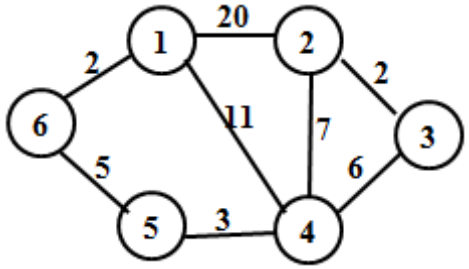
THUẬT TOÁN BELLMAN - FORD

1. Khái niệm đường đi và đường đi ngắn nhất.

Cho đồ thị G=<V,E>, ta có khái niệm đường đi và đường đi ngắn nhất trong đồ thị như sau:

1.1 Đường đi.

Một dãy các đỉnh $P =< p_0, p_1, ..., p_k >$ sao cho $(p_{i-1}, p_i) \in E, \forall i : 1 \le i \le k$ được gọi là một đường đi (path), đường đi này gồm $k+1$ đỉnh $p_0, p_1, ..., p_k$ và k cạnh $(p_0, p_1), (p_1, p_2), ..., (p_{k-1}, p_k)$. Nếu có 1 đường đi như trên thì ta nói p_k đến được từ p_0 hay p_0 đến được p_k. Đỉnh p_0 được gọi là đỉnh đầu và đỉnh p_k được gọi là đỉnh cuối của đường đi P. Các đỉnh $p_1, p_2, ..., p_{k-1}$ được gọi là đỉnh trong của đường đi P.



Ta có thể liệt kê một số đường đi trong đồ thị ở trên từ đỉnh 3 đến đỉnh 6:

P_1= (3, 2, 1, 6);

P_2= (3, 2, 4, 1, 6);

P_3= (3, 2, 1, 4, 5, 6);

P_4= (3, 4, 1, 2, 4, 5, 6);

P_5= (3, 4, 5, 6)

Trong đồ thị có trọng số, trọng số của đường đi sẽ bằng tổng các trọng số của cạnh/cung mà nó đi qua. Trọng số của các đường đi kể trên là:

P_1: 2+20+2=24

P_2: 2+7+11+2=22

P_3: 2+20+11+3+5=31

P_4: 6+11+20+7+3+5=52

P_5: 6+3+5=14

1.2 Đường đi ngắn nhất.

Đường đi $P =< p_0, p_1, ..., p_k >$ được gọi là đường đi ngắn nhất khi trọng số của P là nhỏ nhất trong tất cả các đường đi từ p_0 đến p_k.

Trong các đường đi từ 3 đến 6 ở đồ thị trên, đường đi ngắn nhất là P_5= (3, 4, 5, 6) có trọng số là 14.

Nhận xét:

+ Nếu đồ thị không có chu trình âm thì đường đi ngắn nhất là một đường đi đơn.

+ Nếu đồ thị có chu trình âm thì đường đi ngắn nhất giữa một số cặp đỉnh nào đó có thể không xác định, bởi vì bằng cách đi vòng theo chu trình âm này một số lần đủ lớn ta có thể chỉ ra đường đi giữa hai đỉnh nào đó trong chu trình này nhỏ hơn bất kỳ một số cho trước nào. Trong trường hợp như vậy có thể đặt vấn đề tìm đường đi đơn ngắn nhất. Vấn đề đó là một đường đi đơn NP-đầy đủ. Hiện chưa có ai chứng minh được sự tồn tại hay không một thuật toán đa thức tìm đường đi đơn ngắn nhất trên đồ thị có chu trình âm.

2. Bài toán đường đi ngắn nhất.

Trong các ứng dụng thực tế, ví dụ trong mạng lưới giao thông đường bộ, đường thủy hoặc đường không…Người ta không chỉ quan tâm đến việc tìm đường đi giữa hai điểm bất kỳ mà còn phải lựa chọn một hành trình tiết kiệm nhất (theo không gian, thời gian hay chi phí…). Khi đó phát sinh yêu cầu tìm đường đi ngắn nhất giữa hai đỉnh cyar đồ thị. Bài toán đó có thể được phát biểu như sau:

Cho đồ thị có trọng số G=(V, E, W), hãy tìm một đường đi ngắn nhất xuất phát từ đỉnh s∈V đến một đỉnh đích f∈V. Độ dài đường đi này ký hiện là d[s,f] và gọi là khoảng cách từ ***s*** đến f. Nếu như không tồn tại đường đi từ ***s*** đến f thì ta sẽ đặt d[s,f]=+∞.

Một số dạng bài toán tìm đường đi ngắn nhất:

+ Tìm các đường đi ngắn nhất từ mọi đỉnh tới một đỉnh t cho trước. Bằng cách đảo chiều các cung của đồ thị ta có thể quy về bài toán tìm đường đi ngắn nhất xuất phát từ t.

