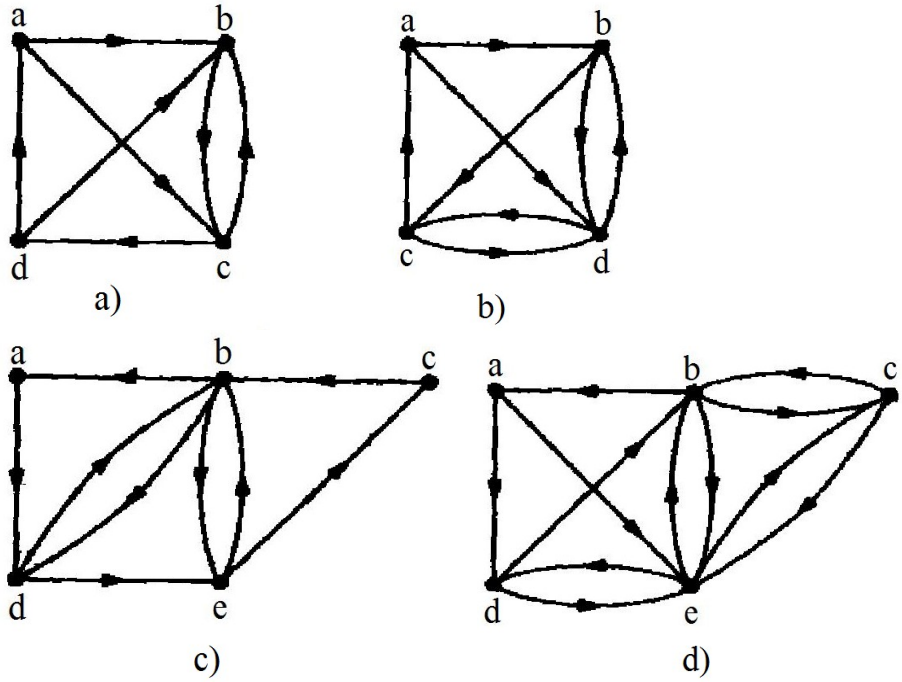
Bài 2. Xác định xem các đồ thị trong Bài 5.1 có đường đi Euler không. Vẽ các đường đi đó nếu có.

Bài 3. Xác định xem có thể vẽ các bức tranh sau bằng một nét liền, không nhấc bút lên khỏi mặt giấy không?



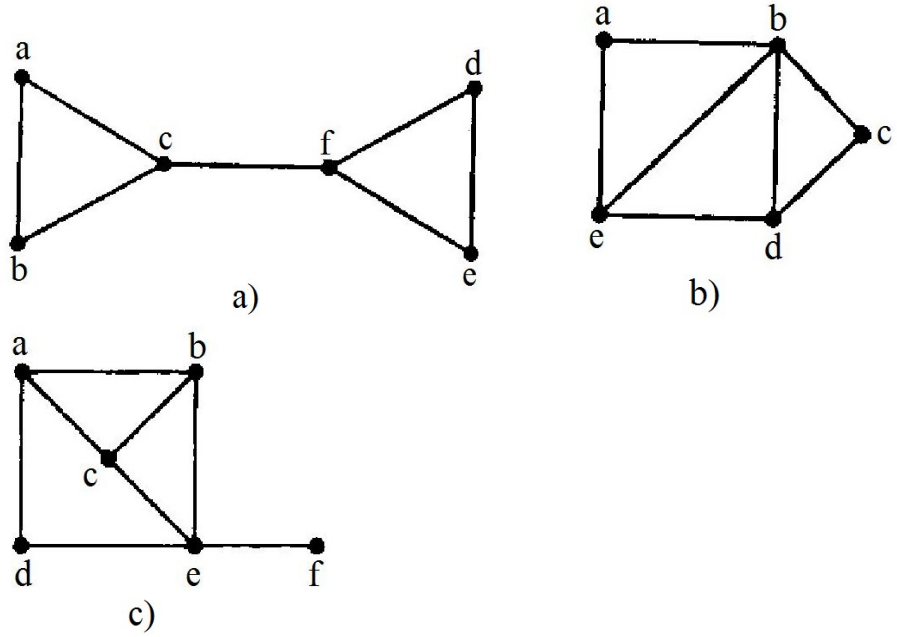
a) b) c)

Bài 4. Xác định sự tồn tại chu trình Euler trong các đồ thị có hướng sau. Vẽ các chu trình này nếu chúng tồn tại.



Bài 5.Xác định xem đồ thị có hướng trong Bài 5.4 có đường đi Euler hay không. Vẽ các đường đi Euler này nếu có.

Bài 6. Xác định đồ thị đã cho có chứa chu trình Hamilton hay không. Nếu có hãy tìm một chu trình như thế. Nếu không có hãy giải thích lý do vì sao không tồn tại.



Bài 7 Đồ thị trong Bài 5.6 có đường đi Hamilton không? Nếu có, hãy tìm đường đó. Nếu không có, cho biết lý do tại sao không tồn tại một đường đi như vậy.

Bài thực hành số 2: Duyệt đồ thị

Bài tập 1:

Cài đặt thuật toán duyệt theo chiều sâu và chiều rộng với đồ thị được cho bởi ma trận kề đọc từ tập tin *matranke.txt*, chương trình cho phép nhập vào đỉnh xuất phát và hiển thị kết quả duyệt.

Bài tập 2:

Cài đặt chương trình cho phép nhập vào 2 đỉnh x và y của đồ thị, kiểm tra xem có đường đi từ x tới y (và ngược lại) hay không?

Hướng dẫn:

4.1.1 Duyệt đồ thị theo chiều sâu (thuật toán DFS)

Ý tưởng chính của thuật toán có thể trình bày như sau:

- Ta sẽ bắt đầu tìm kiếm từ một đỉnh v_0 nào đó của đồ thị.
- Sau đó chọn u là một đỉnh tuỳ ý kề với v_0 và chưa được duyệt.
- Lặp lại quá trình đối với u. Đây là thủ tục đệ quy.
- Quá trình đệ quy kết thúc khi không chọn được đỉnh u nữa.

Thủ tục đệ quy được viết như sau:

Thủ tục DFS(v);
*(*tim kiem theo chieu sau bat dau tu dinh v; cac bien Chuaxet, Ke la bien toan cuc*)*
Bắt đầu
Tham_dinh(v);
Chuaxet[v]:=false;
For u là Ke(v) do
If Chuaxet[u] then DFS(u);
Kết thúc (*dinh v da duyet xong*)
	Duyệt toàn bộ đồ thị theo chiều sâu:
	Bắt đầu
	for v thuộc V do Chuaxet[v]:=true;
	for v thuộc V do
	if Chuaxet[v] then DFS(v);
	Kết thúc
	Độ phức tạp là : O(n+m)

4.1.2 Duyệt đồ thị theo chiều rộng (thuật toán BFS)

Ý tưởng chính của thuật toán như sau:

- Sử dụng 1 hàng đợi để lưu trữ các đỉnh sẽ được duyệt.
- Ban đầu đưa đỉnh bắt đầu duyệt u vào hàng đợi.
- Thực hiện quá trình lặp cho đến khi hàng đợi rỗng. Mỗi bước lặp làm như sau:
  - Lấy 1 đỉnh v ra khỏi hàng đợi. Thực hiện các công việc cần thiết với đỉnh đó.
  - Xác định các đỉnh w kề với đỉnh v mà chưa duyệt.
  - Đưa các đỉnh w này vào hàng đợi.

Thủ tục được viết như sau:

Thủ tục BFS(u); *(*Tim kiem theo chieu rong bat dau tu dinh u, cac bien Chuaxet, Ke la bien cuc bo*)*	Duyệt toàn bộ đồ thị theo thuật toán BFS:

Bắt đầu
QUEUE:= Ø;
QUEUE ← u; *(* nap u vao QUEUE*)*
Chuaxet[v]:=false;
While QUEUE ≠ Ø do
Begin
v ← QUEUE:; *(*lay p tu QUEUE:*)*
Tham_dinh(v);
For w là Ke(v) do
If Chuaxet[w] them
Begin
QUEUE ← w;
Chuaxet[w]:=false;
End;
End;
Kết thúc
	Bắt đầu
	*(*khởi tạo ban đầu *)*
	for mọi k thuộc V do
	Chuaxet[k]:=true;
	for u thuộ V do
	if Chuaxet[u] then BFS(v);
	Kết thúc
	
	Thuật toán BFS có độ phức tạp: O(n+m)

4.1.3 Ứng dụng của thuật toán duyệt

Ta thấy, tư tưởng của thuật toán duyệt là xuất phát từ 1 đỉnh u, và đi thăm các đỉnh v còn lại trong đồ thị nếu như tồn tại 1 đường đi từ đỉnh u đến v.

