MỞ ĐẦU: GIỚI THIỆU BÀI TOÁN

BÀI TOÁN LUỒNG CỰC ĐẠI
THUẬT TOÁN FORD - FULKERSON

I/ Bài toán luồng trên mạng.

Nhiều bài toán quy hoạch tuyến tính có thể quy về bài toán làm cực tiểu phí tổn vận chuyển hàng trong một mạng (gồm các nút và các cung đường) sao cho đảm bảo được các nhu cầu ở một số nút khi đã biết nguồn cung cấp tại một số nút khác. Các bài toán như vậy được gọi là các ***bài toán luồng trên mạng*** (network flow problem) hoặc ***bài toán chuyển vận*** (transshipment problem). Đây là lớp bài toán quan trọng nhất và hay gặp nhất trong quy hoạch tuyến tính. Lớp này bao gồm các bài toán quen thuộc trong thực tế như: bài toán vận tải, các bài toán mạng điện và mạng giao thông, các bài toán quản lý và phân bổ vật tư, bài toán bổ nhiệm, bài toán kế hoạch tài chính, bài toán đường ngắn nhất, bài toán luồng cực đại …

II/ Bài toán luồng cực đại và thuật toán Ford-Fulkerson

Bài toán ***luồng cực đại*** trong mạng cũng là một trong số những 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 những năm 1950, và gắn liền với tên tuổi của hai nhà bác học Mỹ là **L.R.Ford** và **D.R.Fulkerson**. Bài toán luồng cực đại trong mạng có nhiều ứng dụng trong thực tế như: Bài toán xác định cường độ dòng lớn nhất của dòng vận tải giữa hai nút của một bản đồ giao thông, bài toán tìm luồng dầu lớn nhất có thể bơm từ tàu chở dầu vào bể chứa của một hệ thống đường ống dẫn dầu…Ngoài ra, ứng dụng của bài toán còn để giải các bài toán như: Bài toán đám cưới vùng quê, bài toán về hệ thống đại diện chung, bài toán phân nhóm sinh hoạt, bài toán lập lịch cho hội nghị…

Trong bài tiểu luận này **chúng em** sẽ trình bày "Bài toán luồng cực đại trong mạng" sử dụng thuật toán của ***Ford - Fulkerson*** (1962) để giải bài toán đặt ra và nêu một số ứng dụng của bài toán.

PHẦN I: TRÌNH BÀY LÝ THUYẾT

**** Lý thuyết về mạng.***

1.Định nghĩa:

1.1.Định nghĩa 1: *Mạng* (Network)

Mạng (Network) là đơn trọng đồ có hướng G=(V, E, c) thoã mãn

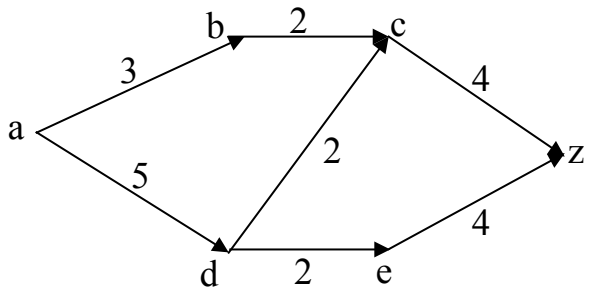
(i) Có duy nhất một đỉnh, gọi là *nguồn* (source), không liên thuộc cung vào

(ii) Có duy nhất một đỉnh, gọi là *đích* (Sink), không liên thuộc cung ra

(iii) Trọng sô c_{ij} của cung (i, j) là các số không âm gọi là *khả năng thông qua của cung* (capaciti).

(iv) Đồ thị liên thông (yếu)

Ví dụ 1: Đồ thị sau là mạng với nguồn là đỉnh a và đích là z



Hình 1

Ta có khả năng thông qua $c_{ab} = 3$, $c_{bc} = 2$, $c_{ad} = 5$, $c_{dc} = 2$, $c_{de} = 2$, $c_{cz} = 4$, $c_{ez} = 4$.

1.2. Định nghĩa 2: *luồng* (hay luồng trong mạng)

Cho mạng G, với khả năng thông qua c_{ij}, $(i,j) \in G$. Tập các giá trị

$$\{f_{ij} \mid (i, j) \in G\}$$

Gọi là luồng trên mạng G nếu thoã mãn

(i) $0 \le f_{ij} \le c_{ij} \ \forall (i, j) \in G$

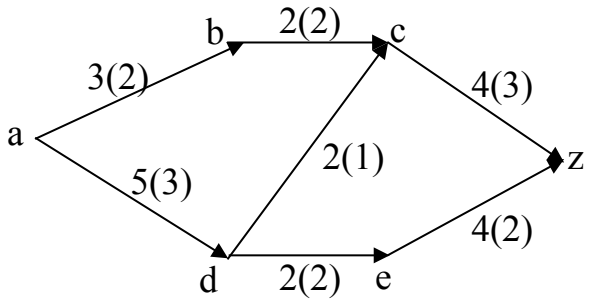
(ii) Với mọi đỉnh k không phải là nguồn hoặc đích

$$\sum_{(i,k)\in G} f_{ik} = \sum_{(k,j)\in G} f_{kj}$$

Ví dụ 2: Với đồ thị ở ví dụ trên, tập $\{f_{ij}\}$ sau là luồng

$$f_{ab} = 2,\ f_{bc} = 2,\ f_{cz} = 3,\ f_{ad} = 3,\ f_{dc} = 1,\ f_{de} = 2,\ f_{ez} = 2$$

