CHƯƠNG 7

BÀI TOÁN LUỒNG CỰC ĐẠI TRONG MẠNG

(tuần 12,13: Tổng cộng có 4 tiết lý thuyết và 4 tiết hướng dẫn bài tập/thực hành)

Bài toán luồng cực đại trong mạng là một trong số bài toán tối ưu trên đồ thị tìm được những ứng dụng rộng rãi trong thực tế cũng như những ứng dụng thú vị trong lý thuyết tổ hợp. Bài toán được đề xuất vào đầu năm 1950, và gắn liên với tên tuổi của hai nhà toán học Mỹ là Ford và Fulkerson. Trong chương này chúng ra sẽ trình bày thuật toán Ford và Fulkerson để giải bài toán đặt ra và nêu một sô ứng dụng của bài toán.

1. MẠNG. LUỒNG TRONG MẠNG. BÀI TOÁN LUỒNG CỰC ĐẠI

Định nghĩa 1.

Ta gọi mạng là đồ thị có hướng G=(V,E), trong đó duy nhất một đỉnh s không có cung đi vào gọi là đỉnh phát, duy nhất một đỉnh t không có cung đi ra gọi là điểm thu và mỗi cung $e=(v,w) \in E$ được gán với một số không âm $c(e) = c(v,w)$ gọi là khả năng thông qua của cung e.

Để thuận tiện cho việc trình bày ta sẽ qui ước rằng nếu không có cung (v,w) thì khả năng thông qua c(v,w) được gán bằng 0.

Định nghĩa 2.

Giả sử cho mạng G=(V,E). Ta gọi luồng f trong mạng G=(V,E) ;là ánh xạ $f: E \rightarrow R_+$ gán cho mỗi cung $e=(v,w) \in E$ một số thực không âm f(e)=f(v,w), gọi là luồng trên cung e, thoả mãn các điều kiện sau:

i)Luồng trên cung $e \in E$ không vượt quá khả năng thông qua của nó:

$$0 \le f(e) \le c(e),$$

ii)Điều kiện cân bằng luồng trên mỗi đỉnh của mạng: Tổng luồng trên các cung đi vào đỉnh v bằng tổng luồng trên các cung đi ra khỏi đỉnh v, nếu v#s, t:

$$Div_f(v) = \sum_{w \in \Gamma^-(v)} f(w,v) - \sum_{w \in \Gamma^+(v)} f(v,w) = 0$$

trong đo $\Gamma^-(v)$ – tập các đỉnh của mạng mà từ đó có cung đến v, $\Gamma^+(v)$ - tập các đỉnh của mạng mà từ v có cung đến nó:

$$\Gamma^-(v) = \{ w \in V : (w,v) \in E\},$$

$$\Gamma^+(v) = \{ w \in V : (v,w) \in E\}.$$

i)))Giá trị của luồng f là số

$$Val(f) = \sum f(s,w) = \sum f(w,t).$$

$$w \in \Gamma^+(s) \qquad w \in \Gamma^-(t)$$

Bài toán luồng cực đại trong mạng:

Cho mạng G(V,E). Hãy tìm luồng f* trong mạng với giá trị luồng val(f*) là lớn nhất. Luồng như vậy ta sẽ gọi là luồng cực đại trong mạng.

Bài toán như vậy có thể xuất hiện trong rất nhiều ứng dụng thực tế. Chẳng hạn khi cần xác định cường độ lớn nhất của dòng vận tải giữa hai nút của một bản đồ giao thông. Trong ví dụ này lời giải của bài toán luồng cực đại sẽ chỉ cho ta các đoạn đường đông xe nhất và chúng tạo thành "chỗ hẹp" tương ứng với dòng giao thông xét theo hai nút được chọn. Một ví dụ khác là nếu xét đồ thị tương ứng với một hệ thống đường ống dẫn dầu. Trong đó các ống tương ứng với các cung, điểm phát có thể coi là tầu chở dầu, điểm thu là bể chứa, còn những điểm nối giữa các ống là các nút của đồ thị. Khả năng thông qua của các cung tương ứng với tiết diện của các ống. Cần phải tìm luồng dầu lớn nhất có thể bơm từ tàu chở dầu vào bể chứa.