Như vậy ta sẽ giải quyết được **2 bài toán** như sau:

- Tìm 1 đường đi giữa 2 đỉnh bất kỳ.
- Kiểm tra tính liên thông của đồ thị và tìm các thành phần liên thông trong đồ thị.

Để **xác định được 1 đường đi từ đỉnh u đến đỉnh v**, ta cần phải làm như sau:

- Dùng thủ tục duyệt theo chiều sâu với đỉnh u.
- Trong quá trình duyệt có sử dụng mảng các biến trạng thái Chuaxet[k] (hay Free[k]) để kiểm tra đỉnh k đã được duyệt chưa.
- Để xác định đường đi ta dùng mảng biến lưu trữ vết Truoc[k] = t, có nghĩa là từ đỉnh t sẽ chuyển đến đỉnh k.

Bổ sung vào các thủ tục DFS và BFS như sau:

Đối với DFS bổ sung code như sau:
 	
If Chuaxet[u] then
Begin
Truoc[u]:= v;
Chuaxet[u] = false;
DFS(u);
End;
	Đối với BFS bổ sung code như sau:
	
	If Chuaxet [u] then
	Begin
	QUEUE ← u;
	Chuaxet[u]:= false;
	Truoc[u]:= p;
	End;

Xác định đường đi từ u đến v như sau:

```
While (v ≠ u) do
Begin
    In giá trị v ra ngoài màn hình;
    v = Truoc[v] ; // gán v bằng đỉnh mà từ nó nhảy đến v hiện tại
End;
In giá trị u ra màn hình;
```

Để **xác định số thành phần liên thông trong đồ thị** thì ta dùng thêm 1 biến count đế đếm. Khi đó các bước phải làm như sau:

- Khởi tạo các mảng biến cần thiết, và count = 0.
- Xét với mọi đỉnh u trong đồ thị, nếu như đỉnh u chưa được duyệt thì:
  - Tăng giá trị của biến count lên 1 đơn vị
  - Thực hiện các thuật toán duyệt đối với đỉnh u.
- Nếu kết quả biến count = 1 thì đồ thị liên thông. Trong trường hợp còn lại thì giá trị của count chính là số thành phần liên thông của đồ thị.

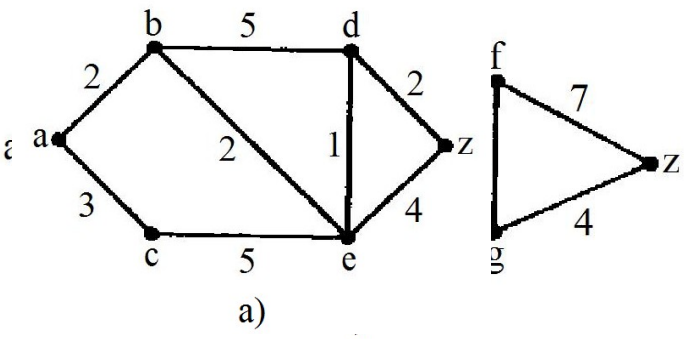
Viết thêm code để xác định số thành phần liên thông:

```
Count :=0;
If Chuaxet[u] then
Begin
    Count := Coutn + 1;
    DFS(u);
End;
```

```
Count :=0;
If Chuaxet[u] then
Begin
    Count := Coutn + 1;
    BFS(u);
End;
```

CHƯƠNG 3
ĐỒ THỊ CÓ TRỌNG SỐ VÀ ĐƯỜNG ĐI NGẮN NHẤT

Bài 1. Tìm chiều dài của đường đi ngắn nhất giữa a và z trong đồ thị có trọng số sau đây

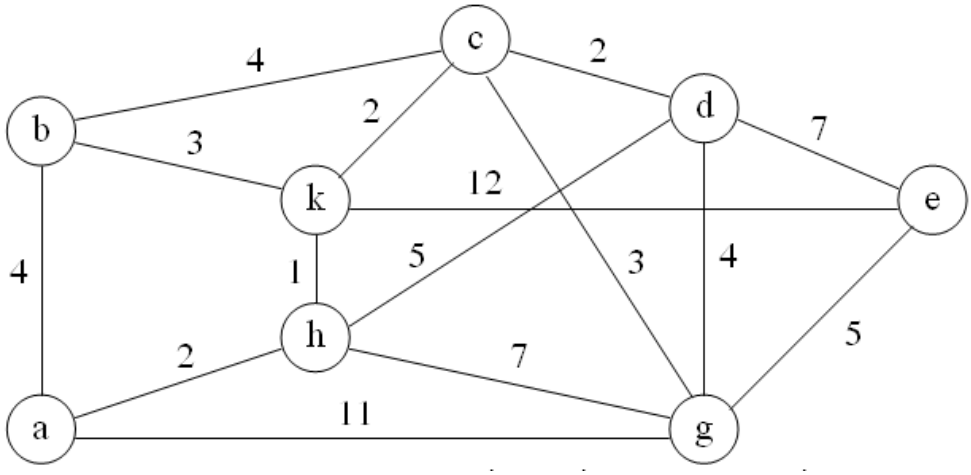


a)

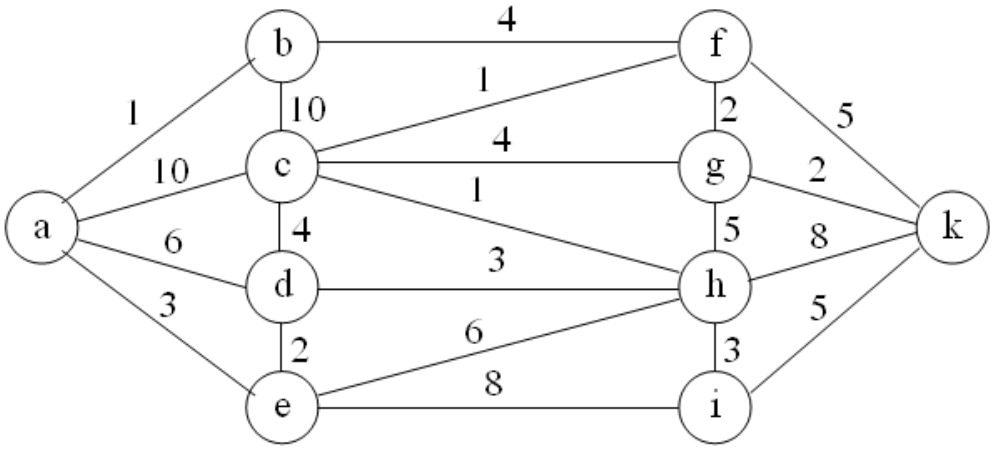
Bài 2. Tìm độ dài của đường đi ngắn nhất giữa các cặp đỉnh sau đây trong các đồ thị có trọng số ở Bài 6.1.

a) a và d b) a và f c) c và f d) b và z

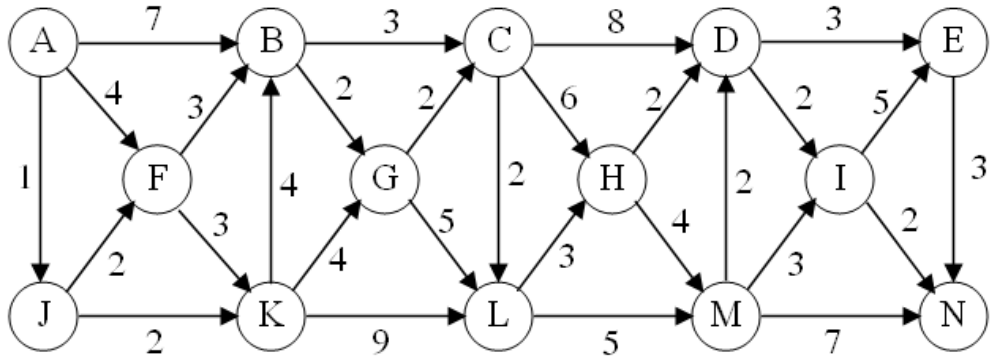
Bài 3. Tìm độ dài đường đi ngắn nhất từ đỉnh A đến tất các đỉnh còn lại trong các đồ thị sau:



Bài 4. Tìm độ dài đường đi ngắn nhất từ đỉnh A đến tất các đỉnh còn lại trong các đồ thị sau:



Bài 5. Cho đồ thị có trọng số như hình dưới đây. Hãy tìm đường đi ngắn nhất từ đỉnh A đến đỉnh N.



