BÀI TOÁN TẬP TƯƠNG THÍCH CỰC ĐẠI

I. Phát biểu bài toán

Cho đơn đồ thị vô hướng G = (V, E) trong đó V là tập đỉnh, E là tập cạnh. Giả sử M là một tập con của E. Với mỗi v ∈ Vta kí hiệu $deg_M(v)$ là số cạnh thuộc M nhận v là một đầu mút.

Bài toán : Cho đơn đồ thị vô hướng G= (V, E), với mỗi đỉnh v ∈ V ta cho tương ứng với một số nguyên dương d(v) gọi là giới hạn bậc của v. Tìm một tập con M của E thoả mãn :

(1) $deg_M(v) \le d(v)$ ∀ v ∈ V.

(2) |M| đạt lớn nhất.

Tập M thoả mãn điều kiện (1) gọi là một tập tương thích của G tương ứng với bộ số d (hay gọi tắt là tập tương thích). Tập M thoả mãn (1) và (2) là nghiệm của bài toán gọi là tập tương thích cực đại.

II. Định lý cơ sở

Ta có nhận xét nếu d(v) = 1 ∀ v ∈ V thì bài toán trên trở thành bài toán cặp ghép cực đại trên đồ thị. Như vậy bài toán cặp ghép là trường hợp riêng của bài toán được nêu. Bài toán cặp ghép cực đại đã được giải quyết bằng giải thuật đường mở của Edmonds. Sau đây ta sẽ nghiên cứu thuật toán cho bài toán đặt ra dựa trên ý tưởng thuật toán đường mở của Edmonds.

Một số khái niệm :

Giả sử M là một tập tương thích của G tương ứng bộ số d.

Một cạnh gọi là *cạnh đậm* nếu nó thuộc M, ngược lại là *cạnh nhạt*.

Với mỗi v ∈ V, ta gọi v là đỉnh *bão hòa* nếu $deg_M(v) = d(v)$, ngược lại ta gọi v là đỉnh *chưa bão hoà*.

Một đường đi gọi là *dây chuyền xen kẽ* nếu nó không lặp cạnh và dọc theo đường đi các cạnh đậm nhạt đan xen nhau (tức là một cạnh đậm rồi đến cạnh nhạt rồi lại cạnh đậm, cạnh nhạt, …).

Một dây chuyền xen kẽ gọi là *đường mở* nếu nó bắt đầu và kết thúc bằng một cạnh mảnh, đồng thời đỉnh đầu và cuối của dây chuyền là hai đỉnh chưa bão hoà khác nhau hoặc trùng nhau nếu đỉnh tương ứng có deg ≤ d - 2.

Dễ thấy vì tính chất đan xen đậm nhạt nên trên đường mở số cạnh đậm nhỏ hơn số cạnh nhạt là 1.

Chú ý : Các khái niệm vừa trình bày ở trên đều phải kèm theo sự tương ứng với M (Ví dụ cạnh đậm tương ứng M, đường mở tương ứng với M), tuy nhiên vì ta chỉ xét đến một tập tương thích M nên có thể được bỏ qua.

Định lý : Tập tương thích M là cực đại khi và chỉ khi không tồn tại đường mở.

Chứng minh :

Trước hết ta thấy rằng nếu tồn tại đường mở thì tập M không phải là cực đại. Thật vậy, giả sử p là một đường mở, nếu trên đường mở ta thay đổi đậm nhạt của các cạnh (đậm thành nhạt và ngược lại) thì tập M' các cạnh đậm trên đồ thị thu được thoả mãn là một tập tương thích. Mặt khác vì trên đường mở số cạnh nhạt hơn số cạnh đậm 1 nên $|M'| = |M| + 1 \Rightarrow$ M không phải là tập tương thích cực đại $\Rightarrow$ Nếu M là tập tương thích cực đại thì không tồn tại đường mở.