được biểu diễn bằng các số trong ngoặc đơn trong mạng



Hình 2

1.3. <u>Định lý 1</u>: Cho f_{ij}, $(i,j)\in G$, là luồng trên mạng G với nguồn a và đích z. Khi đó:

$$\sum_{(a,i)\in G} f_{ai} = \sum_{(i,z)\in G} f_{iz}$$

<u>Chứng minh:</u>

Gọi V là tập các đỉnh. Với các đỉnh i,j không kề nhau ta gán $f_{ij}=0$. Ta có

$$\sum_{j\in V}\sum_{i\in V} f_{ai} = \sum_{j\in V}\sum_{i\in V} f_{iz}$$

$$\Leftrightarrow \sum_{j\in V}\left(\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji}\right) = 0$$

$$\Leftrightarrow \sum_{j\in V\setminus\{a,z\}}\left(\sum_{i\in V} f_{ij} - \sum_{i\in V} f_{ji}\right) + \left(\sum_{i\in V} f_{iz} - \sum_{i\in V} f_{zi}\right) + \left(\sum_{i\in V} f_{ai} - \sum_{i\in V} f_{ai}\right) = 0$$

$$\Leftrightarrow -\sum_{(a,i)\in G} f_{ai} + \sum_{(i,z)\in G} f_{iz} = 0$$

$$\Leftrightarrow \sum_{(a,i)\in G} f_{ai} = \sum_{(i,z)\in G} f_{iz}$$

1.4. <u>Định nghĩa 3</u>: *Giá trị của luồng*

Cho luồng f trên mạng G. *Giá trị của luồng* f được định nghĩa là đại lượng

$$V(f) = \sum_{(a,i)\in G} f_{ai} = \sum_{(i,z)\in G} f_{iz}$$

2.<u>Bài toán luồng cực đại</u>:

*** Phát biểu bài toán luồng cực đại.** Trong thực tế ta thường gặp bài toán gọi là bài toán tìm luồng cực đại như sau: Cho mạng G với nguồn a, đích z và khả năng thông qua c_{ij}, $(i,j)\in G$. Trong số các luồng trên mạng G, tìm luồng có giá trị lớn nhất.

Ý tưởng xây dựng luồng cực đại như sau: xuất phát từ luồng nào đó, ta tìm đường đi (không định hướng) từ a đến z cho phép hiệu chỉnh giá trị luồng

trên đường đi đó sao cho luồng mới có giá trị lớn hơn. Nếu không tìm được đường đi như vậy thì ta có luồng cực đại.

Giả sử

$$P = (a, u, \ldots, i, j, \ldots, v, z)$$

là đường đi không có hướng từ a đến z.

Nếu cạnh (i, j) là cung trên P thì cung đó cùng hướng với P. Ngược lại nếu (j, i) là cung thì cung đó ngược hướng với P.

Tập các cung cùng hướng với P kí hiệu là P_+.

Tập các cung ngược hướng với P kí hiệu là P_-.

Cơ sở của thuật giải là định lý sau.

2.1. <u>Định lý 2</u>: Cho f là luồng trên G. Giả sử

$$P = (a, u, \ldots, i, j, \ldots, v, z)$$

là đường đi không định hướng từ a đến z thoã

(i) Với mỗi cung (i, j) cùng hướng với P

$$f_{ij} < c_{ij}$$

(ii) Với mỗi cung (i, j) ngược hướng với P

$$0 < f_{ij}$$

Đặt

$$\delta := \min\{x \mid x \in M\} > 0$$

Trong đó M là tập các giá trị $c_{ij} - f_{ij}$, $(i, j) \in P_+$ và f_{ij}, $(i, j) \in P_-$

Ta xây dựng luồng f' như sau

$$f'_{ij} := \begin{cases} f_{ij} & \forall (i, j) \notin P \\ f_{ij} + \delta & \forall (i, j) \in P_+ \\ f_{ij} - \delta & \forall (i, j) \in P_- \end{cases}$$

Khi đó luồng f' có giá trị lớn hơn giá trị của luồng f một lượng là δ, tức là

$$v(f') = v(f) + \delta$$

<u>Chứng minh:</u>

Theo cách xây dựng thì f' thoã mãn các điều kiện của luồng. Cung $(a, u) \in P$, có

$$f'_{au} = f_{au} + \delta$$

nên giá trị

$$v(f') = \sum_{(a,i) \in G} f'_{ai} = \sum_{(a,i) \in G} f_{ai} + \delta$$

PHẦN II: PHÂN TÍCH VÀ THIẾT KẾ

I/ Cấu trúc dữ liệu.

Chương trình cài đặt sử dụng:

+ Các mảng hai chiều (ma trận) để biểu diễn khả năng thông qua của các cung trên mạng, biểu diễn luồng trên các cung.

+ Mảng một chiều để lưu vết đường tăng luồng (đặt nhãn cho đỉnh).

+ Sử dụng kiểu file text để lấy dữ liệu và lưu kết quả có được từ chương trình

II/ Phân tích và thiết kế.

II.1. Phân tích bài toán

Xuất phát từ Định lý 2, ta 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 luồng tăng:

(i) Xuất phát từ một luồng ban đầu f:=0

(ii) Ta đi tìm một đường đi tăng luồng P, với giá trị luồng f' lớn hơn giá trị luồng trước đó một lượng là δ. Nếu không có luồng như vậy, thì ta tìm được luồng cực đại, thuật toán kết thúc. Nếu có, ta qua bước dưới đây.

(iii) Hiệu chỉnh luồng hiện hành cho đến khi nhận giá trị $\delta(P) = +\infty$ thuật toán kết thúc, ta tìm được luồng cực đại.