Bài thực hành số 3:

Bài tập:

Cài đặt thuật toán Ford-Bellman, Dijkstra, Floyd- Warshall với đồ thị được cho bởi ma trận trọng số đọc từ tập tin *trongso.txt*, chương trình cho phép nhập vào đỉnh xuất phát và hiển thị kết quả duyệt.

Hướng dẫn:

4.1.1 Thuật toán tìm đường đi ngắn nhất Ford-Bellman

Thuật toán này Ford-Bellman **xác định tất cả các đường đi ngắn nhất từ một đỉnh u cho trước đến các đỉnh v còn lại** trong đồ thị.

Tư tưởng thuật toán như sau:

- Sử dụng mảng D[v] là giá trị khoảng cách bé nhất đi từ đỉnh u đến đỉnh v. Ban đầu chỉ có D[u] = 0, còn tất cả các đỉnh v còn lại đều có D[v] = ∞.
- Để tối ưu các giá trị D[v] thì ta cần chọn các đỉnh trung gian k, để sau đó so sánh giá trị D[v] hiện tại với tổng khoảng cách D[k] và A[k][v]. Lưu ý ở đây A[k][v] chính là trọng số trên cạnh (k➔v).
- Như vậy với mỗi đỉnh trung gian k ta cần phải tối ưu (n-1) đỉnh còn lại. Hơn nữa ta có n cách chọn giá trị của k. Như vậy ta có 2 vòng lặp lồng nhau.
- Giả sử ở thời điểm T1, chúng ta dùng đỉnh k1 là đỉnh trung gian, và tối ưu được giá trị tại đỉnh k2. Đến thời điểm T2 , chúng ta chọn k2 là đỉnh trung gian thì lại có thể tối ưu được giá trị tại đỉnh k1. Như vậy việc chọn 1 đỉnh là trung gian phụ thuộc vào (n-1) đỉnh còn lại.Để vét hết các khả năng, ta cũng cho mỗi đỉnh được chọn làm trung gian (n-1) lần. Như vậy ta có vòng lặp thứ 3 nằm ngoài 2 vòng lặp trên.
- Tuy nhiên để hiệu quả hơn, tại mỗi bước của vòng lặp ngoài cùng ta kiểm tra xem các giá trị tại các đỉnh có thay đổi không,nếu đã tối ưu và không thay đổi nữa thì ta sẽ dừng chương trình.

Về cơ bản thuật toán mô tả như sau:

```
Procedure Ford_Bellman (u)
Begin
    for i:=1 to n-1 do            // số lần chọn một đỉnh làm trung gian
        for k  thuộc V\{u} do  // chọn 1 đỉnh là trung gian
            for v thuộc V do    // tối ưu với tất cả các đỉnh còn lại
                if d[v] > d[k] +a[k][v] then    // nếu đk tối ưu xảy ra
                begin
                    d[v]:=d[k]+a[k][v];     //  gán giá trị tối ưu mới
                     Truoc[v]:= k;             // lưu lại vết di chuyển.
                end;
End;
```

Một số chú ý:

- Thuật toán trên làm việc với đồ thị có trọng số không âm,hoặc không có chu trình mà tổng trọng số là âm.

Thuật toán trên làm việc với ma trận kề sẽ có độ phức tạp là $O(n^3)$.

4.1.2 Thuật toán tìm đường đi ngắn nhất Dijkstra

Thuật toán này **xác định đường đi ngắn nhất giữa 2 đỉnh cụ thể từ u đến v**.

Tư tưởng của thuật toán như sau:

- Dùng một tập S lưu trữ các đỉnh đã được duyệt. Cách đơn giản nhất phân biệt giữa đỉnh đã duyệt với đỉnh chưa duyệt là dùng mảng biến trạng thái Free[k].
- Dùng mảng phụ D[k] xác định giá trị khoảng cách ngắn nhất đi từ u đến k. Ban đầu ta chỉ gán D[u] = 0 và D[k] = ∞ với v ≠ u.
- Bắt đầu quá trình duyệt các đỉnh k khác u và quá trình lặp chỉ kết thúc khi :
  - Đã đạt đến đỉnh đích v .
  - Hoặc không thể duyệt tiếp được nữa.
- Mỗi bước của quá trình duyệt như sau:
  - Xác định đỉnh k trong số các đỉnh chưa được duyệt, sao cho D[k] là bé nhất.
  - Nếu k được chọn chính là đỉnh đích v ta kết thúc vòng lặp.
  - Nếu giá trị D[k] được chọn là ∞ (tức là không chọn thêm được nữa) ta cũng kết thúc vòng lặp.
  - Trong trường hợp còn lại, ta đưa k vào tập S (gán Free[k] = 0) và tối ưu các đỉnh còn lại theo nguyên tắc sau: nếu D[t] > D[k] + A[k][t] ➔ D[t] = D[k] + A[k][t];

Thuật toán được mô tả như sau:

```
Procedure Dijstra (u, v);
Begin
    while True do
    begin
        Tìm đỉnh k thoả mãn Free[k] = 1 và D[k] nhỏ nhất ;
        if D[k] = ∞ hoặc k = v then BREAK;
        For v  thuộc V do
            If Free[v] = 1 và d[v]>d[k]+a[k][v] then
            begin
                d[v]:=d[k]+a[k][v];
                 Truoc[v]:=u;
            end;
    end;
End;
```

Một số chú ý:

- Thuật toán làm việc với đồ thị có trọng số không âm,hoặc không có chu trình mà tổng trọng số âm.
- Độ phức tạp của thuật toán là $O(n^2)$.

4.1.3 Thuật toán tìm đường đi ngắn nhất Floyd- Warshall

Thuật toán này **xác định đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị**.

Tư tưởng thuật toán như sau:

- Để xác định đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị, thì ta phải xác định tất cả n*(n-1) giá trị, tương ứng với các cặp (u,v) trên đồ thị.
- Khi đó ta dùng luôn ma trận trọng số A_{nxn} làm ma trận lưu kết quả. Khi đó A[i][j] là giá trị khoảng cách ngắn nhất đi từ đỉnh i đến đỉnh j.
- Tư tưởng thuật toán này cũng khá đơn giản:
  - Chọn 1 đỉnh là trung gian, giả sử là đỉnh k.
  - Tối ưu hóa đường đi từ đỉnh u đến đỉnh v theo k. Tức là ta so sánh A[u][v] với tổng A[u][k] và A[k][v]. Nếu đi từ u đến v dài hơn so với đi từ u qua k rồi đến v thì ta sẽ tối ưu giá trị A[u][v].
  - Ta thấy có n cách chọn đỉnh k, với mỗi k có n cách chọn đỉnh xuất phát u và với mỗi u ta có n cách chọn đỉnh kết thúc v. Như vậy ta có 3 vòng lặp lồng nhau.

Cũng ứng dụng tư tưởng này để **xác định các thành phần liên thông của đồ thị**. Đây là thuật toán **Warshall**. Tư tưởng chính là:

- Nếu từ u đế k có đường đi, tức là A[u][k] = 1 và từ k đến v cũng có đường đi, tức là A[k][v] = 1 thì chắc chắn có đường đi từ u đến v ➔ A[u][v] = 1.

