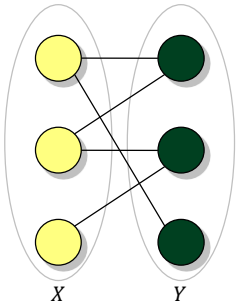
Chương 10. Bộ ghép cực đại trên đồ thị hai phía

10.1. Đồ thị hai phía

Đồ thị vô hướng $G = (V, E)$ được gọi là đồ thị hai phía nếu tập đỉnh V của nó có thể chia làm hai tập con rời nhau: X và Y sao cho mọi cạnh của đồ thị đều nối một đỉnh thuộc X với một đỉnh thuộc Y. Khi đó người ta còn ký hiệu $G = (X \cup Y, E)$. Để thuận tiên trong trình bày, ta gọi các đỉnh thuộc X là các X_đỉnh và các đỉnh thuộc Y là các Y_đỉnh.



Hình 10-1. Đồ thị hai phía

Một đồ thị vô hướng là đồ thị hai phía nếu và chỉ nếu từng thành phần liên thông của nó là đồ thị hai phía. Để kiểm tra một đồ thị vô hướng liên thông có phải đồ thị hai phía hay không, ta có thể sử dụng một thuật toán tìm kiếm trên đồ thị (BFS hoặc DFS) bắt đầu từ một đỉnh s bất kỳ. Đặt:

$$X := \{\text{tập các đỉnh đến được từ } s \text{ qua một số chẵn cạnh}\}$$
$$Y := \{\text{tập các đỉnh đến được từ } s \text{ qua một số lẻ cạnh}\}$$

Nếu tồn tại cạnh của đồ thị nối hai đỉnh $\in X$ hoặc hai đỉnh $\in Y$ thì đồ thị đã cho không phải đồ thị hai phía, ngược lại đồ thị đã cho là đồ thị hai phía với cách phân hoạch tập đỉnh thành hai tập X, Y ở trên.

Đồ thị hai phía gặp rất nhiều mô hình trong thực tế. Chẳng hạn quan hệ hôn nhân giữa tập những người đàn ông và tập những người đàn bà, việc sinh viên chọn trường, thầy giáo chọn tiết dạy trong thời khoá biểu v.v...

10.2. Bài toán

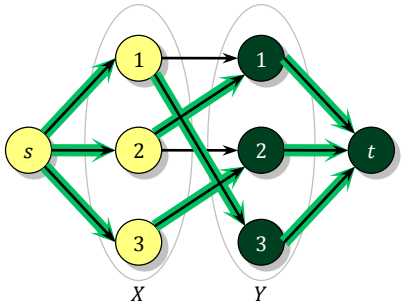
Cho đồ thị hai phía $G = (X \cup Y, E)$. Một *bộ ghép (matching)* của G là một tập các cạnh đôi một không có đỉnh chung. Có thể coi một bộ ghép là một tập $M \subseteq E$ sao cho trên đồ thị $(X \cup Y, M)$, mỗi đỉnh có bậc không quá 1.

Vấn đề đặt ra là tìm *một bộ ghép lớn nhất (maximum matching)* (có nhiều cạnh nhất) trên đồ thị hai phía cho trước.

10.3. Mô hình luồng

Ta xây dựng mạng G' từ đồ thị G bằng cách định hướng các cạnh của G thành cung từ X sang Y. Thêm vào đỉnh phát s và các cung nối từ s tới các X_đỉnh, thêm vào đỉnh thu t và các cung nối từ các Y_đỉnh tới t, sức chứa của tất cả các cung được đặt bằng 1.

Xét một luồng trên mạng G' có luồng trên các cung là số nguyên, khi đó có thể thấy rằng những cung có luồng bằng 1 từ X sang Y sẽ tương ứng với một bộ ghép trên G. Bài toán tìm bộ ghép cực đại trên G có thể giải quyết bằng cách tìm luồng nguyên cực đại trên G' và chọn ra các cung mang luồng 1 nối từ X sang Y.