Nếu M không phải là cực đại thì sẽ chỉ ra rằng tồn tại một đường mở. Thật vậy, giả sử M' là tập tương thích cực đại $\Rightarrow |M'| > |M|$. Ta xây dựng đồ thị G' = (V, E') trong đó $E' = M \oplus M'$. Vì $|M'| > |M|$ nên G' có số cạnh nhạt (cạnh thuộc M') nhiều hơn số cạnh đậm (cạnh thuộc M). Ta cũng dễ thấy nếu với một đỉnh v bất kì có số cạnh nhạt kề với v nhiều hơn số cạnh đậm thì v phải là một đỉnh chưa bão hoà, vì nếu v là đỉnh bão hoà thì trong tập tương thích M' số cạnh kề với v nhiều hơn $deg_M(v) = d(v)$.

Giả sử G' không có đường mở, ta thực hiện các bước sau đây : Bắt đầu từ một đỉnh chưa bão hoà bất kì đi theo một cạnh nhạt (nếu có), rồi lại đi theo một cạnh đậm, … Trên đường đi là dây chuyền xen kẽ này, đi qua cạnh nào thì ta xoá luôn cạnh đó đi. Ta cứ đi theo dây chuyền xen kẽ này cho đến khi nào không thể đi tiếp được nữa. Mỗi bước ta lại có một dây chuyền xen kẽ. Ta gọi các dây chuyền xen kẽ lần lượt là $p_1, p_2, p_3, \ldots$.

Ta chứng minh bằng qui nạp mệnh đề sau : các dây chuyền xen kẽ p_i đều kết thúc bằng một cạnh đậm.

Mệnh đề đúng với i = 1, vì dây chuyền đầu tiên nếu kết thúc bằng một cạnh nhạt thì đỉnh cuối cùng v của p_1 phải là đỉnh chưa bão hoà vì nếu ngược lại thì p_1 là đường mở trái giả thiết phản chứng. Ta chú ý rằng một đỉnh u không phải là đỉnh bắt đầu hoặc kết thúc của một đường mở thì sau khi đi qua u cả hai cạnh đậm và nhạt tương ứng kề nó trên dây chuyền xen kẽ đều bị xoá đi, tức là hiệu số cạnh đậm và số cạnh nhạt kề u không đổi. Trở lại vấn đề ta thấy nếu dây chuyền p_1 kết thúc tại v bằng cạnh nhạt $\Rightarrow$ v không có cạnh đậm đi ra $\Rightarrow$ v có số cạnh nhạt nhiều hơn số cạnh đậm $\Rightarrow$ v là đỉnh chưa bão hoà $\Rightarrow$ mâu thuẫn.

Giả sử mệnh đề đúng với mọi i = 1, 2 , …, r , ta sẽ chứng minh mệnh đề cũng đúng với i = r + 1. Nếu p_{r+1} kết thúc tại một đỉnh u bằng cạnh nhạt thì u không thể là một đỉnh xuất phát của một trong các dây chuyền $p_1, p_2, \ldots p_{r+1}$ vì các đỉnh đó đều là đỉnh chưa bão hoà; mặt khác u cũng không thể là đỉnh kết thúc của một trong các dây chuyền đó (trừ chính dây chuyền p_{r+1}) vì theo giả thiết qui nạp các đỉnh đó kết thúc dây chuyền bằng các cạnh đậm tức là không thể còn cạnh nhạt kề với các đỉnh đó nữa. Từ đó cho thấy cho đến trước khi dây chuyền p_{r+1} kết thúc tại u thì hiệu số cạnh nhạt, đậm kề u không thay đổi sau các phép xoá, bên cạnh đó dây chuyền kết thúc tại u bằng cạnh nhạt nên u phải có số cạnh nhạt nhiều hơn số cạnh đậm $\Rightarrow$ u là đỉnh chưa bão hoà $\Rightarrow$ p_{r+1} là đường mở. Trái giả thiết phản chứng.

Từ mệnh đề trên, ta thấy rằng sau mỗi bước đi và xoá một dây chuyền thì đỉnh có số cạnh nhạt nhiều hơn số cạnh đậm, vẫn là đỉnh chưa bão hoà. Mặt khác, vì các dây chuyền kết thúc tại cạnh đậm nên số cạnh đậm bằng số cạnh nhạt $\Rightarrow$ số cạnh đậm bị xoá bằng số cạnh nhạt bị xoá, do đó G' luôn có số cạnh nhạt nhiều hơn số cạnh đậm nên luôn tìm được đỉnh có số cạnh nhạt nhiều hơn số cạnh đậm là đỉnh chưa bão hoà để bắt đầu một dây chuyền mới.