Cả 2 thuật toán đều có **độ phức tạp là : $O(n^3)$.**
Thuật toán được mô tả như sau:

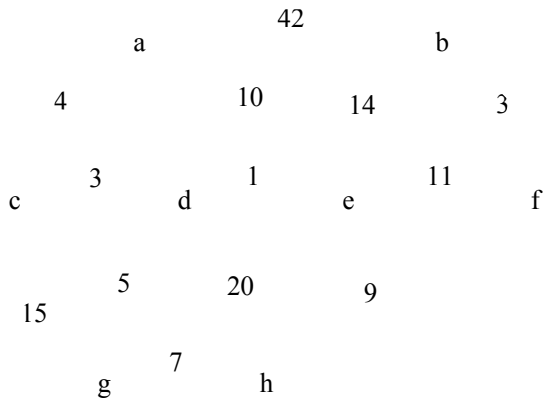
Procedure Floyd_Warshall	**Procedure Warshall**
Begin	**Begin**
For k = 1 to n do	For k = 1 to n do
For u = 1 to n do	For u = 1 to n do
For v = 1 to n do	For v = 1 to n do
If A[u][v] > A[u][k] + A[k][v] then	If A[u][k] = 1 và A[k][v] = 1 then
A[u][v] = A[u][k] + A[k][v];	A[u][v] = 1;
End;	**End;**

CHƯƠNG 4: CÂY

Bài 1. Tìm cây khung nhỏ nhất bằng thuật toán Prim của đồ thị gồm các đỉnh A, B, C, D, E, F, H, I được cho bởi ma trận trọng số sau.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ H \\ I \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & H & I \end{array} \\ \begin{pmatrix} \infty & 15 & 16 & 19 & 23 & 20 & 32 & 18 \\ 15 & \infty & 33 & 13 & 34 & 19 & 20 & 12 \\ 16 & 33 & \infty & 13 & 29 & 21 & 20 & 19 \\ 19 & 13 & 13 & \infty & 22 & 30 & 21 & 11 \\ 23 & 34 & 29 & 22 & \infty & 34 & 23 & 21 \\ 20 & 19 & 21 & 30 & 34 & \infty & 17 & 18 \\ 32 & 20 & 20 & 21 & 23 & 17 & \infty & 14 \\ 18 & 12 & 19 & 11 & 21 & 18 & 14 & \infty \end{pmatrix} \end{array}$$

Bài 2. Tìm cây khung nhỏ nhất của đồ thị sau theo thuật toán Kruskal và Prim.



Bài 3. Tìm cây khung nhỏ nhất bằng thuật toán Prim của đồ thị gồm các đỉnh A, B, C, D, E, F, H, I được cho bởi ma trận trọng số sau.

	A	B	C	D	E	F	G	H
A	∞	16	15	23	19	18	32	20
B	16	∞	13	33	24	20	19	11
C	15	13	∞	13	29	21	20	19
D	23	33	13	∞	22	30	21	12
E	19	24	29	22	∞	34	23	21
F	18	20	21	30	34	∞	17	14
G	32	19	20	21	23	17	∞	18
H	20	11	19	12	21	14	18	∞

Yêu cầu viết các kết quả trung gian trong từng bước lặp, kết quả cuối cùng cần đưa ra tập cạnh và độ dài của cây khung nhỏ nhất.

Bài thực hành số 4: Các thuật toán về cây

Bài tập: Cài đặt các thuật toán tìm cây khung nhỏ nhất.

Hướng dẫn:

Rõ ràng 1 đồ thị cho ta nhiều cây khung. Vấn đề là xác định cây khung nào trong đồ thị có trọng số sao cho tổng trọng số là bé nhất.

Thuật toán Prim – Dijsktra xác định cây khung bé nhất.

Tư tưởng của thuật toán này là dựa trên **thuật toán tìm đường đi tối ưu Dijsktra**.

- Dùng tập S để lưu trữ các đỉnh đã được duyệt. ban đầu S = Ø.
- Đối với **bài toán tìm đường đi ngắn nhất giữa 2 điểm u,v**, thì thuật toán Dijsktra sẽ lần lượt **chọn các đỉnh** trên đồ thị sao cho **giá trị khoảng cách từ đỉnh u đến đỉnh mới duyệt là nhỏ nhất có thể được**. Quá trình **dừng lại khi đỉnh v được duyệt**.
- Đối với **bài toán xác định cây khung nhỏ nhất**, thì quá trình duyệt các đỉnh cũng như trên, nhưng chỉ **dừng lại khi không còn duyệt được thêm đỉnh nào nữa**.

- Chú ý ở đây biến **D[v] không phải là khoảng cách đi từ đỉnh u đến v mà là khoảng cách từ v đến S (là khoảng cách ngắn nhất từ v đến 1 đỉnh trong tập S).**
  - Nếu giá trị khoảng cách tại đỉnh cuối cùng ≠ ∞, thì ta đã có 1 cây khung bé nhất.
  - Trong trường hợp ngược lại thì không xác định được cây khung bé nhất.

Thuật toán được mô tả như sau: độ phức tạp của thuật toán là $O(n^2)$.

```
Procedur Prim_Dijsktra;
Begin
    Chọn 1 đỉnh u là gốc của cây. D[u] = 0; D[v] = ∞ với v ≠ u; S = Ø
    while True do
    begin
        Tìm đỉnh k thoả mãn Free[k] = 1 và D[k] nhỏ nhất ;
        If  (không xác định được k  OR D[k] = ∞) then  Break.
        Else F :=  F U (T[k],k),  S = S U k ; Free[k] = 0
        For v ∈ V do
            If Free[v] = 1 và D[v]> A[k][v] then   // chú ý đk này!!!!!
            begin
                D[v]:= A[k][v];  T[v]:=k;
            end;
    end;
End;
```

4.1.4 Thuật toán Kruskal (thuật toán tham lam – ăn tham)

Tư tưởng của thuật toán như sau:

- Khởi tạo tập các cạnh của cây khung F = Ø.
- Chọn các cạnh của đồ thị theo trọng số từ nhỏ đến lớn, sao cho nó không tạo ra chu trình trong tập F.
- Đưa cạnh vừa chọn vào tập F và xóa nó khỏi tập cạnh E của đồ thị.
- Quá trình lặp lại cho đến khi trong tập F có đúng n-1 cạnh.

Thuật toán mô tả như sau:

```
Procedure Kruskal;
Begin
    F := Ø;  // khởi tạo tập rỗng
    While |F| < (n-1) and (E ≠ Ø) do // kiểm tra số phần tử của tập F
    Begin
        Chọn cạnh {e} sao cho trọng số trên {e} là nhỏ nhất.
        E:=E\{e};
        if (F U {e} không chứa chu trình) then F := T U { e} ;
    End;
    if (|T| < n-1) then
        Đồ thị không liên thông;
```

End;

Độ phức tạp của thuật toán là O(m*log$_2$m) với |E| = m.

Bài tập tổng hợp:

1. Cho ma trận kề (trọng số) của một đồ thị như sau

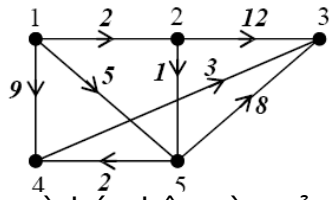
$$\begin{pmatrix} 0 & 3 & \infty & 3 & \infty & \infty & \infty & \infty \\ 3 & 0 & 2 & 6 & 2 & \infty & \infty & \infty \\ \infty & 2 & 0 & 3 & 4 & \infty & \infty & 3 \\ 3 & 6 & 3 & 0 & 4 & 8 & 2 & \infty \\ \infty & 2 & 4 & 4 & 0 & \infty & 3 & 10 \\ \infty & \infty & \infty & 8 & \infty & 0 & 5 & \infty \\ \infty & \infty & \infty & 2 & 3 & 5 & 0 & 4 \\ \infty & \infty & 3 & \infty & 10 & \infty & 4 & 0 \end{pmatrix}$$

   a. Vẽ đồ thị tương ứng
   b. Đồ thị có phải là đồ thị Euler (nửa Euler) hay không? Giải thích. Nếu có,tìm chu trình (đường đi) Euler trong đồ thị.
   c. Đồ thị có phải là đồ thị Hamilton hay không? Nếu có, tìm chu trình Hamilton trong đồ thị.
   d. Tìm đường đi ngắn nhất từ đỉnh số 1 đến đỉnh số 8.
   e. Tìm cây khung nhỏ nhất của đồ thị.
2. Cho ma trận kề (trọng số) của một đơn đồ thị vô hướng như sau:

$$\begin{pmatrix} 0 & 1 & \infty & \infty & \infty & 1 & \infty \\ 1 & 0 & 4 & \infty & \infty & 3 & 2 \\ \infty & 4 & 0 & 9 & 8 & \infty & 5 \\ \infty & \infty & 9 & 0 & 1 & \infty & \infty \\ \infty & \infty & 8 & 1 & 0 & 5 & 2 \\ 1 & 3 & \infty & \infty & 5 & 0 & 4 \\ \infty & 2 & 5 & \infty & 2 & 4 & 0 \end{pmatrix}$$

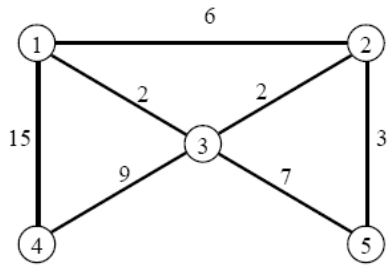
   a. Vẽ đồ thị, cho biết bậc của các đỉnh. Đây có phải là đồ thị Euler hay không?
   b. Tìm đường đi ngắn nhất từ đỉnh 1 đến các đỉnh còn lại bằng thuật toán Dijkstra.
   c. Tìm cây khung nhỏ nhất của đồ thị bằng thuật toán Kruskal.