Hình 10-2. Mô hình luồng của bài toán tìm bộ ghép cực đại trên đồ thị hai phía.

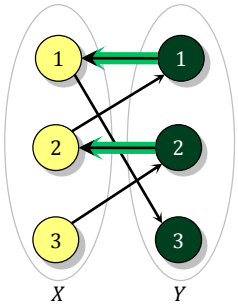
Chúng ta sẽ phân tích một số đặc điểm của đường tăng luồng trong trường hợp này để tìm ra một cách cài đặt đơn giản hơn.

Xét đồ thị hai phía $G = (X \cup Y, E)$ và một bộ ghép M trên G.

- Những đỉnh thuộc M gọi là những *đỉnh đã ghép* (*matched vertices*), những đỉnh không thuộc M gọi là những *đỉnh chưa ghép* (*unmached vertices*).
- Những cạnh thuộc M gọi là những cạnh đã ghép, những cạnh không thuộc M được gọi là những cạnh chưa ghép.
- Nếu định hướng lại những cạnh của đồ thị thành cung: Những cạnh chưa ghép định hướng từ X sang Y, những cạnh đã ghép định hướng ngược lại từ Y về X. Trên đồ thị định hướng đó, một đường đi được gọi là *đường pha (alternating path)* và một đường đi từ một X_đỉnh chưa ghép tới một Y_đỉnh chưa ghép gọi là một *đường mở* (*augmenting path*).

Dọc trên một đường pha, các cạnh đã ghép và chưa ghép xen kẽ nhau. Đường mở cũng là một đường pha, đi qua một số lẻ cạnh, trong đó số cạnh chưa ghép nhiều hơn số cạnh đã ghép đúng một cạnh.

> Ví dụ với đồ thị hai phía trong Hình 10-3 và một bộ ghép $\{(x_1, y_1), (x_2, y_2)\}$. Đường đi $\langle x_3, y_2, x_2, y_1 \rangle$ là một đường pha, đường đi $\langle x_3, y_2, x_2, y_1, x_1, y_3 \rangle$ là một đường mở.



Hình 10-3. Đồ thị hai phía và các cạnh được định hướng theo một bộ ghép

Đường mở thực chất là đường tăng luồng với giá trị thặng dư 1 trên mô hình luồng. Định lý 9-11 (mối quan hệ giữa luồng cực đại, đường tăng luồng và lát cắt hẹp nhất) đã chỉ ra rằng điều kiện cần và đủ để một bộ ghép M là bộ ghép cực đại là không tồn tại đường mở ứng với M.

Nếu tồn tại đường mở P ứng với bộ ghép M, ta mở rộng bộ ghép bằng cách: dọc trên đường P loại bỏ những cạnh đã ghép khỏi M và thêm những cạnh chưa ghép vào M. Bộ ghép mới thu được sẽ có lực lượng nhiều hơn bộ ghép cũ đúng một cạnh. Đây thực chất là phép tăng luồng dọc trên đường P trên mô hình luồng.

10.4. Thuật toán đường mở

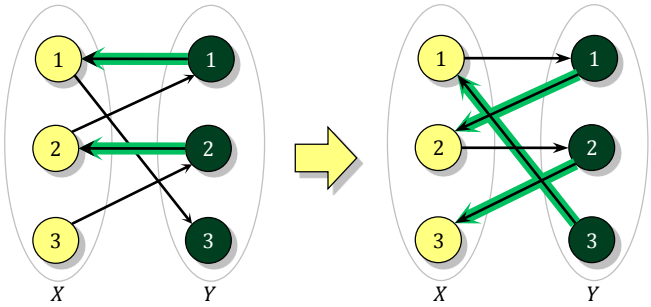
Từ mô hình luồng của bài toán, chúng ta có thể xây dựng được thuật toán tìm bộ ghép cực đại dựa trên cơ chế tìm đường mở và tăng cặp: Thuật toán khởi tạo một bộ ghép bất kỳ trước khi bước vào vòng lặp chính. Tại mỗi bước lặp, đường mở (thực chất là một đường đi từ một X_đỉnh chưa ghép tới một Y_đỉnh chưa ghép) được tìm bằng BFS hoặc DFS và bộ ghép sẽ được mở rộng dựa trên đường mở tìm được.

```
M := «Một bộ ghép bất kỳ, chẳng hạn: ∅»;
while «Tìm được đường mở P» do
  «Dọc trên đường P:
     - Loại bỏ những cạnh đã ghép khỏi M
     - Thêm những cạnh chưa ghép vào M
  »
```