2. LÁT CẮT. ĐƯỜNG TĂNG LUỒNG. ĐỊNH LÝ FORD_FULKERSON

Định nghĩa 3.

Ta gọi lát cắt (X,X^*) là một cách phân hoạch tập đỉnh V của mạng ra thành hai tập X và $X^* = V\backslash X$, trong đó $s \in X$, $t \in X^*$. Khả năng thông qua của lát cắt (X,X^*) là số

$$c(X,X^*) = \sum_{\substack{v \in X \\ w \in X^*}} c(v,w)$$

Lát cắt với khả năng thông qua nhỏ nhất được gọi là lát cắt hẹp nhất.

Bổ đề 1.

Giá trị của luồng f trong mạng luôn nhỏ hơn hoặc bằng khả năng thông qua của lát cắt (X,X^*) bất kỳ trong nó: $val(f) \le c(X,X^*)$.

Hệ quả 1.

Giá trị luồng cực đại trong mạng không vượt quá khả năng thông qua của lát cắt hẹp nhất trong mạng.

Ford và Fulkerson đã chứng minh rằng giá trị luồng cực đại trong mạng đúng bằng khả năng thông qua của lát cắt hẹp nhất. Để có thể phát biểu và chứng minh kết quả này chúng ra sẽ cần thêm một số khái niệm.

Giả sử f là một luồng trong mạng G = (V,E). Từ mạng G =(V,E) ta xây dựng đồ thị có trọng số trên cung Gf = (V, E_f), với tập cung E_f và trọng số trên các cung được xác định theo qui tắc sau:

-Nếu e=(v,w) ∈ E với f(v,w) =0, thì (v,w) ∈ E_f với trọng số c(v,w);

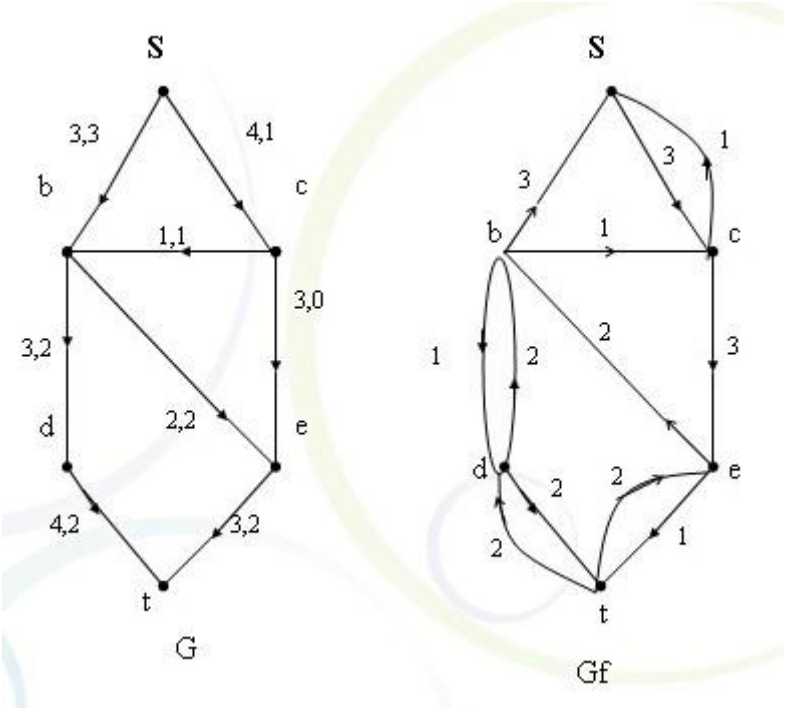
-Nếu e=(v,w) ∈ E với f(v,w) =c(v,w), thì (w,v) ∈ E_f với trọng số f(v,w);

-Nếu e=(v,w) ∈ E với 0<f(v,w)<c(v,w), thì (v,w) ∈ E_f với trọng số c(v,w)-f (v,w) và (w,v) ∈ E_f với trọng số f(v,w).