Trong đó $\delta(P)$ - Lượng luồng tăng thêm, hay nói khác là làm sự tăng luồng dọc theo đường đi tăng luồng P một lượng thích hợp mà các ràng buộc của bài toán vẫn thoả.

- Cách tìm đường đi tăng luồng: Ta sử dụng thuật toán gán nhãn có nội dung như sau. Một đường đi P thoả mãn về đường đi tăng luồng, nhưng chỉ đi từ x đến y nào đó (chưa tới z, nói chung) sẽ được gọi là đường đi chưa bão hoà.

- Ta nói đỉnh i là đã đánh dấu (i is labeled) nếu ta biết là có một đường đi chưa bão hoà từ x tới i. Bây giờ ta sẽ xét tất cả các đỉnh j có nối trực tiếp đến đỉnh i (sẽ gọi là ở cạnh đỉnh i) xem chúng có thể được gán nhãn hay không khi i đã gán nhãn. Việc này được gọi là thăm (scanning) đỉnh i.

- Nếu (i,j) có luồng trên cung $f_{ij} < c_{ij}$ thì ta có thể nối thêm cung (i,j) và đường đi chưa bão hoà P từ x đến i để được đường đi chưa bão hoà tới j. Vậy j có thể gán nhãn.

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

+ Để tìm đường tăng luồng trong Gf có thể sử dụng thuật toán tìm kiếm theo chiều rộng (hay thuật toán tìm kiếm theo chiều sâu) bắt đầu từ đỉnh a, trong đó không cần xây dựng tường minh đồ thị Gf. Ford - Fulkerson đề nghị thuật toán gán nhãn giải bài toán luồng trong mạng. Thuật toán bắt đầu từ luồng chấp nhận được nào đó 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 hai 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 theo cung này. Đầu tiên chỉ có đỉnh a được khởi tạo nhãn và nhãn của nó là chưa xét, còn tất cả các đỉnh còn lại đều chưa có nhãn. Từ a ta gán cho tất cả các đỉnh kề với nó và nhãn của đỉnh a sẽ trở thành nhãn đã xét. Tiếp theo, từ mỗi đỉnh v có nhãn chưa xét ta lại gán nhãn cho tất cả các nhãn chưa có nhãn kề với nó và nhãn của đỉnh v trở thành nhãn đã xét. Quá trình sẽ được lặp lại cho đến khi hoặc là đỉnh z trở thành có nhãn hoặc là nhãn của tất cả các đỉnh có nhãn đều là đã xét nhưng đỉnh z vẫn chưa 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 tìm được đường tăng luồng, ta lại tăng luồng theo đường tìm được, sau đó xoá tất cả các 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.

**** Thuật toán Ford – Fulkerson:***

Đầu vào: Mạng G với nguồn a, đích z, khả năng thông qua C = (c_{ij}), (i, j)∈ G.

Ký hiệu a = v_0, …, v_n = z.

Đầu ra: Luồng cực đại F = (f_{ij}), (i, j)∈ G.

Các bước:

(i) Khởi tạo luồng xuất phát: f_{ij}:=0 ∀(i, j)∈ G.

(ii) Đặt nhãn cho nguồn: Cho nguồn a mang nhãn (,∞)

(iii) Kiểm tra nhãn của đích: nếu đích z có nhãn, sang bước (vi). Ngược lại sang bước (iv).

(iv) Xác định đỉnh đánh dấu. Trong số các đỉnh mang nhãn và chưa được đánh dấu chọn đỉnh v_i với chỉ số I nhỏ nhất. Nếu không tồn tại đỉnh như vậy, kết thúc, luồng F là cực đại. Ngược lại gán v:= v_i và đánh dấu đỉnh v.

(v) Đặt nhãn các đỉnh chưa có nhãn kề đỉnh v: Giả sử (α,Δ) là nhãn đỉnh v. Xét các cung có dạng (v,w), (w,v) theo thứ tự (v,v_0), (v_0,v), (v,v_1),(v_1,v)....trong đó w chưa được mang nhãn.

Với cung dạng (v,w), nếu $f_{vw} < c_{vw}$, đặt nhãn đỉnh w là (v, min {Δ, $c_{vw} - f_{vw}$}), nếu $f_{vw} = c_{vw}$, không đặt nhãn đỉnh w.

Với cung dạng (w, v), nếu $f_{wv} > 0$, đặt nhãn đỉnh w là (v, min {Δ, f_{wv}}), nếu f_{wv} = 0 , không đặt nhãn đỉnh w.

Sang bước (iii).

(vi) Hiệu chỉnh luồng: Giả sử (α, β) là nhãn của đích z. Đặt

$$W_0 := z,\ w_1 := \beta$$

Nếu nhãn của w_i là (β', Δ'), thì đặt $w_{i+1} := \beta'$. Tiếp túc quá trình cho đến khi $w_k = a$. Đến đây nhận được đường đi P từ a đến z.

$$P = (a = w_k, w_{k-1}, \ldots, w_0 = z)$$

Ta hiệu chỉnh luồng f trên P như sau:

$$f_{ij} := \begin{cases} f_{ij} + \delta & \forall (i, j) \in P_+ \\ f_{ij} - \delta & \forall (i, j) \in P_- \end{cases}$$

Sau đó xoá tất cả các nhãn của các đỉnh trên P rồi quay lại bước (ii).