Ví dụ với đồ thị trong Hình 10-3 và bộ ghép $M = \{(x_1, y_1), (x_2, y_2)\}$, thuật toán sẽ tìm được đường mở:

$$x_3 \dashrightarrow y_2 \underbrace{\rightarrow}_{\in M} x_2 \dashrightarrow y_1 \underbrace{\rightarrow}_{\in M} x_1 \dashrightarrow y_3$$

Dọc trên đường mở này, ta loại bỏ hai cạnh (y_2, x_2) và (y_1, x_1) khỏi bộ ghép và thêm vào bộ ghép ba cạnh (x_3, y_2), (x_2, y_1), (x_1, y_3), được bộ ghép mới 3 cạnh. Đồ thị với bộ ghép mới không còn đỉnh chưa ghép (không còn đường mở) nên đây chính là bộ ghép cực đại (Hình 10-4).



Hình 10-4. Mở rộng bộ ghép

10.5. Cài đặt

Chúng ta sẽ cài đặt thuật toán tìm bộ ghép cực đại trên đồ thị hai phía $G = (X \cup Y, E)$, trong đó $|X| = p$, $|Y| = q$ và $|E| = m$. Các X_đỉnh được đánh số từ 1 tới p và các Y_đỉnh được đánh số từ 1 tới q. Khuôn dạng Input/Output như sau:

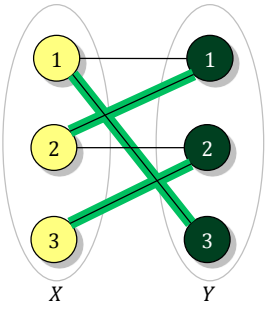
Input

- Dòng 1 chứa ba số nguyên dương p, q, m lần lượt là số X_đỉnh, số Y_đỉnh và số cạnh của đồ thị hai phía. ($p, q \leq 10^4; m \leq 10^6$).
- m dòng tiếp theo, mỗi dòng chứa hai số nguyên dương i, j tương ứng với một cạnh (x_i, y_j) của đồ thị.

Output

Bộ ghép cực đại trên đồ thị.

Sample Input	Sample Output
3 3 5 3 2 2 2 2 1 1 3 1 1	1: x[2] - y[1] 2: x[3] - y[2] 3: x[1] - y[3]



10.5.1. Biểu diễn đồ thị hai phía và bộ ghép

Đồ thị hai phía $G = (X \cup Y, E)$ sẽ được biểu diễn bằng cách danh sách kề của các X_đỉnh. Cụ thể là mỗi đỉnh $x \in X$ sẽ được cho tương ứng với một danh sách các Y_đỉnh kề với x.

Bộ ghép trên đồ thị hai phía được biểu diễn bởi mảng $match[1 \dots ny]$, trong đó $match[j]$ là chỉ số của X_đỉnh ghép với đỉnh y_j. Nếu y_j là đỉnh chưa ghép, ta gán $match[j] := 0$.

10.5.2. Tìm đường mở

Đường mở thực chất là một đường đi từ một X_đỉnh chưa ghép tới một Y_đỉnh chưa ghép trên đồ thị định hướng. Ta sẽ tìm đường mở tại mỗi bước bằng thuật toán DFS:

Bắt đầu từ một đỉnh $x \in X$ chưa ghép, trước hết ta đánh dấu các Y_đỉnh bằng mảng $avail[1 \dots q]$ trong đó $avail[j] =$ True nếu đỉnh $y_j \in Y$ chưa thăm và $avail[j] =$ False nếu đỉnh $y_j \in Y$ đã thăm (chỉ cần đánh dấu các Y_đỉnh).