Điều náy có nghía là số dây chuyền tìm được là vô hạn, trái với tính hữu hạn của G'. Như vậy, nếu M không phải là tập tương thích cực đại thì ta tồn tại một đường mở.

$\Rightarrow$ Định lý được chứng minh.

Định lý trên đã chỉ ra rằng nếu ta cứ thực hiện các bước tìm đường mở rồi thực hiện tăng tập tương thích (bằng cách thay đổi tính đậm nhạt như đã chỉ ra trong phép chứng minh định lý), cho đến khi không tìm được đường mở nữa thì tập tương thích M có được là cực đại. Đây chính là cơ sở để ta có thể giải bài toán tập tương thích cực đại bằng thuật toán đường mở mà ta sẽ đề cập ở đây.

III. Lý thuyết về dây chuyền xen kẽ - đường mở

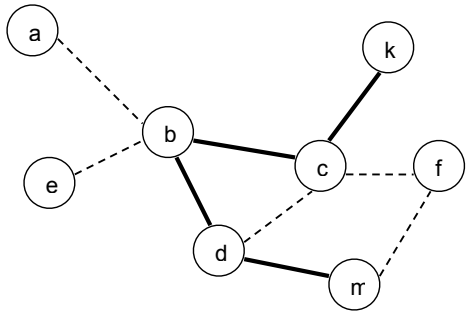
Từ một đỉnh chưa bão hoà bất kì ta chọn làm gốc (root), ta xét tất cả các dây chuyền xen kẽ xuất phát từ gốc. Đường mở xuất phát từ gốc sẽ phải là một trong các dây chuyền xen kẽ này, đặc biệt nếu như $deg_M(root) \leq d(root) - 2$ thì đường mở có thể là một đường khép kín kết thúc tại gốc.

Ở đây ta chỉ xét đến các đỉnh đến được từ root. Trong tập đỉnh đó, ta có những nhận xét sau :

Các đỉnh v mà có thể đến được bằng dây chuyền kêt thúc bởi cạnh đậm, ta gọi là các *đỉnh đậm*.

Các đỉnh u mà có thể đến được bằng dây chuyền kêt thúc bởi cạnh nhạt, ta gọi là các *đỉnh nhạt*.

Có những đỉnh vừa là đậm, vừa là nhạt, ta gọi đó là những *đỉnh hỗn hợp*. Ta coi nút gốc là cũng một đỉnh đậm.



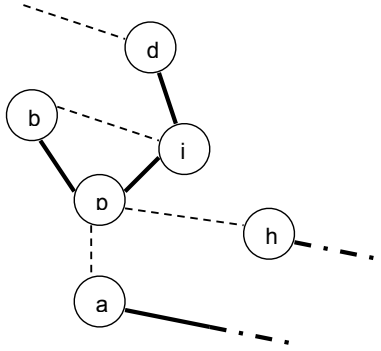
Trong ví dụ trên hình vẽ, với a được chọn làm gốc, thì tập đỉnh đậm gồm a, b, c, d, m, k , các đỉnh nhạt gồm b, c, d, e, f và các đỉnh hỗn hợp gồm b, c, d. Ta chú ý e là đỉnh nhạt vì có dây chuyền xen kẽ (a, b, c, e, d, b, e) kết thúc bởi (b, e) là cạnh nhạt, và đỉnh b được lặp lại hai lần trên đường đi.

Giả sử S là một thành phần liên thông của đồ thị con của G tạo bởi các đỉnh hỗn hợp và S không chứa đỉnh gốc. Xét tất cả các dây chuyền xen kẽ đến S, giả sử x là một dây chuyền có độ dài ngắn nhất và (a, p) là cạnh kết thúc của x. Dễ thấy $a \notin S$, $p \in S$, gọi e = (a,p) và không mất tính tổng quát, giả sử e là cạnh nhạt.

Bổ đề 1 Mọi dây chuyền đến p đều phải qua e.