II.2. Thiết kế:

Sơ đồ của thuật toán Ford – Fulkerson có thể mô tả trong thủ tục sau đây:

```
Procedure Max_Flow;{ Thuật toán Ford – Fulkerson }
begin
    (* Khởi tạo: Bắt đầu từ luồng với giá trị 0 *)
    for u ∈ V do
        for v ∈ V do f_uv:=0;
    Stop := false;
    While not Stop do
    If < Tìm được đường tăng luồng P> then <Tăng luồng dọc theo P>
    else Stop:= true;
end;
```

*** Thuật toán gán nhãn (The labeling algorithm)**

Gọi V_T là tập mọi đỉnh đã gán nhãn nhưng chưa được thăm. Ta có thuật toán để tìm đường đi tăng luồng.

Xuất phát với $V_T = \{a\}$ và a là nút đã đánh dấu duy nhất.

Một bước lặp sẽ có V_T hiện hành và gồm ba bước như sau.

(i) Nếu $z \in V_T$ hoặc $V_T = \varnothing$, thuật toán kết thúc. Ngược lại thì chọn một đỉnh $i \in VT$ để thăm và đưa nó ra khỏi V_T. Xét tất cả các đỉnh cạnh i, tức là xét mọi cung có dạng (i, j) và (j, i).

(ii) Nếu $(i, j) \in E$, $f_{ij} < c_{ij}$ và j chưa gán nhãn thì gán nhãn nó và đưa j vào tập V_T.

(iii) Nếu $(i, j) \in E$, $f_{ij} > 0$ và v chưa gán nhãn thì gán nhãn nó và đưa vào tập V_T.

Bây giờ ta xét kết quả của thuật toán gán nhãn, nó có kết thúc hữu hạn hay không? Nhận xét rằng một đỉnh được vào tập V_T chỉ khi chuyển từ chưa gán nhãn. Do đó một đỉnh chỉ được vào V_T nhiều nhất là một lần, mà mỗi bước lặp bỏ một đỉnh ra khỏi V_T. Do đó, vì số đỉnh của mạng là hữu hạn, thuật toán phải kết thúc hữu hạn.

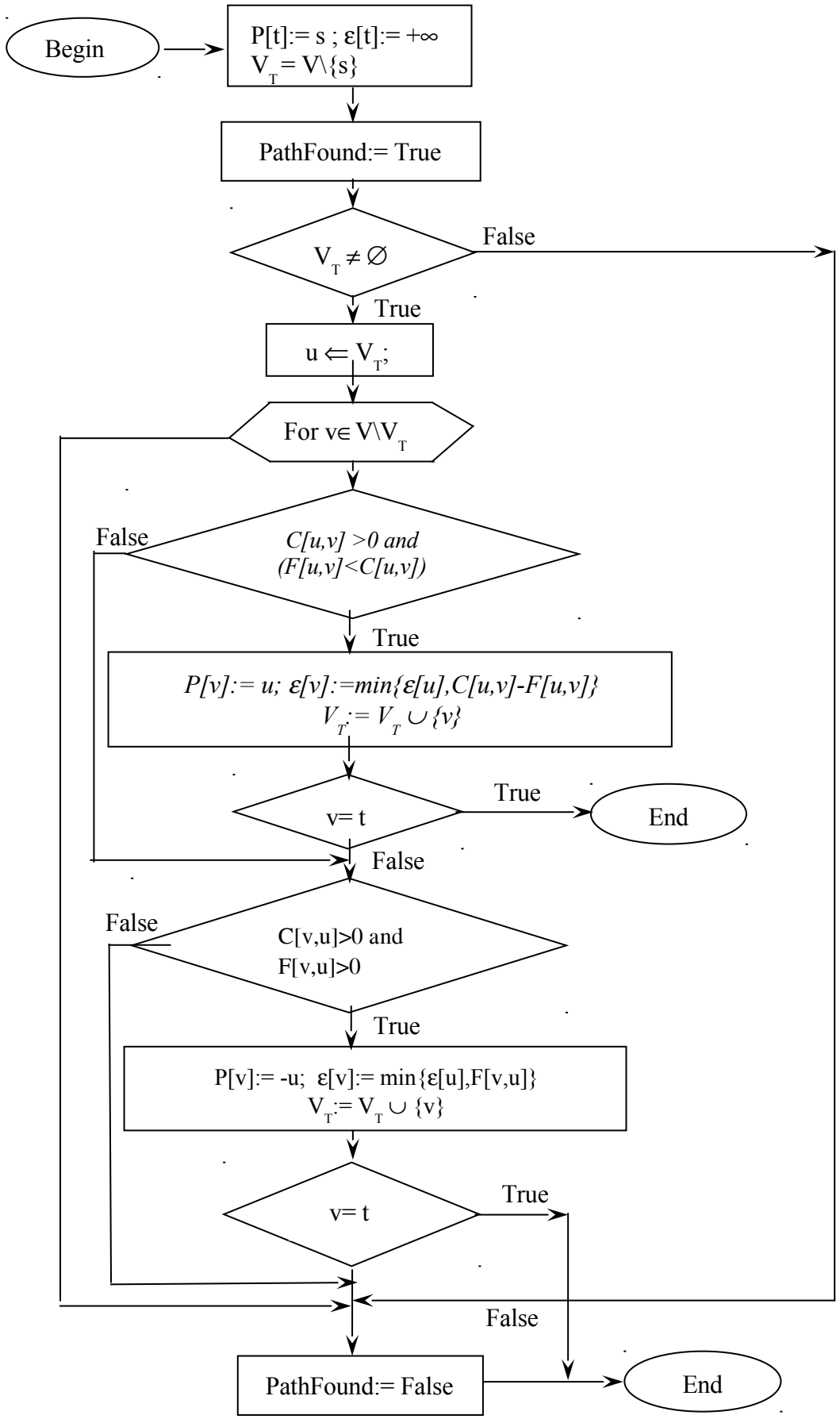
* Hai thủ tục Tìm đường tăng luồng và tăng luồng có thể mô tả bởi chương trình như sau:

```
Procedure Find_Path;{ thủ tục gán nhãn tìm đường tăng luồng }
        p[v], ε[v] là nhãn của đỉnh v;
        V_T – danh sách các đỉnh 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 ) *)
begin
        p[s]:=s;        ε[s]:= +∞;    V_T= V\{s};
        PathFound:=true;{Gán tìm thấy đường tăng luồng là đúng}
        While V_T≠ ∅ do
        Begin
                u<= V_T; (* Lấy u từ V_T *)
                for v ∈ V\V_T  do
                begin
                        if (c[u,v]>0) and (f[u,v]< c[u,v]) then
                        begin
                            p[v]:= u;
                            ε[v]:= min { ε[u],c[u,v] – f[u,v]};
                            V_T = V_T ∪ {v}; (* Nạp v vào danh sách đỉnh có nhãn *)
                                If v=t then exit;
                        end;
                        if (c[v,u]>0) and (f[v,u]>0) then
                        begin
                              p[v]:= -u;
                              ε[v]:= min {ε[u],f[v,u]};
                              V_T = V_T ∪ {v}; (* Nạp v vào danh sách đỉnh có nhãn *)
                               If v=t then exit;
                        end;
                end;
                PathFound:= false;
        end;
procedure Inc_Flow;{ Tăng luồng theo đường tăng }
begin
        v:=p[t]; u:=t; tang:= ε[t];
        while u ≠s  do
        begin
                if v>0 then f[v,u]:= f[v,u] + tang;
                else
                  begin
                        v:= -v;
                        f[u,v]:= f[u,v] – tang;
                  end;
```