Các cung của G_f đồng thời cũng là cung của G được gọi là cung thuận, các cung còn lại được gọi là cung nghịch. Đồ thị G_f được gọi là đồ thị tăng luồng.

Thí dụ:

Các số viết cạnh các cung của G ở hình 1 theo thứ tự là khả năng thông qua và luồng trên cung.



Hình 1. Mạng G và luồng f. Đồ thị có trọng số G_f tương ứng

Giả sử P=(s=v_0, v_1, . . . , v_k=t) là một đường đi từ s đến t trên đồ thị tăng luồng Gf. Gọi δ là giá trị nhỏ nhất của các trọng số của các cung trên đường đi P. Xây dựng luồng f' trên mạng theo qui tắc (3 trường hợp) sau:

$$f'(u,v) = \begin{cases} f(u,v) + \delta, & \text{nếu } (u,v) \in P \text{ là cung thuận} \\ f(u,v) - \delta, & \text{nếu } (v,u) \in P \text{ là cung nghịch} \end{cases}$$

f(u,v), nếu (u,v) ∉ P

Dễ dàng kiểm tra được rằng f' được xây dựng như trên là luồng trong mạng và val(f') = val(f) + δ . Ta sẽ gọi thủ tục biến đổi luồng vừa nêu là tăng luồng dọc theo đường P.

Định nghĩa 4.

Ta gọi đường tăng luồng f là mọi đường đi từ s đến t trên đồ thị tăng luồng G(f).

Định lý 1.

Các mệnh đề dưới đây là tương đương:

(i) f là luồng cực đại trong mạng:

(ii) không tìm được đường tăng luồng f;

(iii) val(f)=c(X,X^*) với một lát cắt (X,X^*) nào đó.

3. THUẬT TOÁN TÌM LUỒNG CỰC ĐẠI

Định lý 1 là cơ sở để xây dựng thuật toán lặp sau đây để tìm luồng cực đại trong mạng: Bắt đầu từ luồng với luồng trên tất cả các cung bằng 0 (ta sẽ gọi luồng như vậy là luồng không), và lặp lại bước lặp sau đây cho đến khi thu được luồng mà đối với nó không còn đường tăng:

Bước lặp tăng luồng (Ford-Fulkerson): Tìm đường tăng P đối với luồng hiện có. Tăng luồng dọc theo đường P.

Khi đã có luồng cực đại, lát cắt hẹp nhất có thể theo thủ tục mô tả trong chứng minh định lý 1. Sơ đồ của thuật toán Ford-Fulkerson có thể mô tả trong thủ tục sau đây:

```
void Max_Flow()
(* Thuật toán Ford-Fulkerson *)
        (* Khởi tạo: Bắt đầu từ luồng với giá trị 0 *)
        for u ∈ V
        for v ∈ V f(u,v)=0;
        stop=0;
        while (!stop)
        if <Tìm được đường tăng luồng P>
                <Tăng luồng dọc theo P>
        else stop=1;
}
```

Để tăng luồng trong G_f có thể tìm kiếm thuật toán theo chiều rộng (hay tìm kiếm theo chiều sâu) bắt đầu từ đỉnh s, trong đó không cần xây dựng đồ thị tường minh G_f. Ford_Fulkerson đề nghị thuật toán gán nhãn chi tiết sau đây để giải bài toán luồng cực đại