Chứng minh

Ta thấy mọi dây chuyền đến p bằng cạnh đậm đều phải qua e. Thật vậy nếu ngược giả sử có dây chuyền xen kẽ đến p bằng cạnh đậm không qua e, khi đó ta có thể mở rộng dây chuyền xen kẽ này bằng đi thêm (p, a) $\Rightarrow$ a là đỉnh hỗn hợp kề p $\Rightarrow$ $a \in S$ trái điều đã chỉ ra rằng $a \notin S$.



Ta cũng chỉ ra rằng mọi dây chuyền đến p bằng cạnh nhạt cũng phải kết thúc tại e. Vì p là đỉnh hỗn hợp nên phải tồn tại một dây chuyền xen kẽ kết thúc tại p bằng cạnh đậm,mặt khác theo điều chỉ ra ở trên dây chuyền này phải qua (a, p). Giữa hai lần qua p này ta được chu trình xen kẽ (p,v_1, …, v_k , p). Giả sử có một dây chuyền xen kẽ y kết thúc bằng cạnh nhạt (h, p) trong đó h # a. Nếu y giao với dây chuyền v_1, v_2, …, v_k , thì tại vị trí giao đầu tiên trên y ta hoàn toàn có thể đi thay vì theo y ta đi tiếp theo một hướng nào đó trên dây chuyền v mà vẫn đảm bảo tính chất dây chuyền xen kẽ (hoặc về v_1,p hoặc đến v_k, p) để được dây chuyền đến p bằng cạnh đậm mà không qua e. Điều này trái với phần vừa chứng minh là không có dây chuyền như vậy.

Nếu y không giao với v thì đường đi (y, p, v_k, v_{k-1}, …, v_1, p, a) là dây chuyền kết thúc tại a bằng cạnh mảnh ⇒ a là đỉnh hỗn hợp ⇒ mâu thuẫn.

Bổ đề 2 Nếu u ∈ S thoả mãn mọi dây chuyền xen kẽ đến u đều qua e thì các đỉnh trên dây chuyền từ e đến u cũng thoả mãn tính chất đó.

Chứng minh

Nếu u = p thì điều này đúng theo Bổ đề 1.

Nếu u # p, giả sử z = (p, … u) là dây chuyền xen kẽ từ p đến u. Ta chứng minh bằng phản chứng, giả sử tồn tại trên z những đỉnh có thể đến bằng các dây chuyền không qua e. Trong các dây chuyền đó, ta xét dây chuyền y có độ dài nhỏ nhất, khi đó y không thể có điểm chung với z trừ điểm cuối cùng ∈ z. Mặt khác ta luôn có thể từ y, theo dây chuyền z theo chiều đến u hoặc về p một cách thích hợp để vẫn tạo ra dây chuyền xen kẽ, đến p hoặc u mà không qua e. Điều này trái với a, p chỉ có thể đến được bằng các dây chuyền qua e theo giả thiết và Bổ đề 1.

Trên cơ sở hai Bổ đề ta có định lý sau :

Định lý 1

Nếu S không chứa đỉnh gốc thì tồn tại duy nhất một cạnh e nối một đỉnh trong S với một đỉnh ngoài S thoả mãn mọi dây chuyền xen kẽ đến S đều qua e.

Chứng minh

Ta sẽ chứng minh e = (a, p) chính là cạnh thoả mãn định lý. Gọi A là tập các đỉnh v ∈ S mà mọi dây chuyền tới v đều qua e.

Nếu A = S thì định lý đúng hiển nhiên.

Nếu A # S, vì S liên thông nên phải có u ∈ A, v ∈ (S - A) sao cho (u, v) là cạnh. Không mất tính tổng quát, ta có thể coi (u, v) là cạnh nhạt. Xét dây chuyền xen kẽ z đến v bằng cạnh đậm, Khi đó nếu z không chứa cạnh (u,v) thì z + (u, v) sẽ là một dây chuyền xen kẽ đến u. Theo Bổ đề 2, mọi đỉnh trên đường đi đều thuộc A, trong đó có v ⇒ v ∈ A ⇒ Mâu thuẫn . Ngược lại nếu z chứa (u,v) thì không thể có dây chuyền đến v không qua e, vì khi đó u cũng có thể đi theo dây chuyền này được, tức là có thể đến u không qua e ⇒ Mâu thuẫn. Cả hai trường hợp đều dẫn đến mâu thuẫn, tức là A = S ⇒ e thoả mãn định lý.