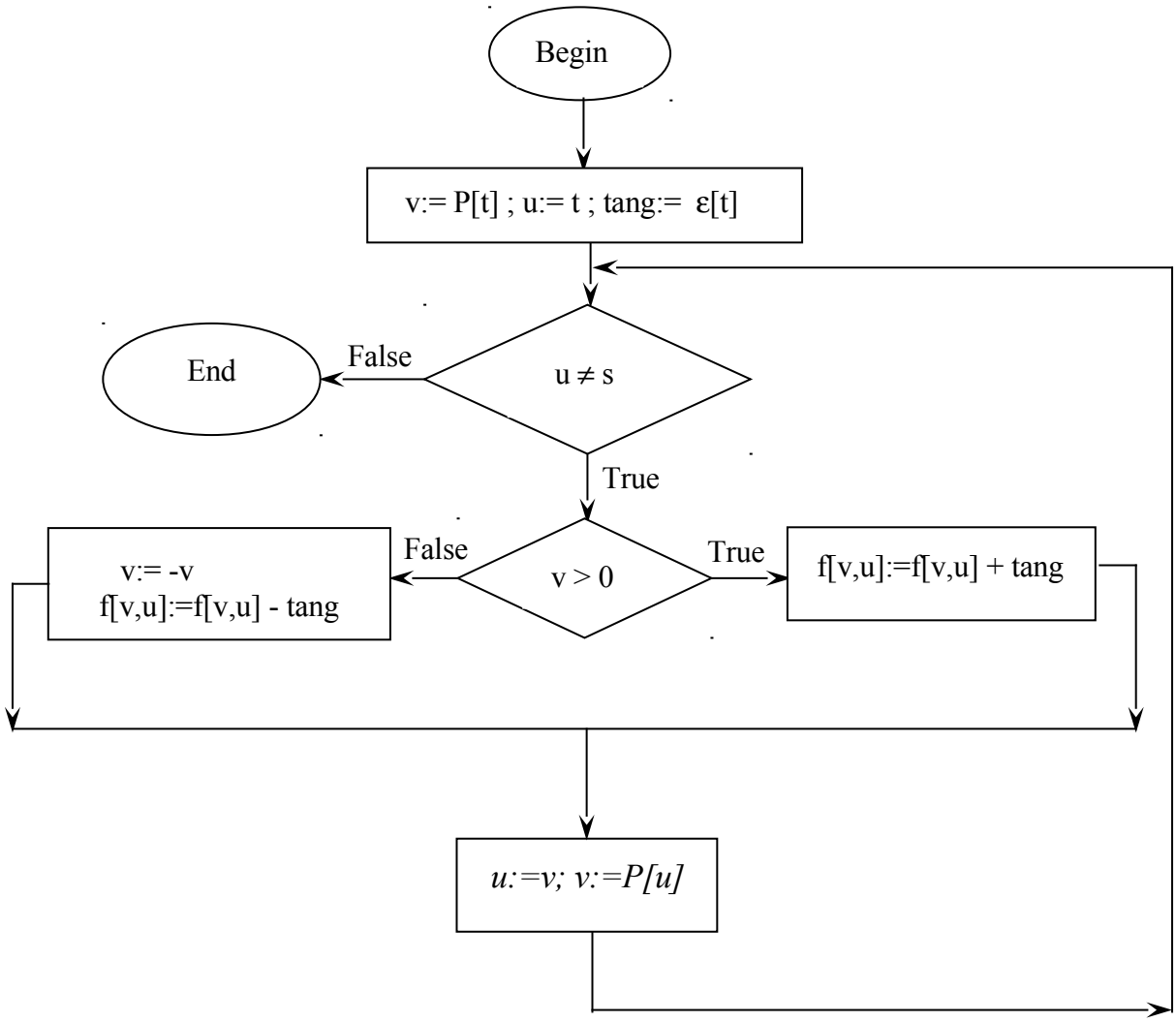
```
            u:= v; v:= p[u];
        end;
end;
```

III./ Sơ đồ khối.