17. Cho đồ thị như hình vẽ:



   a. Xác định bán bậc ra và bán bậc vào của các đỉnh của đồ thị.
   b. Đây có phải là đồ thị liên thông mạnh không? Tại sao?
   c. Xây dựng ma trận kề, trọng số của đồ thị.
   d. Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 3.

18. Cho đồ thị như hình vẽ:



a. Xác định bậc của các đỉnh của đồ thị, từ đó cho biết đây có là đồ thị Euler hay nửa Euler hay không?
b. Xây dựng ma trận kề trọng số của đồ thị
c. Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 5 trong đồ thị.
d. Tìm cây khung nhỏ nhất của đồ thị.

Bài tập nâng cao:

1. Bài toán truyền tin trên mạng.

   Một mạng máy tính gồm N máy đánh số từ 1 đến N, và M kênh truyền tin một chiều gữa một số cặp máy trong mạng được đánh số từ 1 đến M. Mạng máy tính là thông suốt, nghĩa là từ một máy có thể truyền tin đến một máy bất kỳ khác bằng các đường nối trực tiếp hoặc thông qua các máy trung gian. Một máy trong mạng là số chẵn nếu số kênh truyền tin trực tiếp từ nó đến các máy khác là chẵn. Một máy trong mạng là số lẻ nếu số kênh truyền tin trực tiếp từ nó đến các máy khác là lẻ. Giả sử s và t là hai máy lẻ trong mạng, hãy đổi hướng truyền tin của một số kênh để biến đổi mạng đã cho thành mạng (không nhất thiết phải thông suốt) mà trong nó hia máy s và t trở thành 2 máy chẵn và không làm thay đổi tính chẵn lẻ của các máy khác trong mạng. Số kênh đổi hướng càng ít càng tốt.

   Dữ liệu nhập vào từ file văn bản Net.inp:

Net.inp	Net.out
6 9	3
1 6	1
2 3	7
3 4	9
4 1	
4 6	
6 3	
2 5	
5 3	
5 6	

   - Dòng đầu tiên chứa 2 số N, M (N<2000, M<10000).
   - Dòng thứ 2 chứa 2 số s và t.
   - Dòng thứ I trong số M dòng tiếp theo chứa 2 số u_i và v_i cho biết kênh truyền tin thứ i truyền tin trực tiếp từ máy u_i đến v_i

- Kết quả ghi ra file Net.out

- Dòng đầu ghi số lượng kênh cần thay đổi hướng truyền tin (số q).
- Mỗi dòng trong q dòng tiếp theo ghi chỉ số kênh cần đảo ngược hướng truyền tin.

- Ví dụ: (xem bảng số liệu bên cạnh)

2. Bài toán thang máy

Một tòa nhà gồm N tầng, đánh số từ 1 đến N (0<N<201), chỉ có một thang máy phục vụ vận chuyển hàng từ tầng nọ đến tầng kia. Thang máy hiện đang ở tầng một. Trong một ngày có M yêu cầu vận chuyển hàng bằng thang máy (M<101), mỗi yêu cầu được mô tả bằng 2 số a và b cho biết cần vận chuyển hàng từ tầng a đến tầng b. Thang máy không thể phục vụ 2 yêu cầu cùng một lúc và chỉ có thể thực hiện yêu cầu này xong thì mới thực hiện tiếp yêu cầu khác. Cần lập kế hoạch cho thang máy thực hiện đủ M yêu cầu rồi trở về tầng một sao cho tổng quảng đường thang máy phải di chuyển là ít nhất.

Dữ liệu nhập vào từ file văn bản Tm.inp

- Dòng đầu ghi 2 số nguyên dương N và M.
- M dòng tiếp theo, dòng thứ I ghi 2 số a_i và b_i mô tả yêu cầu thứ i

Kết quả ghi ra file văn bản Tm.out

- Dòng đầu là tổng quãng đường tìm được
- Dòng thứ 2 mô tả thứ tự thực hiện các công việc bởi một hoán vị của 1, 2, …M.

3. Bài toán K đường đi ngắn nhất

Vùng đất X có N thành phố (3<N<800) được đánh số từ 1 đến N, giữa 2 thành phố có thể có đường nối trực tiếp hoặc không. Các con đường này được đánh số từ 1 đến M (0<M<4000). Ban lãnh đạo thể thao vùng X tổ chức cuộc chạy đua tiếp sức "thông minh" theo quy luật sau:

- Thành phố xuất phát là thành phố 1 và kết thúc là thành phố N.
- Mỗi đội thi đấu có K người dự thi. Khi người thứ nhất đến được thành phố N thì người thứ hai mới bắt đầu rời khỏi thành phố 1, khi người thứ 2 đến thành được thành phố N thì người thứ 3 mới rời khỏi thành phố 1, cứ như vậy cho đến khi người thứ K về tới đích thì được xem như thời điểm tính cho toàn đội.
- Đường chạy của các đội viên không được giống nhau hoàn toàn.
- Có thể chạy lại đoạn đường đã chạy.

Hãy viết chương trình tính thời gian nhỏ nhất để một đội hoàng thành cuộc chạy đua tiếp sức nếu trên nếu các vận động viên có tốc độ chạy như nhau.

Dữ liệu nhập vào từ file văn bản Kminpath.inp

Net.inp	Net.out
4 5 8 1 2 1 1 1 3 2 1 4 2 2 3 2 2 5 3 3 4 3 3 5 4 4 5 6	23 1 1325 135 12125 125

- Dòng đầu ghi 2 số nguyên dương K, N và M.
- M dòng tiếp theo, mỗi dòng chứa 3 số nguyên i, j, w thể hiện một đường đi trực tiếp giữa hai thành phố i và j mất thời gian chạy là w.

Kết quả ghi ra file văn bản Kminpath.out

- Dòng thứ nhất chứa một số nguyên duy nhất là thời gian chạy nhỏ nhất của 1 đội
- K dòng tiếp theo, mỗi dòng thể hiện hành trình chạy của một vận động viên trong đội là dãy các thành phố liên tiếp trên hành trình đó.

Ví dụ: (xem bảng bên)

4. Bài toán công chúa kén chồng

Magachip IV the Splendid, vua của Byteotia, có ý định chọn phò mã cho công chúa Ada của mình. Công chúa thì muốn chồng mình phải là người thông minh, không keo kiệt và không lãng phí quá. Nên nhà vua suy nghĩ mới chọn phương pháp thử tài bằng cách chọn lâu đài của mình làm nơi thi đấu. Lâu đài gồm nhiều phòng trưng bày đồ quí hiếm và nối với nhau bằng dãy các hành lang để thần dân có thể tham quan, phòng cuối cùng là phòng của công chúa, để thăm một phòng phải trả một đồng bytealer và sẽ xuất phát từ phòng đầu tiên. Nhà vua trao cho mỗi chàng trai đến cầu hôn một số đồng tiền chứa trong 1 cái túi và yêu cầu là mỗi người thăm 1 số phòng sao cho đến phòng cuối cùng thì tiêu hết số tiền đã cho, nếu đến phòng cuối cùng mà vẫn còn tiền thì phải thăm lại một số phòng

trưng bày nữa mới quay lại. Hãy lập trình giải quyết vấn đề trên để 1 chàng trai luôn có thể thực hiện được yêu cầu của nhà vua.

Zam.inp
5 6 3 4 9 1 2 3 4 5 2 4 5 4 1 5 1 2 2 3 3 1
Zam.out
3 2 4

Dữ liệu nhập từ file Zam.inp miêu tả lâu đài, số hiệu phòng công chúa đang ở, tổng số tiền trong túi:

- Dòng đầu tiên có 5 số nguyên dương n(số lượng phòng), m(số hành lang), e(số hiệu phòng xuất phát), p(số hiệu phòng công chúa), b(tổng số tiền vua ban cho).
- Dòng 2 có n số nguyên dương c_i, mỗi số là chi phí mỗi lần vào thăm trong phòng i.
- Trong m dòng tiếp theo có từng cặp số nguyên dương (x, y), mỗi cặp nối biểu thị một hành lang nối phòng x với phòng y.