Mặt khác, ta cũng có thể dễ dàng nhận thấy e là duy nhất có tính chất đó.

Từ định lý 1, ta suy ra một hệ quả rất quan trọng là không thể có dây chuyền đi ngang qua S, mà chỉ có thể đi vào qua cạnh e duy nhất, rồi đi ra qua từ một đỉnh nào đó thuộc S. Thật vậy, nếu ngược lại thì một đỉnh u nào đó thuộc S sẽ có dây chuyền đến không qua e, trái định lý 1. Khi trong S không có đỉnh chưa bão hoà để thành đường mở thì ta không cần quan tâm đến từng quan hệ các đỉnh trong S, mà chỉ cần biết các cạnh nối ra.

Ta cũng dễ thấy, mọi dây chuyền đi qua S đều có thể chỉ ra một dãy đỉnh cụ thể trên dây chuyền xen kẽ khi nó đi xuyên qua S. Mặt khác, một dãy đỉnh như thế sẽ cho một đường đi ra tương ứng. Ngoài ra, không thể có dây chuyền nào vào S được hai lần. Mỗi thành phần liên thông hỗn hợp S có thể qui về một đỉnh. Đặc biệt, trên đồ thị mới thu được sẽ không có chu trình lẻ do mỗi S phải là cả một thành phần liên thông của các đỉnh hỗn hợp, nên không thể nhóm thêm, tức là không thể có chu trình xen kẽ độ dài lẻ. Như vậy, mọi dây chuyền

xen kẽ trong đồ thị mới sẽ là những đường đi đơn, và không hình thành chu trình lẻ. Điều này cho phép ta có thể tím đường mở giống như tìm trên đồ thị hai phía.

IV. Thuật toán đường mở

Từ định lý ở trên ta đã chỉ ra có thể biến đổi đồ thị bằng cách co mỗi thành phần liên thông của các đỉnh hỗn hợp S về một đỉnh. Khi đó mọi dây chuyền xen kẽ đều là các dây chuyền đơn, và việc tìm đường mở trở nên đơn giản hơn rất nhiều so với đồ thị ban đầu. Bây giờ lại nảy sinh hai vấn đề :

+ Chỉ ra các thành phần liên thông của các đỉnh hỗn hợp (1).

+ Trên đồ thị với các tập S được co lại, ta tìm được đường mở. Vậy thì khi đi qua một điểm co S như thế, ta sẽ tìm lại đường tương ứng đi qua S như thế nào ? (2).

Vấn đề (1) ta có thể giải quyết nhờ tính chất sau : Mỗi lần tìm thấy một đỉnh có hai dây chuyền xen kẽ đến bằng hai cạnh khác nhau thì ta có thể mở rộng tập S. Do đó ta có thể xây dựng thuật toán như sau :

Mỗi lần phát hiện có hai dây chuyền xen kẽ đến u bằng cạnh đậm và nhạt khác nhau thì ta kết nạp thêm các đỉnh trên hai dây chuyền kể từ đỉnh chung p tính từ u về gốc. Mỗi lần tìm được như vậy, ta lại co tất cả các đỉnh mới tìm được của S vào thành một đỉnh. Như thế, sau mỗi bước, việc co lại làm mất chu trình con mới tạo ra, tức là ở bước sau ta không cần quan tâm đến chu trình này nữa vì nó đã là một đỉnh. Việc mở rộng kết thúc cho đến khi ta không còn thấy một đỉnh được đến bằng hai dây chuyền kết thúc bằng hai cạnh đậm nhạt khác nhau.

Vấn đề (2) cũng sẽ được giải quyết một cách đơn giản bằng hai mảng Trace1 và Trace2 với ý nghĩa : Trace1(u) lưu lại vết - đỉnh đứng trước u trong dây chuyền đến u bằng cạnh nhạt, Trace2(u) lưu lại đỉnh đứng trước u trong dây chuyền xen kẽ đến u bằng cạnh đậm.