+ Tìm đường đi ngắn nhất từ đỉnh ***s*** tới đỉnh t cho trước. Nếu ta tìm được đường đi ngắn nhất từ ***s*** đến tất cả các đỉnh khác thì dĩ nhiên bài toán tìm đường đi ngắn nhất từ ***s*** đến t cũng được giải quyết.

+ Tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị.

+ Tìm đường đi ngắn nhất trên đồ thị có chu trình âm. Đây sẽ là vấn đề cần giải quyết trong đề tài này. Mặc dù có nhiều thuật toán tìm đường đi khác nhau nhưng không phải thuật toán nào cũng tìm được được đi ngắn nhất trong đồ thị có chu trình âm.

2.1. Nhãn khoảng cách và phép co.

Trong tất cả các thuật toán về tìm đường đi ngắn nhất trong đồ thị đều sử dụng kỹ thuật gán nhãn khoảng cách: Với mỗi đỉnh v∈V, nhãn khoảng cách d[v] là độ dài một đường đi nào đó từ ***s*** đến v. Trong trường hợp chúng ta chưa xác định được đường đi nào từ ***s*** đến v thì nhãn d[v] được gán bằng giá trị +∞. d[v] có giá trị khởi tạo là:

$$d[v] = \begin{cases} 0, \text{nếu } v = s \\ +\infty, \text{nếu } v \neq s \end{cases} \quad (v = 1,2,\dots,n)$$

Đoạn khởi tạo trên có thể được viết bằng giả mã Pascal như sau:

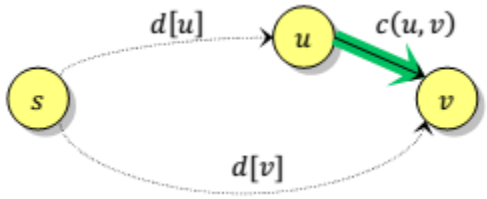
```
Procedure Init;
Begin
  For ∀v∈V do d[v]:=+∞;
  D[s]:=0;
End;
```

Gọi δ(s,v) là độ dài đường đi ngắn nhất từ ***s*** đến v.

Do tính chất của nhãn khoảng cách d[v] là độ dài của một đường đi bất kỳ từ đỉnh đến v nên d[v]>= δ(s,v), ∀v∈V. Các thuật toán tìm đường đi ngắn nhất sẽ cực tiểu hóa dần các nhãn d[.] cho tới khi d[v]>= δ(s,v), ∀v∈V. Việc cực tiểu hóa sẽ được thực hiện bằng phép co.

Phép co theo cạnh (u,v) ∈ E, gọi tắt là phép co (u,v) dd[cj] thực hiện như sau: Giải sử chúng ta đã xác định được d[u] là độ dài một đường đi từ ***s*** đến u, ta nối thêm cạnh (u,v) để được một đường đi từ ***s*** đến v có độ dài d[u]+w(u,v) với w(u,v) là trọng số của cạnh (u,v). Nếu đường đi này có độ dài ngắn hơn d[v] thì ta ghi nhận lại d[v] bằng d[u]+w(u,v).

Minh họa phép co như hình dưới:



Có thể hình dung hoạt động của phép co như sau: Căng một đoạn dây đàn hồi dọc theo đường đi từ ***s*** đến v, đoạn dây sẽ dãn ra tới độ dài d[v]. Tiếp theo ta thử lấy đoạn dây đó căng dọc theo đường đi từ ***s*** tới ***u*** rồi đến v. Nếu đoạn dây bị chùng xuống (co lại) hơn so với đoạn dây cũ ta ghi nhận đường đi tương ứng với cách căng mới, nếu đoạn dây không chùng xuống thì ta vẫn giữ đường đi đó căng theo đường cũ. Chính vì vậy phép co không làm dài thêm d[v], ta nói rằng d[v] vị cực tiểu hóa qua phép co (u,v)

Phép co(u,v) được thực hiện bởi hàm Relax, hàm nhận vào cạnh (u,v) và trả về True nếu nhãn d[v] bị giảm đi qua phép co (u,v):

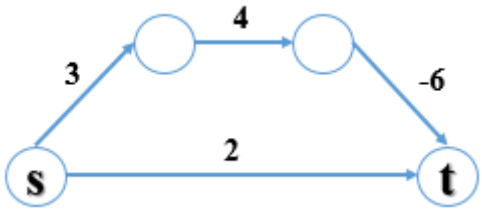
```
Function Relax(e:canh):boolean;
begin
  with e do
  if (d[x]<maxd)and(d[y]>d[x]+w) then
    begin
       d[y]:=d[x]+w;
       trace[y]:=x;
       exit(true);
     end;
     exit(false);
end;
```

Mỗi khi d[v] bị giảm xuống sau phép co(u,v), ta lưu lại vết trace[v]:=u với ý nghĩa đường đi ngắn nhất từ ***s*** đến v tới thời điểm được ghi nhận là đường đi qua ***u*** trước rồi đi tiếp theo cung (u,v). Vết này được sử dụng để truy vết tìm đường đi khi thuật toán kết thúc.