Tính ra dãy các phòng đi qua đến phòng của công chúa và lưu hành trình vào file Zam.out

Ví dụ: (như bảng dữ liệu bên cạnh)

5. Bài toán chia cắt địch

Cuối năm 1944, quân Liên Xô phản công vào vùng X, nơi có N thành phố bị phát xít Đức chiếm đóng. Dọc theo các con đường giữa 2 thành phố đều có quân Đức canh giữ. Bộ tham mưu quân sự của Liên Xô mới chỉ đạo phản công tiêu diệt địch, bước đầu thực hiện chia cắt quân địch thành 2 vùng tách biệt để chúng không liên lạc được với nhau, nhưng cần tính toán tiêu diệt như thế nào là lợi nhất mà chỉ cần huy động lực lượng ít nhất để giành thắng lợi. Biết rằng đi tiêu diệt quân Đức thì Liên Xô phải bố trí số quân ít nhất bằng với số quân địch chiếm đóng. Hãy viết chương trình giúp bộ tham mưu quân Liên Xô chỉ cần điều ít nhất lực lượng để giành chiến thắng.

Chiacat.inp	Chiacat.out
10 18 1 2 8 1 4 4 1 7 1 2 3 5 2 4 5 2 9 9 3 4 4 3 6 2 3 9 7 4 5 5 5 6 6 5 7 5 5 8 3 6 8 7 6 10 1 7 8 5 8 10 1 9 10 9	10 1 7 3 6 4 5 6 10 8 10

Dữ liệu nhập vào từ file Chiacat.inp

- Dòng đầu tiên là 2 số nguyên dương N (số thành phố tối đa 100) và M(số con đường nối các cặp 2 thành phố trong vùng X).
- M dòng tiếp theo, mỗi dòng có 3 số nguyên dương i, j và w thể hiện trên con đường nối thành phố i với thành phố j hiện có w lính Đức.

Kết quả ghi ra file Chiacat.out

- Dòng đầu là số lượng quân Liên Xô cần điều động
- Dòng sau, mỗi dòng có 2 số u và v thể hiện con đường (u, v) mà quân Liên Xô cần chiếm lại trong đợt phản công đầu tiên này.

Ví dụ: (xem bảng bên)

Viết tiểu luận

ĐỀ TÀI 1:

SỬ DỤNG PHƯƠNG PHÁP ĐỒ THỊ ĐỂ THỂ HIỆN VIỆC BỐ TRÍ LỊCH THI CHO SINH VIÊN KHOA TOÁN – TIN HỌC.

ĐẶC TẢ ĐỀ TÀI :

Sử dụng phương pháp đồ thị để thể hiện việc bố trí lịch thi cho sinh viên khoa Toán – Tin học 7 môn thi trong 7 ngày.

Yêu cầu phải bố trí lịch thi sao cho hai môn thi của cùng một giáo viên không được rơi vào hai ngày liên tiếp nhau.

Biết rằng không có giáo viên nào có nhiều hơn 5 môn thi.

YÊU CẦU CỦA ĐỀ TÀI

Về lý thuyết : Tìm hiểu và trình bày các khái niện cơ bản về :

- Đồ thị và các khái niệm cơ bản về đồ thị có hướng, đồ thị vô hướng.
- Các thủ tục (hàm) có liên quan đến giao diện của màn hình đồ họa.
- Đường đi và chu trình Hamilton.
- Thiết lập thuật toán của đề tài và minh họa kết quả bằng đồ thị Hamilton.

Về lập trình:

- Viết chương trình dựa vào thuật toán đã thiết lập.
- Giao diện thân thiện với người sử dụng.
- Kết quả cho ra là một đồ thị với màu sắc phân biệt.

MÔI TRƯỜNG CÀI ĐẶT

Ngôn ngữ sử dụng : C, C++, Visual C++, Visual Basic

ĐỀ TÀI 2:

SỬ DỤNG PHƯƠNG PHÁP ĐỒ THỊ ĐỂ GIẢI BÀI TOÁN DÂN GIAN (BÀI TOÁN 1).

ĐẶC TẢ ĐỀ TÀI :

Có một vị khách đến xin nhà vua ban cho một quả cam trong vườn ngự uyển. Nhà vua chấp thuận. Ông ta đến vườn mới hay phải qua 3 cổng lính canh.

Đến cổng thứ nhất, người lính canh bảo vị khách: " Vua ban cho thì anh cứ vào mà hái, nhưng lúc ra phải đưa cho ta một nữa số cam và thêm một trái".

Qua cổng thứ hai và thứ ba, hai lính canh cũng nói với anh như người lính canh thứ nhất. Vị khách phải hái bao nhiêu quả cam để lúc ra khỏi vườn còn được một quả trong tay?

YÊU CẦU CỦA ĐỀ TÀI

Về lý thuyết : Tìm hiểu và trình bày các khái niện cơ bản về :

- Các thủ tục (hàm) có liên quan đến giao diện của màn hình đồ họa.
- Đồ thị và các khái niệm cơ bản về đồ thị có hướng, đồ thị vô hướng.
- Minh họa bài toán bằng đồ thị.
- Thiết lập thuật toán.
- Có thể mở rộng bài toán bằng cách cho người sử dụng thay đổi kết quả số lượng cam mà vị khách có được sau khi ra khỏi vườn.

Về lập trình:

- Viết chương trình dựa vào thuật toán đã thiết lập.
- Giao diện thân thiện với người sử dụng.
- Kết quả cho ra là một đồ thị với màu sắc phân biệt.

MÔI TRƯỜNG CÀI ĐẶT

Ngôn ngữ sử dụng : C, C++, Visual C++, Visual Basic

ĐỀ TÀI 3:

SỬ DỤNG PHƯƠNG PHÁP ĐỒ THỊ ĐỂ GIẢI BÀI TOÁN DÂN GIAN (BÀI TOÁN 2).

ĐẶC TẢ ĐỀ TÀI :

Có hai cha con người nông dân đi mua vé tàu hỏa. Người bán vé tàu hỏi: " Chú bé này bao nhiêu tuổi ?". Ông cha trả lời: "Con trai tôi tuổi gấp 5 lần em gái nó, mẹ nó tuổi gấp 6 lần tuổi nó. Tuổi tôi thì bằng tuổi của vợ và hai con tôi cộng lại. Còn mẹ tôi thì bằng tuổi của tất cả gia đình chúng tôi cộng lại". Người bán vé tàu nói: "Thôi đủ rồi! Con ông được miễn vé".

Dựa vào đâu mà người vé tàu miễn vé cho chú bé? Biết rằng, theo luật đường sắt thì trẻ em dưới 6 tuổi đi cùng người lớn sẽ được miễn vé.

YÊU CẦU CỦA ĐỀ TÀI

Về lý thuyết : Tìm hiểu và trình bày các khái niện cơ bản về :

- Các thủ tục (hàm) có liên quan đến giao diện của màn hình đồ họa.
- Đồ thị và các khái niệm cơ bản về đồ thị có hướng, đồ thị vô hướng.
- Minh họa bài toán bằng đồ thị.
- Thiết lập thuật toán.

Về lập trình:

- Viết chương trình dựa vào thuật toán đã thiết lập.
- Giao diện thân thiện với người sử dụng.
- Kết quả cho ra là một đồ thị với màu sắc phân biệt.

MÔI TRƯỜNG CÀI ĐẶT

Ngôn ngữ sử dụng : C, C++, Visual C++, Visual Basic

ĐỀ TÀI 4:

CÁC THUẬT TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN ĐỒ THỊ

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để cài đặt chương trình cho phép biểu diễn đồ thị, kiểm tra tính liên thông và tìm đường đi ngắn nhất giữa 2 đỉnh cho trước bằng giải thuật Dijkstra, Ford-Bellman trên đồ thị vô hướng.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**
  - Các thao tác cơ bản về đồ họa.
  - Các khái niệm về đồ thị có hướng và đồ thị vô hướng
  - Các cách biểu diễn đồ thị, các phương pháp tìm kiếm trên đồ thị (tìm theo chiều rộng và chiều sâu) và tính liên thông.
  - Các giải thuật có liên quan như: kiểm tra tính liên thông, tìm đường đi ngắn nhất.
  - Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.
- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về đồ thị.
- Biểu diễn đồ thị trên màn hình.
- Kiểm tra tính liên thông.
- Cho phép tìm đường đi ngắn nhất giữa 2 đỉnh bất kỳ.