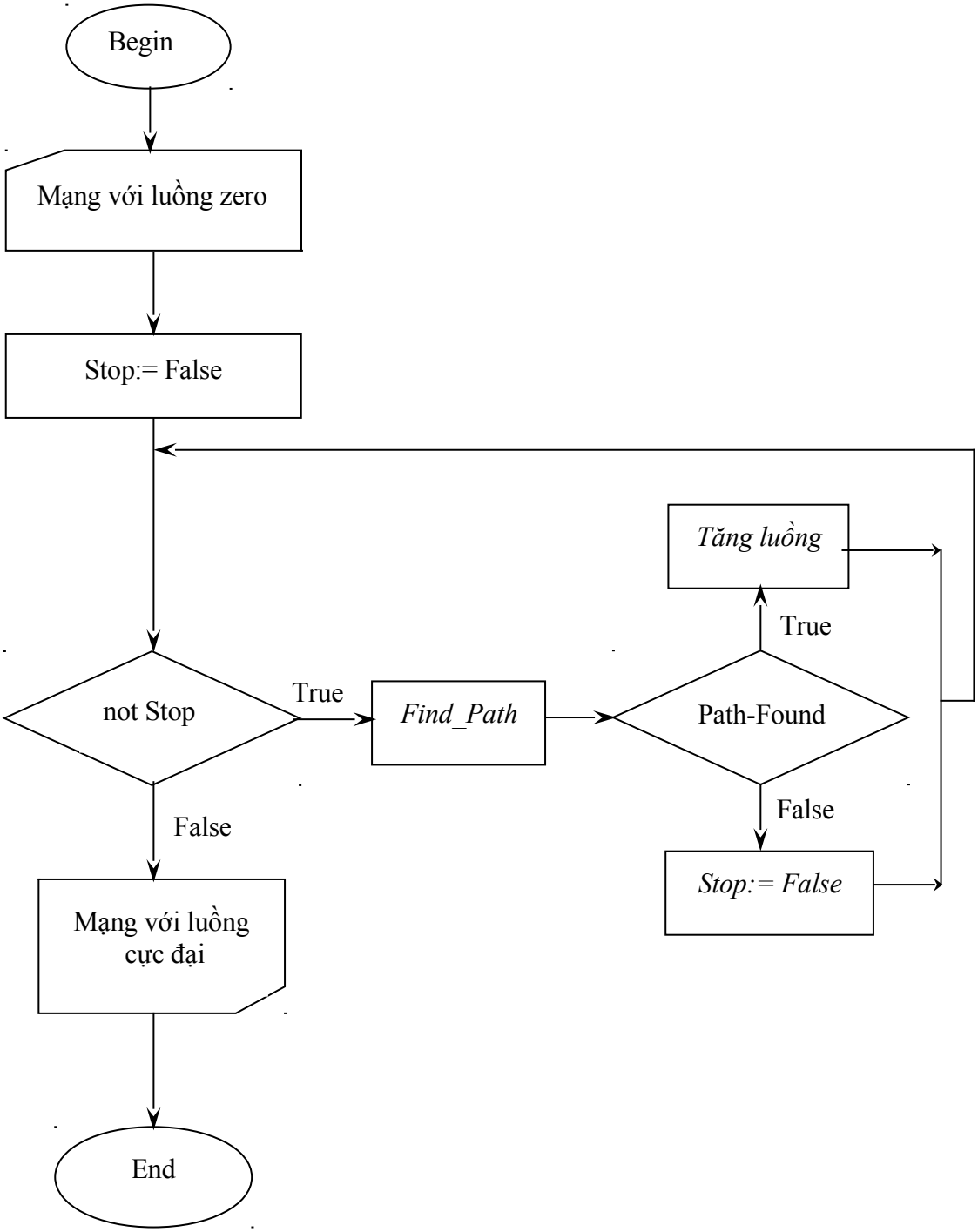
Sơ thuật toán Find_Path (Chi tiết) { Trả về TRUE nếu có đường tăng luồng }



Sơ đồ thuật toán tăng luồng (Inc_Flow)
{ Tăng luồng nếu có đường tăng }



Sơ đồ thuật toán Ford-Fullkerson tổng quát



PHẦN III: CÀI ĐẶT VÀ SỬ DỤNG CHƯƠNG TRÌNH

I/ *Giới thiệu ngôn ngữ lập trình:*

Bài toán được cài đặt bằng ngôn ngữ lập trình Pascal, bao gồm 3 file sau: FORDFULK.PAS, FORDFULK.INP, FORDFULK.INP.

II/ *Cài đặt chương trình:*

Ta cài đặt thuật toán Ford Fulkerson với file input và file output như sau:

Input: File văn bản FORDFULK.INP. Trong đó:

∗Dòng 1: chứa số đỉnh n(<100), số cạnh m của đồ thị, đỉnh nguồn a, đỉnh thứ z theo đúng thứ tự cách nhau ít nhất một khoảng trắng.

∗m dòng tiếp theo mỗi dòng là bộ ba số x_i, y_i, c_i các nhau một dấu cách thể hiện cung (x_i, y_i), đỉnh đầu x_i, đỉnh cuối y_i, và khả năng thông c_i qua của cung đó (i=1..m) là một số nguyên dương.

Output: File văn bản FORDFULK.OUT. Trong đó:

∗Dòng đầu ghi giá trị luồng cực đại tìm được E.