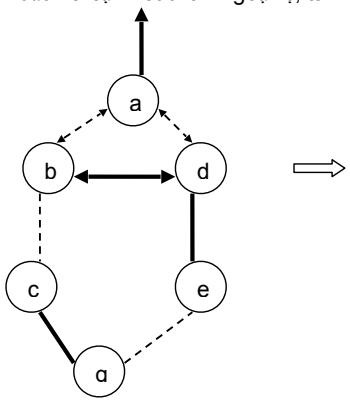
Nếu trong quá trình tìm kiếm, từ u ta đến một đỉnh v mới chưa thăm thì Trace tương ứng của v sẽ có giá trị u. Còn khi tìm được đỉnh u đến bằng hai dây chuyền xen kẽ bằng hai cạnh khác màu thì dùng các mảng Trace hiện có ta có thể tìm lại được hai dây chuyền xen kẽ đến u. Gọi p là đỉnh chung đầu tiên của hai dây chuyền này nếu xét dọc từ u về gốc. Hai đoạn (p, u) và (p, u) tạo thành một chu trình khép kín có dạng (p, … , u , … p). Dọc các đỉnh trên chu trình này, ta xây dựng chu trình xen kẽ tương ứng trên G bằng cách với mỗi s trên chu trình là đỉnh co của một tập S nào đó, ta thay (a, s, b) bằng dây chuyền di qua S có dạng (a, v_1, v_2, … v_k, b).

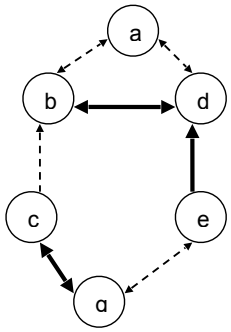
Sau phép thế, có thể ta không còn nhận được chu trình nữa vì p khi nở ra sẽ cho hai đỉnh khác nhau, tổng quát dây chuyền nhận được có dạng p_1, v_1, …, v_m, p_2. Dọc theo các cạnh, ta thiết lập lại trace như sau :

gặp cạnh (u, v) đậm : Trace2(u) = v và Trace2(v) = u

(u, v) nhạt Trace1(u) = v và Trace1(v) = u

Với cạnh đầu và cạnh cuối thì ngoại lệ, ta không biến đổi Trace với các p_1 và p_2.





Ví dụ ở hình vẽ trên, tập p cho hai đỉnh tương ứng b và d. Dây chuyền xen kẽ tìm được là (b, c, g, e, d). Ta đặt trace như hình vẽ.
Trace1(c) = b là vết trong dây chuyền đến c bằng cạnh nhạt.
Trace2(c) = g là vết trong dây chuyền đến c bằng cạnh đậm
Trace1(g) = e là vết trong dây chuyền đến g bằng cạnh nhạt.
Trace2(g) = c là vết trong dây chuyền đến g bằng cạnh đậm.
Trace1(e) = g là vết trong dây chuyền đến e bằng cạnh nhạt.
Trace2(e) = d là vết trong dây chuyền đến e bằng cạnh đậm.

Bằng phép lưu vết này, ta có thể dễ dàng tìm lại dây chuyền xen kẽ từ một đỉnh u bất kì về gốc bằng sử dụng Trace1 và Trace2 đan xen nhau

```
Repeat
   u = Trace1(v)
   Done1(u, v);    xử lý với cạnh nhạt
   v = u
```

u = Trace2(v)
Done2(u, v) xử lý với cạnh đậm
v = u
Until dừng = True

Thuật toán đường mở gồm hai phần : Tìm các tập S rồi co lại thành một đỉnh duy nhất. Đây là những đỉnh đặc biệt, từ nó này có thể đi ra bằng cả cạnh đậm và nhạt, cho dù được đến bằng cạnh đậm hay nhạt. Cuối cùng ta thu được đồ thị không có đỉnh hỗn hợp, tức là chỉ có các dây chuyền đơn. Khi đó, ta có thể dễ dàng tìm đường mở.

Chú ý : Trong phép xây dựng các tập S đã bao hàm phép tìm đường. Do đó ta có thể kết hợp với tìm đường mở. Cách này có ưu điểm là trong quá trình xây dựng các thành phần hỗn hợp, khi tìm được đường mở rồi là ta dừng ngay lại vì mục đích đã đạt được,

Đường mở có thể có hai đỉnh đầu và cuối trùng nhau nên toàn bộ đường mở sẽ giới nội trong tập hỗn hợp S chứa đỉnh xuất phát. Do đó, ta cần chú ý điều này.