MÔI TRƯỜNG CÀI ĐẶT :

Ngôn ngữ lập trình sử dụng: C hay C $^{++}$

ĐỀ TÀI 5: CÁC GIẢI THUẬT TÌM CÂY PHỦ TỐI TIỂU

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để cài đặt chương trình cho phép biểu diễn đồ thị, kiểm tra tính liên thông và tìm cây có trọng lượng nhỏ nhất bằng giải thuật Kruscal.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**

- Các thao tác cơ bản về đồ họa.
- Các khái niệm về đồ thị có hướng và đồ thị vô hướng
- Các cách biểu diễn đồ thị, các phương pháp tìm kiếm trên đồ thị (tìm theo chiều rộng và chiều sâu) và tính liên thông.
- Các giải thuật có liên quan như: kiểm tra tính liên thông, giải thuật kiểm tra tính liên thông và giải thuật Kruscal tìm cây có trọng lượng nhỏ nhất.
- Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.

- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về đồ thị.
- Biểu diễn đồ thị trên màn hình.
- Kiểm tra tính liên thông.
- Cho phép tìm cây có trọng lượng nhỏ nhất.

MÔI TRƯỜNG CÀI ĐẶT :

Ngôn ngữ lập trình sử dụng: C hay C $^{++}$

ĐỀ TÀI 6: BÀI TOÁN TỔ CHỨC THI CÔNG

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để cài đặt chương trình cho phép biểu diễn đồ thị, biểu diễn đồ thị sau khi xếp hạng, xác định các thời điểm sớm nhất, trễ nhất của từng công việc, thời gian hoàn thành công trình và vẽ sơ đồ GANT thể hiện kế hoạch hoàn thành công trình.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**
  - Các thao tác cơ bản về đồ họa.
  - Các khái niệm về đồ thị có hướng và đồ thị vô hướng
  - Các cách biểu diễn đồ thị, Các phép biểu diễn đồ thị.
  - Giải thuật xếp hạng trên đồ thị, giải thuật xác định các thời gian sớm nhất và thời gian trễ nhất.
  - Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.
- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về bài toán tổ chức thi công.
- Biểu diễn đồ thị trước và sau khi xếp hạng lên màn hình.
- Xác định các thời điểm sớm nhất, trễ nhất của từng công việc, thời gian hoàn thành công trình
- Vẽ sơ đồ GANT

MÔI TRƯỜNG CÀI ĐẶT :

Ngôn ngữ lập trình sử dụng: C hay C $^{++}$

ĐỀ TÀI 7: BÀI TOÁN QUẢN LÝ DỰ ÁN

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để cài đặt chương trình cho phép biểu diễn đồ thị, biểu diễn đồ thị sau khi xếp hạng, xác định các thời điểm sớm nhất, trễ nhất của từng công việc, thời gian hoàn thành công trình và vẽ sơ đồ GANT thể hiện kế hoạch hoàn thành công trình.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**
  - Các thao tác cơ bản về đồ họa.
  - Các khái niệm về đồ thị có hướng và đồ thị vô hướng
  - Các cách biểu diễn đồ thị, Các phép biểu diễn đồ thị.
  - Giải thuật xếp hạng trên đồ thị, giải thuật xác định các thời gian sớm nhất và thời gian trễ nhất.
  - Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.
- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về bài toán tổ chức thi công.
- Biểu diễn đồ thị trước và sau khi xếp hạng lên màn hình.
- Xác định các thời điểm sớm nhất, trễ nhất của từng công việc, thời gian hoàn thành công trình
- Vẽ sơ đồ GANT

MÔI TRƯỜNG CÀI ĐẶT :

Sử dụng: MS . PROJECT 2003

ĐỀ TÀI 8: BÀI TOÁN "8 QUÂN HẬU"

ĐẶC TẢ ĐỀ TÀI :

Bài toán : Đặt 8 quân hậu trên bàn cờ vua 8×8 sao cho không có quân hậu nào có thể tấn công được con khác (theo luật chơi cờ vua), nghĩa là phải đặt các quân hậu sao cho không có hàng, cột hoặc đường chéo nào trên bàn cờ có hơn 1 quân hậu. Chẳng hạn, một cách đặt quân hậu đúng như sau :

			Q				
	Q						
							Q
					Q		
Q							
		Q					
				Q			
						Q	

YÊU CẦU CỦA ĐỀ TÀI :

Về lý thuyết :
- Nắm vững lý thuyết cơ bản về cấu trúc dữ liệu và giải thuật
- Giải thuật tìm kiếm sâu kết hợp quay lui (Backtracking)

Về lập trình:
- Cài đặt cấu trúc dữ liệu tổ chức bàn cờ.
- Cài đặt thuật toán tìm kiếm sâu kết hợp quay lui theo nguyên tắc : Trước tiên, đặt quân hậu vào ô thứ nhất của cột 1, rõ ràng tất cả các ô của cột đó đã bị khống chế nên không thể đặt quân hậu khác. Đặt tiếp một quân hậu vào cột thứ hai, hai ô đầu của cột đó đã bị cấm bởi quân hậu thứ nhất, do vậy ta đặt quân hậu vào ô thứ ba. Tiếp tục với cột thứ ba, ô đầu tiên có thể đặt quân hậu của cột này là ô thứ năm … Tiếp tục với các cột còn lại trên bàn cờ cho đến khi tìm được một lời giải đúng.
- Hiển thị bàn cờ sau mỗi nước đi.
- Dịch chương trình sang file thực thi.

Ngôn ngữ lập trình sử dụng : C, C $^{++}$, Visual C^{++}

ĐỀ TÀI 9: BÀI TOÁN "QUÂN MÃ ĐI TUẦN"

ĐẶC TẢ ĐỀ TÀI :

Bài toán : Trên bàn cờ vua 8 × 8, một quân mã được phép đi theo luật cờ vua. Vị trí đầu tiên của quân mã đặt tại một ô nào đó. Hãy tìm cách di chuyển quân mã qua tất cả các ô của bàn cờ sao cho mỗi ô chỉ được đi qua 1 lần duy nhất. Chẳng hạn 10 vị trí hợp lệ đầu tiên cho quân mã nếu quân mã bắt đầu khởi hành tại ô (1, 1) trên bàn cờ vua như sau :

1	**4**				**6**		
		2	**5**				**7**
3							
						8	
							9
				...			
						10	

YÊU CẦU CỦA ĐỀ TÀI :

Về lý thuyết :
- Nắm vững lý thuyết cơ bản về cấu trúc dữ liệu và giải thuật.
- Thuật toán đệ qui

Về chương trình:
- Cài đặt cấu trúc dữ liệu tổ chức bàn cờ.
- Khởi tạo ngẫu nhiên vị trí đặt quân mã đầu tiên.
- Cài đặt chương trình máy tính đệ qui theo kiểu thử sai, vét cạn mọi khả năng để tìm lời giải: tìm kiếm nước đi kế tiếp bằng cách chọn một trong những ô có thể đặt quân mã hợp lệ tiếp theo trên bàn cờ. Cứ tiếp tục cho những nước sau đó đến khi tìm thấy một lời giải.
- Hiển thị bàn cờ sau mỗi nước đi.
- Dịch chương trình sang file thực thi.

Ngôn ngữ lập trình sử dụng : C, C $^{++}$, Visual C^{++}

ĐỀ TÀI 10:

BÀI TOÁN PHÂN BỐ KÊNH TRUYỀN HÌNH TẠI ĐBSCL

DÙNG THUẬT TOÁN TÔ MÀU ĐỒ THỊ

ĐẶC TẢ ĐỀ TÀI :

Bài toán (Phân bố kênh truyền hình đồng bằng Sông Cửu long): Các kênh truyền hình từ số 2 đến số 13 được phân chia cho các đài truyền hình các tỉnh trong vùng đồng bằng Sông Cửu long (Cà Mau, Bạc Liêu, Sóc Trăng, Cần thơ, Vĩnh long, An giang, Kiên giang, Đồng tháp, Trà Vinh, Bến Tre, Tiền giang, Long An) sao cho không có hai đài phát nào ở hai tỉnh nằm cạnh nhau trên bản đồ địa lý lại dùng cùng một kênh ?