Thuật toán DFS để tìm đường mở xuất phát từ x được thực hiện bằng một thủ tục đệ quy $Visit(x)$, thủ tục này sẽ quét tất cả những đỉnh $y \in Y$ chưa thăm nối từ X (dĩ nhiên qua một cạnh chưa ghép), với mỗi khi xét đến một đỉnh $y \in Y$, trước hết ta đánh dấu thăm y. Sau đó:

- Nếu y đã ghép, dựa vào sự kiện từ y chỉ đi đến được $match[y]$ qua một cạnh đã ghép hướng từ Y về X, lời gọi đệ quy $Visit(match[y])$ được thực hiện để thăm luôn đỉnh $match[y] \in X$ (thăm liền hai bước).
- Ngược lại nếu y chưa ghép, tức là thuật toán DFS tìm được đường mở kết thúc ở y, ta thoát khỏi dây chuyền đệ quy. Quá trình thoát dây chuyền đệ quy thực chất là lần ngược đường mở, ta sẽ lợi dụng quá trình này để mở rộng bộ ghép dựa trên đường mở.

Để thuật toán hoạt động hiệu quả hơn, ta sử dụng liên tiếp các pha xử lý lô: Ký hiệu X^* là tập các X_đỉnh chưa ghép, mỗi pha sẽ cố gắng mở rộng bộ ghép dựa trên không chỉ một mà nhiều đường mở không có đỉnh chung xuất phát từ các đỉnh khác nhau thuộc X^*. Cụ thể là một pha sẽ khởi tạo mảng đánh dấu $avail[1 \dots q]$ bởi giá trị True, sau đó quét tất cả những đỉnh $x \in X^*$, thử tìm đường mở xuất phát từ x và mở rộng bộ ghép nếu tìm ra đường mở. Trong một pha có thể có nhiều X_đỉnh được ghép thêm.

```
procedure Visit(x∈X); //Thuật toán DFS
begin
  for ∀y: (x, y)∈E do //Quét các Y_đỉnh kề x
    if avail[y] then //y chưa thăm, chú ý (x, y) chắc chắn là cạnh chưa ghép
      begin
        avail[y] := False; //Đánh dấu thăm y
        if match[y] = 0 then Found := True //y chưa ghép, dựng cờ báo tìm thấy đường mở
        else Visit(match[y]); //y đã ghép, gọi đệ quy tiếp tục DFS
        if Found then // Ngay khi đường mở được tìm thấy
          begin
            match[y] := x; //Chỉnh lại bộ ghép theo đường mở
            Exit; //Thoát luôn, lệnh Exit đặt ở đây sẽ thoát cả dây chuyền đệ quy
          end;
      end;
end;

begin //Thuật toán tìm bộ ghép cực đại trên đồ thị hai phía
  «Khởi tạo một bộ ghép bất kỳ, chẳng hạn ∅»;
  X* := «Tập các đỉnh chưa ghép»;
```

```
  repeat //Lặp các pha xử lý theo lô
    Old := |X*|; //Lưu số đỉnh chưa ghép khi bắt đầu pha
    for ∀y∈Y do avail[y] := True; //Đánh dấu mọi Y_đỉnh chưa thăm
    for ∀x∈X* do
      begin
        Found := False; //Cờ báo chưa tìm thấy đường mở
        Visit(x); //Tìm đường mở bằng DFS
        if Found then X* := X* - {x}; //x đã được ghép, loại bỏ x khỏi X*
      end;
  until |X*| = Old; //Lặp cho tới khi không thể ghép thêm
end;
```