trong mạng (có thể bắt đầu từ luồng không), sau đó ta sẽ tăng luồng bằng cách tìm các đường tăng luồng. Để tìm đường tăng luồng ta sẽ áp dụng phương pháp gán nhãn cho các đỉnh. Mỗi đỉnh trong quá trình thực hiện thuật toán sẽ ở một trong ba trạng thái: chưa có nhãn, có nhãn chưa xét, có nhãn đã xét. Nhãn của một đỉnh v gồm 2 phần và có một trong 2 dạng sau: [+p(v), ε (v)] hoặc [-p(v), ε (v)]. Phần thứ nhất +p(v) (-p(v)) chỉ ra là cần tăng (giảm) luồng theo cung (p(v), v) (cung v, p(v)) còn phần thứ hai ε (v) chỉ ra lượng lớn nhất có thể tăng hoặc giảm luồng trên các cung của đường tăng luồng từ s tới v. Đầu tiên chỉ có đỉnh s được khởi tạo và nhãn của nó là chưa xét, còn tất cả các đỉnh còn lại là đều chưa có nhãn. Từ s ta gán nhãn cho tất cả các đỉnh kề với nó và nhãn của đỉnh s sẽ trở thành đã xét. Tiếp theo, từ mỗi đỉnh v có nhãn chưa xét ta lại gán nhãn cho tất cả các đỉnh chưa có nhãn kề nó và nhãn của đỉnh v trở thành đã xét. Quá trình sẽ lập lại cho đến khi hoặc là đỉnh t trở thành có nhãn hoặc là nhãn của tất cả các đỉnh có nhãn đều là đã xét nhưng đỉnh t vẫn không có nhãn. Trong trường hợp thứ nhất ta tìm được đường tăng luồng, còn trong trường hợp thứ hai đối với luồng đang xét không tồn tại đường tăng luồng (tức là luồng đã là cực đại). Mỗi khi ta tìm được đường tăng luồng, ta lại tăng luồng theo đường tìm được, sau đó xoá tất cả nhãn và đối với luồng mới thu được lại sử dụng phép gán nhãn các đỉnh để tìm đường tăng luồng. Thuật toán sẽ kết thúc khi nào đối với luồng đang có trong mạng không tìm được đường tăng luồng.

Hai thủ tục tìm đường tăng luồng và tăng luồng có thể mô tả như sau.

```
void Find_Path()
(* Thủ tục gán nhãn tìm đường tăng luồng
        p[v], ε [v] là nhãn của đỉnh v;
        VT – danh sách các đỉnh có nhãn nhưng chưa xét;
        c[[u,v] – khả năng thông qua của cung (u,v), u, v ∈ V;
        f[u, v] – luồng trên cung (u, v), u, v ∈ V . *)
        p[s]=s;
        ε [s]=+∞ ;
        VT=γ { s} ;
        PathFound=1;
        While VT<>∅
        {
                U ⇐ VT; (* Lấy u từ VV *)
                For v ∈ VT
                {
```

```
                    If (f[u,v] <c[u,v]
                    {
                        p[v]=u;
                        ε [v]=min { ε [u], c[u,v] – f[u,v] };
                        VT= VT ∪ { v} ; (* Nạp v vào danh sách đỉnh có
                        nhãn *)
                        if v = t
                            exit;
                        else
                            if (f[v,u]>0)
                            {
                                p[v]=u;
                                ε [v]=min{ ε [u], f[v,u]} ;
                                VT= VT ∪ { v} ; (* Nạp v vào danh
                                sách đỉnh có nhãn *)
                                if v = t
                                    exit;
                            }
                    }
        }
        PathFound=0;
}
```

void Inc_Flow()

```
(* Tăng luồng theo đường tăng *)
        v=p[t]; u=t; tang=ε [t];
        while u<> s
        {
            if v>0 then
                f[v,u] = f[v,u] + tang;
            else
            {
                v = -v;
                f[u,v] = f[u,v] - tang;
```

```
                }
                u = v;
                v = p[u];
            }
        }
```

Thuật toán Ford_Fulkerson được thực hiện nhờ thủ tục:

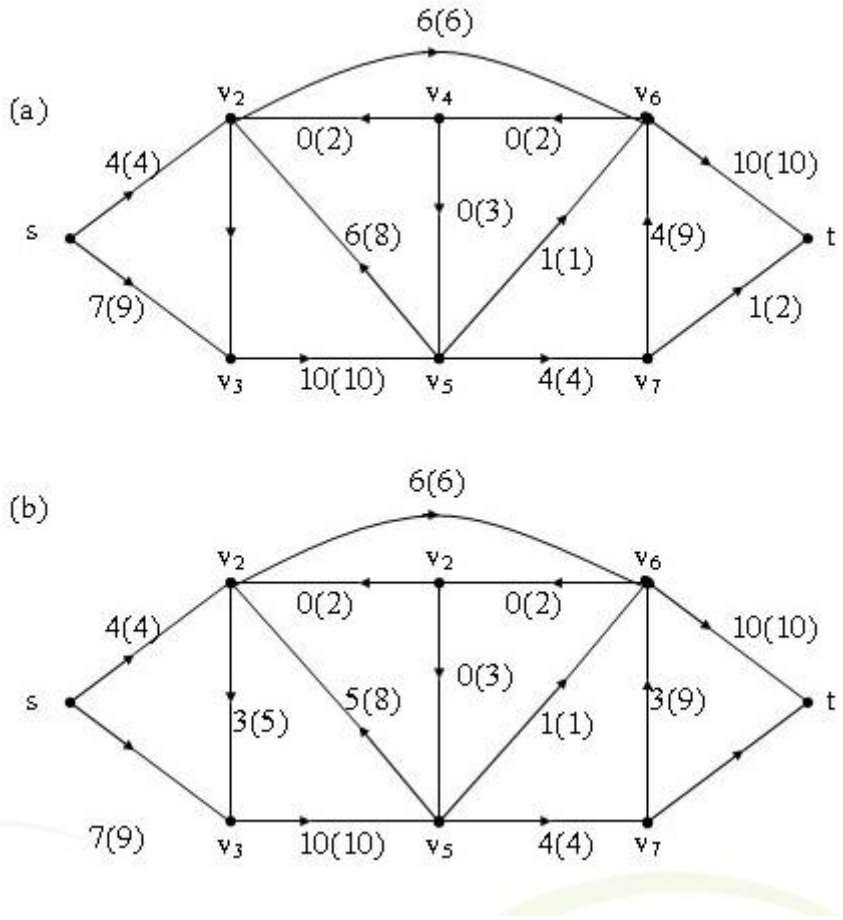
```
        void Max_Flow()
        (*Thuật toán Ford_Fulkerson *)
         (* Khởi tạo: Bắt đầu từ luồng với giá trị 0 *)
            for u ∈ V
                for v ∈ V
                    f[u,v] = 0;
            stop=0;
            while (!stop)
            {
                find_path;
                if pathFound
                        Inc_Flow
                else stop=1;
            }
            <Luồng cực đại trong mạng là f[u,v], u, v ∈ V >
            <Lát cắt hẹp nhất là (VT, V\VT)>
        }
```

Giả sử là khả năng thông qua của tất cả các khung của đồ thị là các số nguyên. Khi đó sau mỗi lần tăng luồng, giá trị luồng sẽ tăng lên ít nhất là 1. Từ đó suy ra thuật toán Ford_Fulkerson sẽ dừng sau không quá val(f*) lần tăng luồng và cho ta luồng cực đại trong mạng. Đồng thời, rõ ràng $f^*(u,v)$ sẽ là số nguyên đối với mỗi cung $(u,v) \in E$. Từ đó ra có các kết quả sau:

Định lý 2 (Định lý về luồng cực đại trong mạng và lát cắt hẹp nhất). Luồng cực đại trong mạng bằng khả năng thông qua của lát cắt hẹp nhất.

Ta kết thúc mục này bởi ví dụ minh hoạ cho thuật toán Ford_Fulkerson sau đây. Hình 2(a) cho mạng G cùng với thông qua của tất cả các cung và luồng giá trị 10 trong nó. Hai số viết bên cạnh mỗi cung là khả năng thông qua của cung (số trong ngoặc) và luồng trên

cung. Đường tăng luồng có dạng (s, v_3, v_2, v_6, v_7, t). Ta tính ược ε (t) = 1, giá trị luồng tăng từ 10 lên 1. Hình 2 (b) cho luồng thu được sau khi tăng luồng theo đường tăng tìm được.