∗m dòng tiếp theo mỗi dòng là bộ ba số x_i, y_i, f_i các nhau một dấu cách thể hiện cung (x_i, y_i), đỉnh đầu x_i, đỉnh cuối y_i, và giá trị luồng trên cung f_i đó (với i=1..m).

Nội dung cài đặt:

```
Program Luong_cuc_dai_thuat_toan_Ford_Fulkerson;
Const
  file_vao = 'FORDFULK.INP';
  file_ra = 'FORKFULK.OUT';
  max = 100;
type
  Mang_du_lieu = array[1..max,1..max] of longInt;
var
  c: Mang_du_lieu;
  f: ^Mang_du_lieu;
  Nhan: array[1..max] of Integer;
  n, a, z: integer;
{------------------------------------------------------------------}
{Lấy dữ liệu từ file}
{------------------------------------------------------------------}
Procedure Nhapdulieu;
```

```
var
  m, i, x, y: Integer;
  fi: Text;
begin
  Assign(fi, file_vao); reset(fi);
  FillChar(c, SizeOf(c), 0);
  Readln(fi, n, m, a, z);{số đỉnh, số cung, đỉnh nguồn, đỉnh thứ}
  for i:=1 to m do
    Readln(fi, x, y, c[x, y]);
  Close(fi);
end;
{------------------------------------------------------------------}
{Hàm tìm đường tăng luồng trên G, trả về True nếu tìm thấy}
{------------------------------------------------------------------}
function FindPath: Boolean;
var
  x, y: Integer;
  hangdoi: array[1..max] of Integer; {Hàng đợi dùng cho BFS}
  Truoc, Sau: Integer;
begin
  FillChar(Nhan, SizeOf(Nhan), 0);{Đặt nhãn cho nguồn}
  Truoc:= 1; Sau:= 1;
  hangdoi[1]:= a;
  Nhan[a]:= n + 1;{Nhãn=0 => y chưa thăm}
  Repeat
    x:= hangdoi[Truoc]; Inc(Truoc);{Lấy x khỏi hàng đợi}
    for y:= 1 to n do
      if (Nhan[y] = 0) and (c[x, y]>f^[x, y]) then {xét mọi y chưa thăm kề x trên
G}
        begin
          Nhan[y]:= x;
          if y = z then {đến được z thì thuật toán dừng}
            begin
```

```
                Findpath:= true; Exit;
              end;
            Inc(Sau); hangdoi[Sau]:= y;{đẩy y vào hang đợi}
          end;
  until Truoc > Sau;
  Findpath := False;
end;
{-------------------------------------------------------------------}
{Thủ tục tăng luồng dọc đường tăng luồng: f:=(f+fp)}
{-------------------------------------------------------------------}
Procedure Tangluong;
var
  Delta, x, y: Integer;
begin
  Delta:= MaxInt;
  y:= z;
  repeat
    x:= Nhan[y];
    if (c[x, y] - f^[x, y]) <  Delta then Delta:= c[x, y]- f^[x, y];
    y:= x;
 until y = a;{f:=f+fp}
 y:= z;
  repeat
    x:= Nhan[y];
    f^[x, y]:= f^[x, y] + Delta;
    f^[y, x]:= f^[y, x] - Delta;
    y:= x;
  until y = a;
end;
{-------------------------------------------------------------------}
{Ghi kết quả vào file}
{-------------------------------------------------------------------}
Procedure Duakqvaofile;
```

```
var
  x, y: Integer;
  m: LongInt;
  fo: Text;
begin
  Assign(fo, file_ra); rewrite(fo);
  m:= 0;
  for x:=1 to n do
    for y:= 1 to n do
      if f^[x, y] > 0 then
      begin
        if x = a then m:= m + f^[a, y];
      end;
      writeln(fo,m:5);
  for x:=1 to n do
    begin
      for y:= 1 to n do if f^[x, y]>0 then
        begin
          write(fo, x:5, y:5, f^[x, y]:10);
          writeln(fo);
        end;
    end;
  close(fo);
end;
{------------------------------------------------------------------}
{CHƯƠNG TRÌNH CHÍNH}
{------------------------------------------------------------------}
BEGIN
  Nhapdulieu;
  New(f);
  FillChar(f^, SizeOf(f^), 0);
  repeat
    if not Findpath then Break;
```

```
    Tangluong;
  until False;
  Duakqvaofile;
  Dispose(f);{giải phóng bộ nhớ cấp cho f^};
END.
```

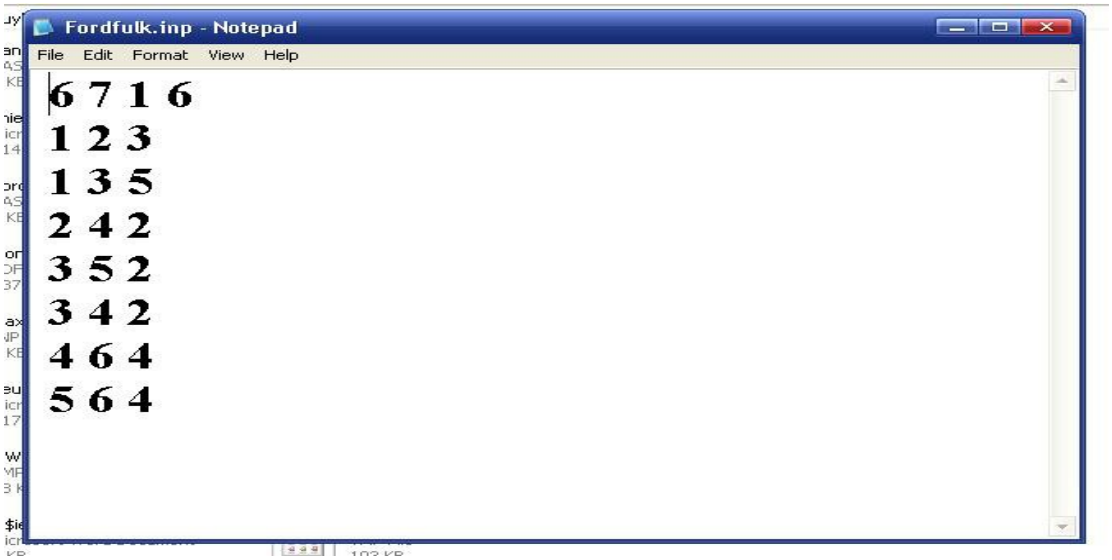
III/ <u>Chức năng và giao diện chương trình</u>:

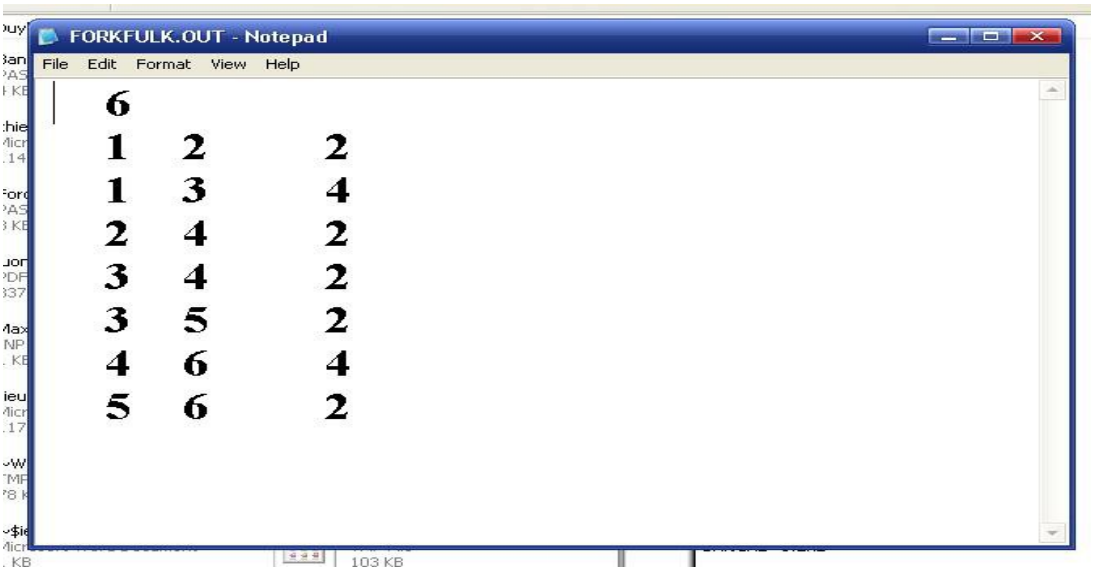
III.1.Chức năng:

Sử dụng chương trình cài đặt thuật toán Ford – Fulkerson bằng pascal để tìm luồng cực đại từ một mạng với dữ liệu cho trước chứa trong file text "Fordfulk.inp". Kết quả được chương trình tự động xử lý và in ra file text "Fordfulk.out"

III.2.Giao diện:

Chương trình không có giao diện giao tiếp, các dữ liệu vào và ra có thể nhập hoặc xem từ trong chương trình pascal, hoặc trên hệ điều hành windows.





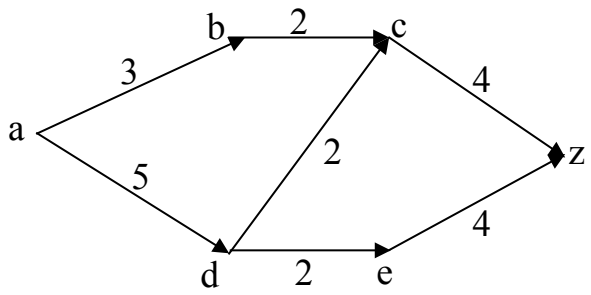
IV/ *Kết quả chạy chương trình:*

Kết quả chạy thử chương trình:

File dữ liệu vào: FORDFULK.INP

(Ví dụ 4.3 trang 162 Chương 4 – Giáo trình LTĐT – PGS Trần Quốc Chiến)

```
6 7 1 6
1 2 3
1 3 5
2 4 2
3 5 2
3 4 2
4 6 4
5 6 4
```



Hình 1

Các đỉnh a, b, c, d, e, z ứng với đỉnh 1, 2, 3, 4, 5, 6 trong chương trình.

File dữ liệu vào: FORDFULK.INP

```
6
  1  2     2
  1  3     4
  2  4     2
  3  4     2
  3  5     2
  4  6     4
  5  6     2
```

Giá trị luồn cực đại E = 6

TÀI LIỆU THAM KHẢO

[1] Giáo trình Lý thuyết đồ thị - Pgs. Trần Quốc Chiến

[2] Ebook Giải thuật và lập trình – Lê Minh Hoàng (ĐHSP Hà Nội)

[3] Nguyễn Đức Nghĩa - Nguyễn Tô Thành – *Toán rời rạc* – NXBGD. 1996

[4] Đinh Mạnh Tường - Đỗ Xuân Lôi - *Cấu trúc dữ liệu và giải thuật* – NXBTK.1990

[5] Quách Tuấn Ngọc – *Ngôn ngữ lập trình PASCAL* – NXBGD.1985

[6] Trần Đức Huyên - *Các thuật giải trong tin học* – NXBGD.1991