💻 BMATCH.PAS ✓ Tìm bộ ghép cực đại trên đồ thị hai phía

```
{$MODE OBJFPC}
program MaximumBipartiteMatching;
const
  maxN = 10000;
  maxM = 1000000;
type
  TAdj = record //Cấu trúc nút trong danh sách kề
    y: Integer; //đỉnh
    link: Integer; //nút kế tiếp
  end;
var
  p, q, m: Integer;
  adj: array[1..maxM] of TAdj;
  head: array[1..maxN] of Integer;
  match: array[1..maxN] of Integer;
  avail: array[1..maxN] of Boolean;
  list: array[1..maxN] of Integer;
  nList: Integer;

procedure Enter; //Nhập dữ liệu
var
  i: Integer;
  x, y: Integer;
begin
  ReadLn(p, q, m);
  FillChar(head[1], p * SizeOf(head[1]), 0);
  for i := 1 to m do
    begin
      ReadLn(x, adj[i].y);
      adj[i].link := head[x]; head[x] := i;
    end;
end;

procedure Init; //Khởi tạo bộ ghép rỗng
var
  i: Integer;
begin
  FillChar(match[1], q * SizeOf(match[1]), 0);
  for i := 1 to p do list[i] := i; //Mảng list chứa các X_đỉnh chưa ghép
  nList := p;
end;
```

```
procedure SuccessiveAugmentingPaths;
var
  Found: Boolean;
  Old, i: Integer;

  procedure Visit(x: Integer); //Thuật toán DFS từ x ∈X
  var
    i: Integer;
  begin
    i := head[x]; //Từ đầu danh sách kề của x
    while i <> 0 do
      with adj[i] do
        begin
          if avail[y] then //y chưa thăm, hiển nhiên (x,y) là cạnh chưa ghép
            begin
              avail[y] := False; //Đánh dấu thăm y
              if match[y] = 0 then Found := True //y chưa ghép thì báo hiệu tìm thấy đường mở
              else Visit(match[y]); //Thăm luôn match[y] ∈X (thăm liền 2 bước)
              if Found then //Tìm thấy đường mở
                begin
                  match[y] := x; //Chỉnh lại bộ ghép
                  Exit; //Thoát dây chuyền đệ quy
                end;
            end;
          i := link; //Chuyển sang đỉnh kế tiếp trong danh sách các đỉnh kề x
        end;
  end;

begin
  repeat
    Old := nList; //Lưu lại số X_đỉnh chưa ghép
    FillChar(avail[1], q * SizeOf(avail[1]), True);
    for i := nList downto 1 do
      begin
        Found := False;
        Visit(list[i]); //Cố ghép list[i]
        if Found then //Nếu ghép được
          begin //Xóa list[i] khỏi danh sách các X_đỉnh chưa ghép
            list[i] := list[nList];
            Dec(nList);
          end;
      end;
  until Old = nList; //Không thể ghép thêm X_đỉnh nào nữa
end;

procedure PrintResult; //In kết quả
var
  j, k: Integer;
begin
  k := 0;
  for j := 1 to q do
    if match[j] <> 0 then
      begin
        Inc(k);
        WriteLn(k, ': x[', match[j], '] - y[', j, ']');
      end;
```

```
end;

begin
  Enter;
  Init;
  SuccessiveAugmentingPaths;
  PrintResult;
end.
```

10.6. Đánh giá

Nếu đồ thị có n đỉnh ($n = p + q$) và m cạnh, do mảng đánh dấu $avail[1 \dots q]$ chỉ được khởi tạo một lần trong pha, thời gian thực hiện của một pha sẽ bằng $O(n + m)$ (suy ra từ thời gian thực hiện giải thuật của DFS).

Các pha sẽ được thực hiện lặp cho tới khi $X^* = \emptyset$ hoặc khi một pha thực hiện xong mà không ghép thêm được đỉnh nào. Thuật toán cần không quá p lần thực hiện pha xử lý lô, nên thời gian thực hiện giải thuật tìm bộ ghép cực đại trên đồ thị hai phía là $O(n^2 + nm)$ trong trường hợp xấu nhất. Còn trong trường hợp tốt nhất, ta có thể tìm được bộ ghép cực đại chỉ qua một lượt thực hiện pha xử lý lô, tức là bằng thời gian thực hiện giải thuật DFS. Cần lưu ý rằng đây chỉ là những đánh giá O lớn về cận trên của thời gian thực hiện. Thuật toán này chạy rất nhanh trên thực tế nhưng hiện tại tôi chưa biết đánh giá nào chặt hơn.