YÊU CẦU CỦA ĐỀ TÀI :

Về lý thuyết:

- Nắm vững lý thuyết cơ bản về cấu trúc dữ liệu và giải thuật
- Giải thuật tô màu đồ thị (giáo trình *Toán rời rạc*, Nguyễn Đức Nghĩa, NXB ĐH Quốc Gia TP. Hồ Chí Minh)

Về chương trình :

- Dữ liệu nhập vào và xuất ra dưới dạng file chứa các thông tin về đồ thị.
- Tổ chức đồ thị: Mỗi đài phát tương ứng 1 đỉnh, mỗi màu biểu thị 1 kênh. Việc phân chia kênh tương ứng với việc tô màu đồ thị (tô màu theo thuật toán Greedy xét lần lượt theo số thứ tự các đỉnh)
- Cài đặt quá trình thực hiện giải thuật tô màu (hiển thị *Tên tỉnh* và *Số kênh* tương ứng)
- Dịch chương trình sang file thực thi.

Ngôn ngữ lập trình sử dụng : C, C $^{++}$, Visual C^{++}

ĐỀ TÀI 11:

BÀI TOÁN PHÂN BỐ KÊNH TRUYỀN HÌNH TẠI MIỀN ĐÔNG NAM BỘ

DÙNG THUẬT TOÁN TÔ MÀU ĐỒ THỊ

ĐẶC TẢ ĐỀ TÀI :

Bài toán (Phân bố kênh truyền hình miền đông Nam Bộ): Các kênh truyền hình từ số 15 đến số 25 được phân chia cho các đài truyền hình các tỉnh trong vùng miền đông Nam Bộ (TP.Hồ Chí Minh, Đồng Nai, Bà Rịa – Vũng Tàu, Tây Ninh, Bình Dương, Bình Phước) sao cho không có hai đài phát nào ở hai tỉnh nằm cạnh nhau trên bản đồ địa lý lại dùng cùng một kênh ?

YÊU CẦU CỦA ĐỀ TÀI :

Về lý thuyết:

- Nắm vững lý thuyết cơ bản về cấu trúc dữ liệu và giải thuật
- Giải thuật tô màu đồ thị (giáo trình *Toán rời rạc*, Nguyễn Đức Nghĩa, NXB ĐH Quốc Gia TP. Hồ Chí Minh)

Về chương trình :

- Dữ liệu nhập vào và xuất ra dưới dạng file chứa các thông tin về đồ thị.

- Tổ chức đồ thị: Mỗi đài phát tương ứng 1 đỉnh, mỗi màu biểu thị 1 kênh. Việc phân chia kênh tương ứng với việc tô màu đồ thị (tô màu theo thuật toán Greedy xét lần lượt theo số thứ tự các đỉnh)
- Cài đặt quá trình thực hiện giải thuật tô màu (hiển thị *Tên tỉnh* và *Số kênh* tương ứng)
- Dịch chương trình sang file thực thi.

Ngôn ngữ lập trình sử dụng : C, C ++ , Visual C++

ĐỀ TÀI 12:

XÂY DỰNG BẢN ĐỒ TRỰC TUYẾN NHỮNG CON ĐƯỜNG LỚN TRONG NỘI THÀNH TP.HỒ CHÍ MINH VỚI VIỆC TÍNH CHI PHÍ THẤP NHẤT ĐỂ DI CHUYỂN GIỮA TRUNG TÂM CÁC QUẬN.

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để cài đặt chương trình cho phép xác định con đường đi trên đồ thị, kiểm tra tính liên thông và tìm đường đi ngắn nhất giữa 2 đỉnh , mỗi đỉnh là 1 địa danh cụ thể tại trung tâm Tp.Hồ Chí Minh bằng giải thuật Dijkstra, Ford-Bellman trên đồ thị vô hướng.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**
  - Các thao tác cơ bản về đồ họa.
  - Các khái niệm về đồ thị có hướng và đồ thị vô hướng
  - Các cách biểu diễn đồ thị, các phương pháp tìm kiếm trên đồ thị (tìm theo chiều rộng và chiều sâu) và tính liên thông.
  - Các giải thuật có liên quan như: kiểm tra tính liên thông, tìm đường đi ngắn nhất.
  - Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.
- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về đồ thị.
- Biểu diễn đồ thị trên màn hình.
- Kiểm tra tính liên thông.
- Cho phép tìm đường đi ngắn nhất giữa 2 đỉnh bất kỳ.

MÔI TRƯỜNG CÀI ĐẶT :

Ngôn ngữ lập trình sử dụng: C hay C ++

ĐỀ TÀI 13:

XÂY DỰNG HỆ THỐNG KẾT NỐI MẠNG CỦA CÁC TRƯỜNG ĐẠI HỌC TRONG THÀNH PHỐ VỚI CHI PHÍ THẤP NHẤT

ĐẶC TẢ ĐỀ TÀI :

Vận dụng các lý thuyết cơ bản về đồ thị để thiết kế ra đồ thị thực tế kết nối các trường đại học, xây dựng chương trình cho phép biểu diễn đồ thị, kiểm tra tính liên thông và tìm cây có trọng lượng nhỏ nhất bằng giải thuật Kruscal hoặc Prim.

YÊU CẦU CỦA ĐỀ TÀI :

- **Lý thuyết:**

- Các thao tác cơ bản về đồ họa.
- Các khái niệm về đồ thị có hướng và đồ thị vô hướng
- Các cách biểu diễn đồ thị, các phương pháp tìm kiếm trên đồ thị (tìm theo chiều rộng và chiều sâu) và tính liên thông.
- Các giải thuật có liên quan như: kiểm tra tính liên thông, giải thuật kiểm tra tính liên thông và giải thuật Kruscal tìm cây có trọng lượng nhỏ nhất.
- Những cấu trúc dữ liệu cần thiết để cài đặt chương trình.

- **Chương trình:**

Phải có những chức năng cơ bản sau:

- Cập nhật dữ liệu về đồ thị.
- Biểu diễn đồ thị trên màn hình.
- Kiểm tra tính liên thông.
- Cho phép tìm cây có trọng lượng nhỏ nhất.

MÔI TRƯỜNG CÀI ĐẶT :

Ngôn ngữ lập trình sử dụng: C hay C $^{++}$

ĐỀ TÀI 14: BÀI TOÁN ĐƯỜNG ĐI NGƯỜI GIAO HÀNG

ĐẶC TẢ ĐỀ TÀI

Nội dung bài toán:

Xét bài toán rất nổi tiếng có tên là bài toán tìm đường đi của người giao hàng (TSP - Traveling Salesman Problem): Có một người giao hàng cần đi giao hàng tại n thành phố. Xuất phát từ một thành phố nào đó, đi qua các thành phố khác để giao hàng và trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần, khoảng cách từ một thành phố đến các thành phố khác là xác định được. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất.

YÊU CẦU CỦA ĐỀ TÀI

Nắm vững cơ sở lý thuyết về cấu trúc dữ liệu. Các kỹ thuật thiết kế giải thuật.

Chương trình cần có các chức năng sau: Cho phép nhập vào bài toán: số thành phố, khoảng cách giữa các thành phố (có thể lấy số liệu từ trong tập tin). Xuất ra phương án tìm được. Nếu thể hiện dưới dạng đồ hoạ càng tốt.

Mô phỏng thực tế của các tỉnh ĐBSCL.

MÔT TRƯỜNG CÀI ĐẶT

Ngôn ngữ lập trình sử dụng: C, C++ , Visual C++

ĐỀ TÀI 15: BÀI TOÁN ĐƯỜNG ĐI NGƯỜI ĐƯA THƯ

ĐẶC TẢ ĐỀ TÀI

Nội dung bài toán:

Xét bài toán về người đưa thư, xuất phát từ Bưu điện Tp. Hồ Chí Minh, anh ta đi đến tất các bưu điện chính trong nội ô, mỗi nơi đúng 1 lần. Cuối cùng anh ta quay trở lại nơi xuất phát sao cho quảng đường mà anh ta đi qua là ngắn nhất. Hãy tìm một chu trình (một đường đi khép kín thỏa mãn điều kiện trên) sao cho tổng độ dài các cạnh là nhỏ nhất.

YÊU CẦU CỦA ĐỀ TÀI

Nắm vững cơ sở lý thuyết về cấu trúc dữ liệu. Các kỹ thuật thiết kế giải thuật.

Chương trình cần có các chức năng sau: Cho nhập vào bài toán: số bưu điện chính tại các quận, khoảng cách nối giữa các bưu điện (có thể lấy số liệu từ trong bản đồ hoặc khảo sát thực tế). Xuất ra phương án tìm được. Nếu thể hiện dưới dạng đồ hoạ càng tốt.

MÔT TRƯỜNG CÀI ĐẶT

Ngôn ngữ lập trình sử dụng: C, C++ , Visual C++