Về nguyên tắc, ta phải thử tất cả các đỉnh là đỉnh gốc để kiểm tra sự tồn tại đường mở. Tuy nhiên, ta chỉ cần thử đối với từng đỉnh mà thôi, tức là nếu từ u không tìm thấy đường mở thì các bước sau cũng sẽ không có đường mở từ u (dễ dàng chứng minh nhận xét này), do đó ta không phải tìm lại từ u nữa.

V. Chương trình mẫu

Các bạn có thể xem chương trình mẫu sau viết bằng C để hiểu rõ hơn thuật toán.

```
#include <stdio.h>
#include <conio.h>
#define InputFile  "compset.in"
#define OutputFile "compset.out"
const max  = 501;
const maxe = 10000;
struct TEdge
 {
  int node;
  char property;
 } E[maxe] , Queue[2*max];

int P[max] , Trace[2][max] , Free[max] ,
    Blossom[max] , MaxDeg[max];
int n, first , last, finish , count;

void InputData(void)
{
 int Count[max];
 int m , u , v ;
 FILE *f;
 f = fopen(InputFile , "r");
 fscanf(f , "%d%d" , &n , &m);
 for (u=1; u<=n ; u++) fscanf(f , "%d" , &MaxDeg[u]);
 for (u=1; u<=n ; u++) Count[u] = 0;
```

```
 for (; m > 0 ;m--)
  {
   fscanf(f , "%d%d" , &u , &v);
   Count[u]++; Count[v]++;
  }
 fclose(f);
 P[0] = 0;
 for (u=1; u<=n ; u++)
  {
   P[u] = P[u-1] + Count[u];
   Count[u] = P[u-1];
  }
 f = fopen(InputFile , "r");
 fscanf(f , "%d%d" , &n , &m);
 for (u=1; u<=n ; u++) fscanf(f,"%d" , &v);
 for (; m>0 ; m--)
  {
   fscanf(f,"%d%d" , &u , &v);
   E[Count[u]++].node = v;
   E[Count[v]++].node = u;
  }   /*ta luu danh sách cạnh bằng cấu trúc forward star */
 fclose(f);
}

void Swap(int *a , int *b)
{
 int temp;
 temp = *a;
 *a = *b;
 *b = temp;
}

void Sort(int l , int r)
{
 int i=l,j=r;
 int Key = (E[i].node + E[j].node) >> 1;
 for (;i <= j;)
  {
   for (; E[i].node < Key ; i++);
   for (; E[j].node > Key ; j--);
   if (i <= j)
    {
     Swap(&E[i].node , &E[j].node);
     i++;  j--;
    }
  }
 if (i <= r) Sort(i,r);
 if (l <= j) Sort(l,j);
}

void Prepare(void)
{
 int i;
 for (i=1; i<=n; i++) Sort(P[i-1] , P[i]-1);
 for (i=0; i<P[n] ; i++) E[i].property = 0;
 count = 0;
}
```

```
void Push(int p_node , int p_property)
{
 last++;
 Queue[last].node = p_node;
 Queue[last].property = p_property;
 if ((p_property) && (MaxDeg[p_node])) finish = p_node;
}

void Get(int *p_node , int *p_property)
{
 first++;
 *p_node = Queue[first].node;
 *p_property = Queue[first].property;
}

void Init(int start)
{
 int i;
 for (i=1; i<=n ; i++)
  {
   Trace[0][i] = 0;
   Trace[1][i] = 0;
   Blossom[i] = i;
  }
 first = 0; last = 0;
 finish = 0;
 MaxDeg[start]--;
 Push(start , 0);
 Trace[1][start] = -1;
}

int FindRoot(int u , int v , int property)
{
 int r;
 for (r=1; r <= n ; r++) Free[r] = 0;
 r = property;
 for (;u > 0;)
  {
   r = !r;
   Free[Blossom[u]] = 1;
   u = Trace[r][u];
  }
 r = property;
 for (; v > 0 ;)
  {
   r = !r;
   if (Free[Blossom[v]]) break;
   v = Trace[r][v];
  }
 return Blossom[v];
}

void Connect(int u , int root , int property)
{
  int v;
  if (Blossom[u] == root) goto Exit;
  if (!Trace[property][u]) Push(u,!property);
  for (;;)
```

```
   {
    property = !property;
    Free[Blossom[u]] = 1;