Ý tưởng tìm một lúc nhiều đường mở không có đỉnh chung đã được nghiên cứu trong bài toán luồng cực đại bởi Dinic (Dinic, 1970). Dựa trên ý tưởng này, Hopcroft và Karp (Hopcroft & Karp, An n^(5/2) algorithm for maximum matchings in bipartite graphs, 1973) đã tìm ra thuật toán tìm bộ ghép cực đại trên đồ thị hai phía trong thời gian $O\left(\sqrt{|V|}|E|\right)$. Thuật toán Hopcroft-Karp trước hết sử dụng BFS để phân lớp các đỉnh theo độ dài đường đi ngắn nhất sau đó mới sử dụng DFS trên rừng các cây BFS để xử lý lô tương tự như cách làm của chúng ta ở trên.

Bài tập 10-1

Có p thợ và q việc. Mỗi thợ cho biết mình có thể làm được những việc nào, và mỗi việc khi giao cho một thợ thực hiện sẽ được hoàn thành xong trong đúng 1 đơn vị thời gian. Tại một thời điểm, mỗi thợ chỉ thực hiện không quá một việc.

Hãy phân công các thợ làm các công việc sao cho:

- Mỗi việc chỉ giao cho đúng một thợ thực hiện.
- Thời gian hoàn thành tất cả các công việc là nhỏ nhất. Chú ý là các thợ có thể thực hiện song song các công việc được giao, việc của ai người nấy làm, không ảnh hưởng tới người khác.

Bài tập 10-2

Một bộ ghép M trên đồ thị hai phía gọi là tối đại nếu việc bổ sung thêm bất cứ cạnh nào vào M sẽ làm cho M không còn là bộ ghép nữa.

a) Chỉ ra một ví dụ về bộ ghép tối đại nhưng không là bộ ghép cực đại trên đồ thị hai phía

b) Tìm thuật toán O($|E|$) để xác định một bộ ghép tối đại trên đồ thị hai phía

c) Chứng minh rằng nếu A và B là hai bộ ghép tối đại trên cùng một đồ thị hai phía thì $|A| \leq 2|B|$ và $|B| \leq 2|A|$. Từ đó chỉ ra rằng nếu thuật toán đường mở được khởi tạo bằng một bộ ghép tối đại thì số lượt tìm đường mở giảm đi ít nhất một nửa so với việc khởi tạo bằng bộ ghép rỗng.

Bài tập 10-3 (Phủ đỉnh – Vertex Cover)

Cho đồ thị hai phía $G = (X \cup Y, E)$. Bài toán đặt ra là hãy chọn ra một tập C gồm ít nhất các đỉnh sao cho mọi cạnh $\in E$ đều liên thuộc với ít nhất một đỉnh thuộc C.

Bài toán tìm phủ đỉnh nhỏ nhất trên đồ thị tổng quát là NP-đầy đủ, hiện tại chưa có thuật toán đa thức để giải quyết. Tuy vậy trên đồ thị hai phía, phủ đỉnh nhỏ nhất có thể tìm được dựa trên bộ ghép cực đại.

Dựa vào mô hình luồng của bài toán bộ ghép cực đại, giả sử các cung (X, Y) có sức chứa $+\infty$, các cung (s, X) và (Y, t) có sức chứa 1. Gọi (S, T) là lát cắt hẹp nhất của mạng. Đặt $C = \{T \cap X\} \cup \{S \cap Y\}$.

a) Chứng minh rằng C là một phủ đỉnh

b) Chứng minh rằng C là phủ đỉnh nhỏ nhất

c) Giả sử ta tìm được M là bộ ghép cực đại trên đồ thị hai phía, khi đó chắc chắn không còn tồn tại đường mở tương ứng với bộ ghép M. Đặt:

$$Y^* = \{y \in Y: \exists x \in X \text{ chưa ghép, } x \text{ đến được } y \text{ qua một đường pha}\}$$
$$X^* = \{x \in X: x \text{ đã ghép và đỉnh ghép với } x \text{ không thuộc } Y^*\}$$

Chứng minh rằng $X^* \cup Y^*$ là phủ đỉnh nhỏ nhất .