Hình 2. Tăng luồng dọc theo đường tăng

Luồng trong hình 3 (b) đã là cực đại. Lát cắt hẹp nhất

X = { s, v_2, v_3, v_5} , X = { v_4, v_6, v_7, t} .

Giá trị luồng cực đại là 11.

4. ỨNG DỤNG TRONG TỔ HỢP

Bài toán luồng cực đại có rất nhiều ứng dụng trong việc giải bài toán tổ hợp. Khó khăn chính ở đây là phải xây dựng mạng tương ứng sao cho việc tìm luồng cực đại trong nó sẽ tương đương với việc giải bài toán tổ hợp đặt ra. Mục này sẽ giới thiệu một bài toán như vậy: **Bài toán đám cưới vùng quê.**

Có m chàng trai ở một vùng quê nọ. Đối với mỗi chàng trai ta biết các cô gái mà anh ta vừa ý. Hỏi khi nào thì có thể tổ chức các đám cưới trong đó chàng trai nào cũng sánh duyên với các cô gái mà mình vừa ý.

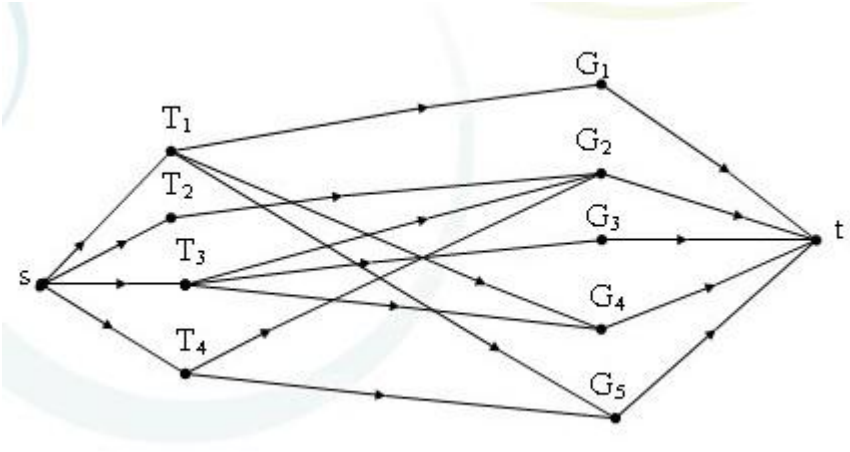
Ta có thể xây dựng đồ thị với các đỉnh biểu thị các chàng trai và các cô gái, còn các cung biểu thị sự vừa ý của các chàng trai với các cô gái. Khi đó ta thu được một đồ thị hai phía.

Thí dụ.

Có 4 chàng trai {T_1, T_2, T_3,T_4} và 5 cô gái {G_1, G_2, G_3,G_4, G_5}. Sự vừa ý cho trong bảng sau

Chàng trai	Các cô gái mà chàng trai ưng ý
T_1	G_1, G_4, G_5
T_2	G_2
T_3	G_2, G_3,G_4
T_4	G_2, G_4

Đồ thị tương ứng được cho trong hình 7.



Hình 3. Mạng tương ứng với bài toán đám cưới vùng quê

Đưa vào điểm phát s và điểm thu t. Nối s với tất cả các đỉnh biểu thị các chàng trai, và nối t với tất cả các đỉnh biểu thị các cô gái. Tất cả các cung của đồ thị đều có khả năng thông qua bằng 1. Bắt đầu từ luồng 0, ta tìm luồng cực đại trong mạng xây dựng được theo thuật toán Ford-Fulkerson. Từ định lý về tính nguyên, luồng trên các cung là các số hoặc 1. Rõ ràng là nếu luồng cực đại trong đồ thị có giá trị Vmax = m, thì bài toán có lời giải, và các cung với luồng bằng 1 sẽ chỉ ra cách tổ chức đám cưới thoả mãn điều kiện đặt ra. Ngược lại, nếu bài toán có lời giải thì Vmax = m. Bài toán về đám cưới vùng quê là một trường hợp riêng của bài toán về cặp ghép trên đồ thị hai phía mà để giải nó có thể xây dựng thuật toán hiệu quả hơn.