    v = Trace[property][u];
    if (Blossom[v] == root) break;
    if (!Trace[property][v]) Push(v, !property);
    Trace[property][v] = u;
    u = v;
   }
  if (!Trace[property][v])
   {
    Trace[property][v] = u;
    Push(v , !property);
   }
  Exit:
}

void ShrinkBlossom(int u , int v , int property)
{    /* Sau khi tím thấy chu trình xen kẽ, ta mở rộng S*/
 int root , i;
 root = FindRoot(u,v,property);
 for (i=1 ;i <= n ; i++) Free[i] = 0;
 Connect(u , root, property); /* Cập nhật Trace */
 Connect(v , root, property); /* Cập nhật Trace */
 if (Blossom[u] != root) Trace[property][u] = v;
 if (Blossom[v] != root) Trace[property][v] = u;
 for (i=1; i<=n; i++)
  if (Free[Blossom[i]]) Blossom[i] = root;
 /* Phép co tất cả về một đỉnh duy nhất mang nhãn root*/
}

void Done(int u , int v , int property)
{
 if ( (Blossom[u] == Blossom[v]) ||
      (Trace[property][u] == v) ) goto Exit;

 if (Trace[!property][v])
  {
    ShrinkBlossom(u,v,property);
    goto Exit;
  }
 if (!Trace[property][v])
  {
   Trace[property][v] = u;
   Push(v , !property);
  }
 Exit:
}

int FindAugmentingPath(int start)  /* Tìm đường mở */
{
 int u , i , property;
 Init(start);
 for (;first < last;)
  {
   Get(&u , &property);
   for (i=P[u-1]; i < P[u] ; i++)
    if (E[i].property == property)
```

```
     {
      Done(u , E[i].node , property);
      if (finish) return 1;
     }
  }
 MaxDeg[start]++;
 return 0;
}

int Search(int u , int v)
{
 int l , r , mid;
 l = P[u-1]; r = P[u]-1;
 for (;l < r;)
  {
   mid = (l+r) >> 1;
   if (v > E[mid].node) l = mid+1;
   else r = mid;
  }
 return l;
}

void Replace(int u , int v , int p_property)
{
 int p;
 p = Search(u,v);
 if (E[p].property == p_property) printf("Wrong!\n");
 E[p].property = p_property;
}

void Enlarge(void)
{
 int u , v , property;
 v = finish;
 MaxDeg[v]--;
 property = 0;
 count++;
 for (;v > 0;)
  {
   u = Trace[property][v];
   if (u < 0) break;
   property = !property;
   Replace(u,v,property);
   Replace(v,u,property);
   v = u;
  }
}

void Process(void)
{
 int i;
 Prepare();
 for (i=1; i<=n ; i++)
  for (; (MaxDeg[i]) && (FindAugmentingPath(i));) Enlarge();
}

void OutputResult(void)
{
```

```
 FILE *f;
 int u , i;
 f = fopen(OutputFile , "w");
 fprintf(f,"%d\n" , count);
 for (u=1; u<n ; u++)
  for (i=P[u-1] ; i < P[u] ; i++)
   if ( (E[i].property) && (E[i].node > u) )
    fprintf(f , "%d %d\n" , u , E[i].node);
 fclose(f);
}

void main()
{
 InputData();
 Process();
 OutputResult();
}
```

VI. Một số bài tập áp dụng

Bài toán 1 Cho đồ thị G, chỉ ra một đồ thị con G' = (V, E') của G với bậc của các đỉnh là những số cố định cho trước.

Rõ ràng bài toán này có thể đưa về tìm tập tương thích cực đại với d là những bậc cố định tương ứng cho trước.

Bài toán 2 Cho đồ thị G, hãy chọn ra tập $E' \subset E$, gồm ít nhất cạnh sao cho trên đồ thị con G' = (V, E') thoả mãn $deg(u) \geq d(u)$ trong đó d(u) là các số cố định cho trước.

Nếu ta cho đối ngẫu với bai toán tìm tập M = E - E' thì ta có thể đưa về bài toán tập tương thích cực đại.