Bài tập 10-4 (Tập độc lập cực đại)

Cho đồ thị hai phía $G = (X \cup Y, E)$. Bài toán đặt ra là hãy chọn ra một tập I gồm nhiều đỉnh sao cho hai đỉnh bất kỳ của I không kề nhau.

a) Chứng minh rằng nếu I là tập độc lập cực đại thì $|I| = |X| + |Y| - |M|$ với $|M|$ là số cạnh của bộ ghép cực đại

b) Xây dựng thuật toán tìm tập độc lập cực đại trên đồ thị hai phía (Gợi ý: Quy về bài toán tìm phủ đỉnh)

Bài tập 10-5

Cho M là một bộ ghép trên đồ thị hai phía $G = (X \cup Y, E)$. Gọi k là số X_đỉnh chưa ghép. Chứng minh rằng ba mệnh đề sau đây là tương đương:

- M là bộ ghép cực đại.
- G không có đường mở tương ứng với bộ ghép M.
- Tồn tại một tập con A của X sao cho $|N(A)| = |A| - k$. Ở đây $N(A)$ là tập các Y_đỉnh kề với một đỉnh nào đó trong A (*Gợi ý: Chọn A là tập các X_đỉnh đến được từ một X_đỉnh chưa ghép bằng một đường pha*)

Bài tập 10-6 (Định lý Hall)

Cho $G = (X \cup Y, E)$ là đồ thị hai phía có $|X| = |Y|$. Chứng minh rằng G có bộ ghép đầy đủ (bộ ghép mà mọi đỉnh đều được ghép) nếu và chỉ nếu $|A| \leq |N(A)|$ với mọi tập $A \subseteq X$.

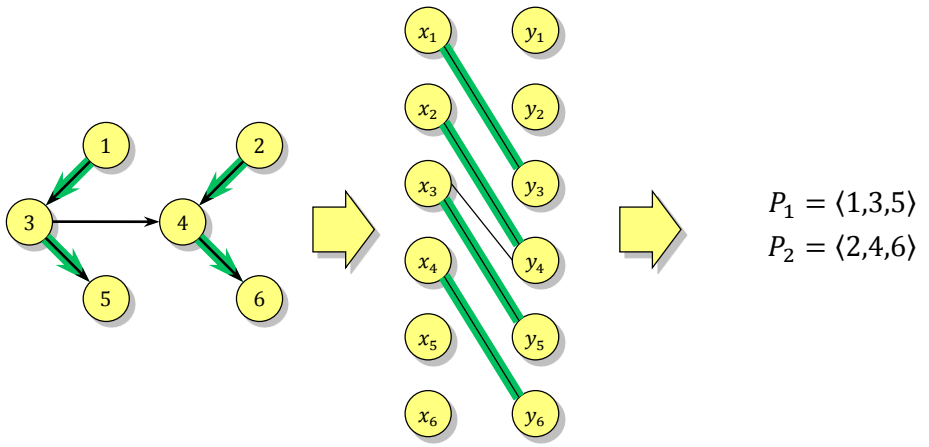
Bài tập 10-7 (Phủ đường tối thiểu)

Cho $G = (V, E)$ là đồ thị có hướng không có chu trình. Một *phủ đường* (p*ath cover*) là một tập P các đường đi trên G thỏa mãn: Với mọi đỉnh $v \in V$, tồn tại duy nhất một đường đi trong P chứa v. Đường đi có thể bắt đầu và kết thúc ở bất cứ đâu, tính cả đường đi độ dài 0 (chỉ gồm một đỉnh). Bài toán đặt ra là tìm *phủ đường tối thiểu* (*minimum path cover*): Phủ đường gồm ít đường đi nhất.