Bài tập lý thuyết

Thực hiện thuật toán Ford-Fulkerson tìm luồng cực đại trong mạng của các hình vẽ sau.

Trình bày các kết quả tình toán trong mỗi bước lặp bao gồm :

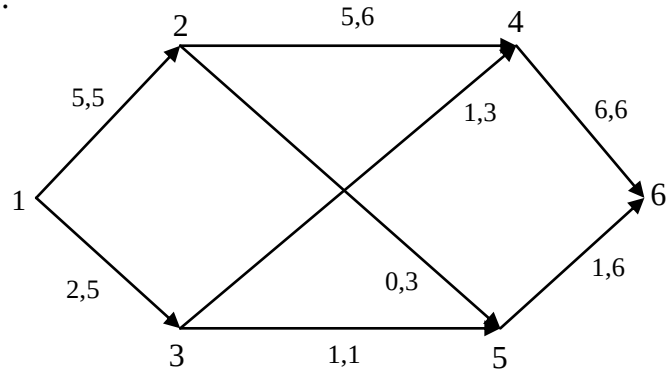
-Đồ thị tăng luồng (mạng thặng dư)

-Đường tăng luồng (đường đi bổ sung) tìm được theo tìm kiếm theo chiều rộng và khả năng thông qua của nó (giả thiết khi duyệt các đỉnh kề của một đỉnh ta duyệt theo thứ tự tăng dần của chỉ số).

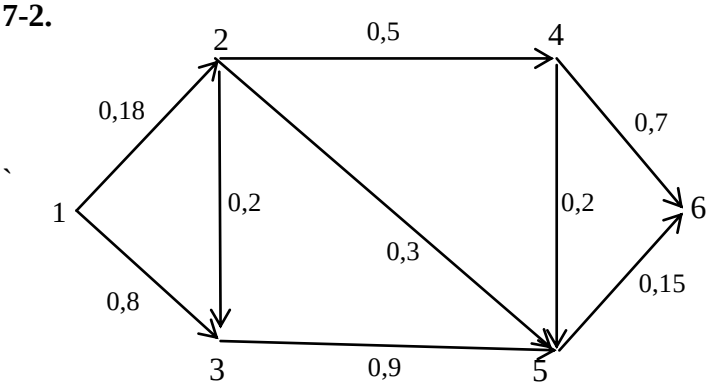
-Mạng cùng luồng tìm được sau khi tăng luồng.

Kết quả cuối cùng: Cần đưa ra giá trị luồng cực đại.

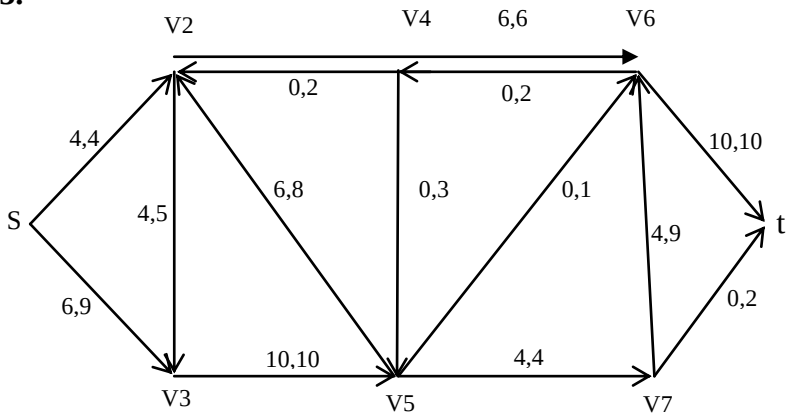
7-1.



7-2.



7-3.



Bài tập thực hành

7-4.Có m chàng trai, n cô gái và k bà mối, Mỗi bà mối p (p=1,2,…,k) có một danh sách L_p một số chàng trai và cô gái trong số các chàng trai và cô gái nói trên là khách hàng của bà ta. Bà mối p có thể se duyên cho bất cứ cặp trai gái nào là khách hàng của bà ta, nhưng không đủ sức tổ chức quá d_p đám cưới. Hãy xây dựng thuật toán căn cứ vào danh sách L_p, d_p, p=1,2,…,k; đưa ra cách tổ chức nhiều nhất các đám cưới giữa m chàng trai và n cô gái với sự giúp đỡ của các bà mối.

7-5. Cho hai dãy số nguyên dương $\{p_i, i=1,2,\ldots,m\}$ và $\{q_j, j=1,2,\ldots,n\}$. Hãy xây dựng ma trận $A=\{a_{ij}:i=1,2,\ldots,m; j=1,2,\ldots n\}$ với các phần tử $a_{i\,j} \in \{0,1\}$ có tổng các phần tử trên dòng i là p_i , tổng các phần tử trên cột j là q_j.

Phần cài đặt các thuật toán căn bản quan trọng

Luồng cực đại trong mạng

```
1. #include <conio.h>
2. #include <stdio.h>
3. #include <iostream.h>
4. #define max 100
5. int c[max][max],f[max][max],d[max],p[max];
6. int pathfound,n,m,s,t;
7. void Nhapsolieu()
8. {
9. FILE *ftext;
10. int u,v;
11. clrscr();
12. ftext=fopen("D:\\DOTHI\\FM.inp","rt");
13. fscanf(ftext,"%d%d%d%d",&n,&m,&s,&t);
14. for (int i=1;i<=m;i++)
15. {
16. fscanf(ftext,"%d",&u);
17. fscanf(ftext,"%d",&v);
18. fscanf(ftext,"%d",&c[u][v]);
19. }
20. fclose(ftext);
21. }
22. int min(int a,int b)
23. {
24. if (a< b) return a;
25. return b;
26. }
27. void Find_path()
28. {
29. int  nvt=1,u,vt[max];
30. for (int i=1;i<=max;i++)
31. {p[i]=0;d[i]=0;}
```

```
32. p[s]=s;
33. d[s]=max;
34. vt[nvt]=s;
35. pathfound=1;
36. while (nvt!=0)
37. {
38. u=vt[nvt--];
39. for (int v=1;v<=n;v++)
40. if (p[v]==0)
41. {
42. if (c[u][v]>0 && f[u][v]<c[u][v])
43. {
44. p[v]=u;
45. d[v]=min(d[u],c[u][v]-f[u][v]);
46. vt[++nvt]=v;
47. if (v==t) return;
48. }
49. if (c[v][u]>0 && f[v][u]>0)
50. {
51. p[v]=-u;
52. d[v]=min(d[u],f[v][u]);
53. vt[++nvt]=v;
54. if (v==t) return;
55. }
56. }
57. }
58. pathfound=0;
59. }
60. void Inc_flow()
61. {
62. int  v=p[t],u=t,tang=d[t];
63. while (u!=s)
64. {
65. if (v>0) f[v][u]+=tang;
```

```
66. else
67. {
68. v=-v;
69. f[u][v]-=tang;
70. }
71. u=v;
72. v=p[u];
73. }
74. }
75. void Max_flow()
76. {int stop=0;
77. while (stop==0)
78. {
79. Find_path();
80. if (pathfound==1) Inc_flow();
81. else stop=1;
82. }
83. }
84. void main()
85. {
86. Nhapsolieu();
87. Max_flow();
88. int valf=0;
89. for (int u=1;u<=n;u++)
90. if (c[s][u]>0)
91. valf+=f[s][u];
92. cout<<"Ma tran c va f ket qua(dinh dau, dinh cuoi,Tai nang, luong tren canh):\n";
93. for (u=1;u<=n;u++)
94. for (int v=1;v<=n;v++)
95. if (c[u][v]>0)
96. cout<<u<<" "<<v<<" "<<c[u][v]<<" "<<f[u][v]<<endl;
97. cout<<"Gia tri cuc dai cua luong trong mang la :"<<valf;
98. getch();
99. }
```

FM.INP

6 9 1 6

1 2 6

1 3 9

2 3 1

2 4 4

2 5 3

3 5 7

5 6 7

4 6 8

5 4 3