Gọi n là số đỉnh của đồ thị, ta đánh số các đỉnh thuộc V từ 1 tới n. Xây dựng đồ thị hai phía $G' = (X \cup Y, E')$ trong đó:

$$X = \{x_1, x_2, \dots x_n\}$$
$$Y = \{y_1, y_2, \dots y_n\}$$

Tập cạnh E' được xây dựng như sau: Với mỗi cung $(i, j) \in E$, ta thêm vào một cạnh $(x_i, y_j) \in E'$ (Hình 10-5)



Hình 10-5. Bài toán tìm phủ đường tối thiểu trên DAG có thể quy về bài toán bộ ghép cực đại trên đồ thị hai phía.

Gọi M là một bộ ghép trên G'. Khởi tạo P là tập n đường đi, mỗi đường đi chỉ gồm một đỉnh trong G, khi đó P là một phủ đường. Xét lần lượt các cạnh của bộ ghép, mỗi khi xét tới cạnh (x_i, y_j) ta đặt cạnh (i, j) nối hai đường đi trong P thành một đường...Khi thuật toán kết thúc, P vẫn là một phủ đường.

a) Chứng minh tính bất biến vòng lặp: Tại mỗi bước khi xét tới cạnh $(x_i, y_j) \in M$, cạnh $(i, j) \in E$ chắc chắn sẽ nối hai đường đi trong P: một đường đi kết thúc ở i và một đường đi khác bắt đầu ở j. Từ đó chỉ ra tính đúng đắn của thuật toán. (*Gợi ý:* mỗi khi xét tới cạnh $(x_i, y_j) \in M$ và đặt cạnh (i, j) nối hai đường đi của P thành một đường thì $|P|$ giảm 1. Vậy khi thuật toán trên kết thúc, $|P| = n - |M|$, tức là muốn $|P| \to \min$ thì $|M| \to \max$).

b) Viết chương trình tìm phủ đường cực tiểu trên đồ thị có hướng không có chu trình.

c) Chỉ ra ví dụ để thấy rằng thuật toán trên không đúng trong trường hợp G có chu trình.

d) Chứng minh rằng nếu tìm được thuật toán giải bài toán tìm phủ đường cực tiểu trên đồ thị tổng quát trong thời gian đa thức thì có thể tìm được đường đi Hamilton trên đồ thị đó (nếu có) trong thời gian đa thức. (Lý thuyết về độ phức tạp tính toán đã chứng minh được rằng trên đồ thị tổng quát, bài toán tìm đường đi Hamilton là NP-đầy đủ và bài toán tìm phủ đường cực tiểu là NP-khó. Có nghĩa là một thuật toán với độ phức tạp đa thức để giải quyết bài toán phủ đường cực tiểu trên đồ thị tổng quát sẽ là một phát minh lớn và đáng ngạc nhiên).

Bài tập 10-8

Tự tìm hiểu về thuật toán Hopcroft-Karp. Cài đặt và so sánh tốc độ thực tế với thuật toán trong bài.

Bài tập 10-9 (Bộ ghép cực đại trên đồ thị chính quy hai phía)

Một đồ thị vô hướng $G = (V, E)$ gọi là *đồ thị chính quy bậc k* (k-*regular graph*) nếu bậc của mọi đỉnh đều bằng k. Đồ thị chính quy bậc 0 là đồ thị không có cạnh nào, đồ thị chính quy bậc 1 thì các cạnh tạo thành bộ ghép đầy đủ, đồ thị chính quy bậc 2 có các thành phần liên thông là các chu trình đơn.

a) Chứng minh rằng đồ thị hai phía $G = (X \cup Y, E)$ là đồ thị chính quy thì $|X| = |Y|$.

b) Chứng minh rằng luôn tồn tại bộ ghép đầy đủ trên đồ thị hai phía chính quy bậc k ($k \geq 1$).

c) Tìm thuật toán $O(|E| \log |E|)$ để xác định một bộ ghép đầy đủ trên đồ thị chính quy hai phía bậc $k \geq 1$.