2.2. Tính đúng đắn của thuật toán Dijkstra trong đồ thị có trọng số âm.

Trong trường hợp đồ thị với các cạnh có trọng số không âm, thuật toán Dijkstra cho kết quả chính xác trong thời gian cho phép (Có độ phức tạp thuật toán là $O(n^2)$ với n là số đỉnh của đồ thị). Tuy nhiên trong đồ thị với trọng số của 1 số cung nào đó âm nhưng không tạo thành chu trình âm thì thuật toán Dijkstra thường không chính xác. Sau đây ta xét một ví dụ để chứng minh điều đó.

Ví dụ cho đồ thị như sau:



Lúc đó đỉnh t sẽ được cố định nhãn d[t]=2, nhưng trên thực tế độ dài đường đi ngắn nhất từ s đến t là 1 (3+4-6=1)

Từ đó có thể kết luận rằng trong đồ thị không có chu trình âm nhưng có cung âm, thuật toán Dijkstra cho kết quả chính xác trong một số trường hợp. Do vậy ta cần tìm một thuật toán khác có độ phức tạp tương đương để thay thế.

3.. Thuật toán Bellman-Ford.

3.1. Thuật toán.

Thuật toán Bellman-Ford được phát biểu như sau: Khởi tạo các nhãn khoảng cách d[s]:=0 và d[v]:=+∞, ∀v≠s, sau đó thực hiện phép co theo mọi cạnh của đồ thị. Cứ lặp lại như vậy cho đến khi không thể cực tiểu hóa thêm bất kỳ một nhãn d[v] nào nữa.

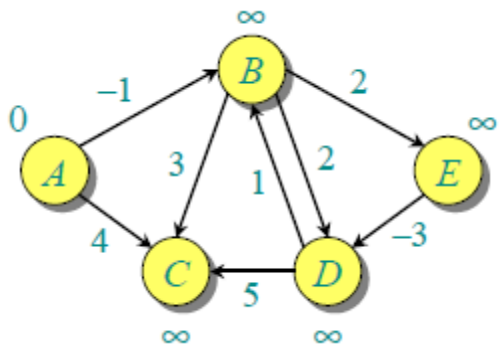
Thuật toán bằng giả mã Pascal:

```
Init; //Khởi tạo
Repeat
 Stop:=true;
 For ∀e∈E do
    If Relax(e) then Stop:=false;
Until false;
```

3.2. Ví dụ minh họa.

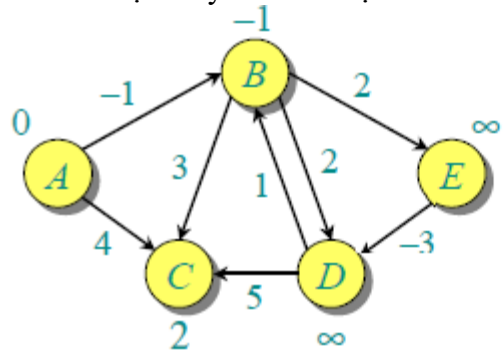
Nguồn: http://www.geeksforgeeks.org

Với đồ thị như hình ở dưới, trong mỗi lượt sẽ xét lần lượt các cung là: (B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D). Các đỉnh ban đầu đều được gán nhãn ∞, còn đỉnh A được gán nhãn 0 vì đây là đỉnh xuất phát.



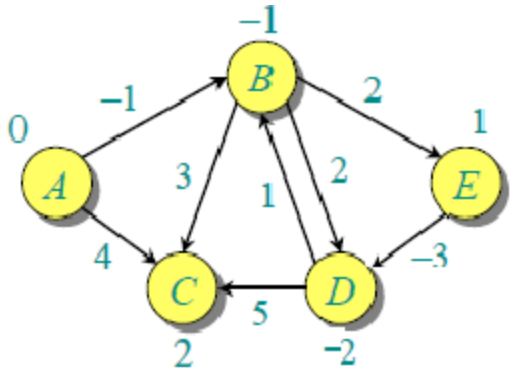
A	B	C	D	E
0	∞	∞	∞	∞

Lượt 1: + Lần lượt xét các cung (B,E), (B,D), (D,B): không thay đổi nhãn
+ Xét cung (A,B) làm thay đổi D[B]=-1
+ Xét cung (A,C) làm thay đổi D[C]=4
+ Xét cung (D,C) không làm thay đổi nhãn
+ Xét cung (B,C) làm thay đổi D[C]=2
+ Xét cung (E,D) không làm thay đổi nhãn
Nhãn các đỉnh được thay đổi sau lượt đầu tiên là:



A	B	C	D	E
0	∞	∞	∞	∞
0	−1	∞	∞	∞
0	−1	4	∞	∞
0	−1	2	∞	∞

Lượt 2: + Xét cung (B,E) làm thay đổi nhãn D[E]=1
+ Xét cung (B,D) làm thay đổi nhãn D[D]=1
+ Xét cung (D,B), (A,B), (A,C), (D,C), (B,C): không thay đổi nhãn
+ Xét cung (E,D) làm thay đổi nhãn D[D]=-1
Nhãn các đỉnh được thay đổi sau lượt thứ 2 là:



A	B	C	D	E
0	∞	∞	∞	∞
0	−1	∞	∞	∞
0	−1	4	∞	∞
0	−1	2	∞	∞
0	−1	2	∞	1
0	−1	2	1	1
0	−1	2	−2	1

Lượt 3: Xét lần lượt tất cả các cung: (B,E), (D,B), (B,D), (A,B), (A,C), (D,C), (B,C), (E,D) đều không làm thay đổi nhãn.
Thuật toán kết thúc.

4. Bài tập

Bài 1: HAI ĐƯỜNG ĐI

Nguồn: http://vn.spoj.com/problems/HIWAY/

Một mạng giao thông gồm N nút giao thông, và có M đường hai chiều nối một số cặp nút, thông tin về một đường gồm ba số nguyên dương u, v là tên hai nút đầu mút của đường, và ***l*** là độ dài đoạn đường đó. Biết rằng hai nút giao thông bất kì có không quá 1 đường hai chiều nhận chúng làm hai đầu mút.

Cho hai nút giao thông ***s*** và t, hãy tìm hai đường đi nối giữa s với t sao cho hai trên hai đường không có cạnh nào được đi qua hai lần và tổng độ dài 2 đường đi là nhỏ nhất.

Dữ liệu vào: Từ tệp văn bản HIWAY.INP

+ Dòng đầu ghi N, M ($N \le 100$)
+ Dòng thứ 2 ghi hai số s, t.
+ M dòng tiếp theo, mỗi dòng mô tả một đường gồm ba số nguyên dương u, v, l.

Dữ liệu ra: ghi vào tệp văn bản HIWAY.OUT

+ Dòng đầu ghi T là tổng độ dài nhỏ nhất tìm được hoặc -1 nếu không tìm được.
+ Nếu tìm được, hai dòng sau, mỗi dòng mô tả một đường đi gồm: số đầu là số nút trên đường đi này, tiếp theo là dãy các nút trên đường đi bắt đầu từ s, kết thúc tại t.

Ví dụ:

HIWAY.INP	HIWAY.OUT
5 7 1 5 1 2 3 1 4 8 2 3 5 2 4 4 3 5 5 4 3 8 4 5 3	24 3 1 4 5 4 1 2 4 5

Hướng dẫn

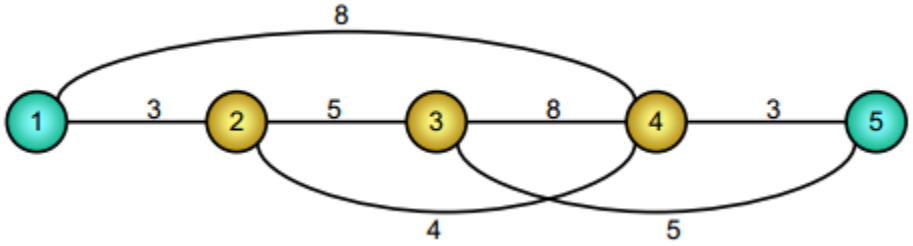
Coi mỗi cạnh của đồ thị tương đương với hai cung có hướng ngược chiều nhau, trọng số trên cung (u,v) được gán bằng c[u,v]

Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ s đến t: <s=v_0, v_1, v_2,…,v_{p-1}, v_p=t>. Dọc trên đường đi Dijkstra, với mỗi cạnh (v_{i-1}, v_i) ta bỏ đi cung (v_{i-1},v_i) giữ lại cung (v_i, v_{i-1}) và gán trọng số cung này là -c[v_i,v_{i-1}]. Sau những phép biến đổi có những cung có trọng số âm nhưng không tạo thành chu trình âm.

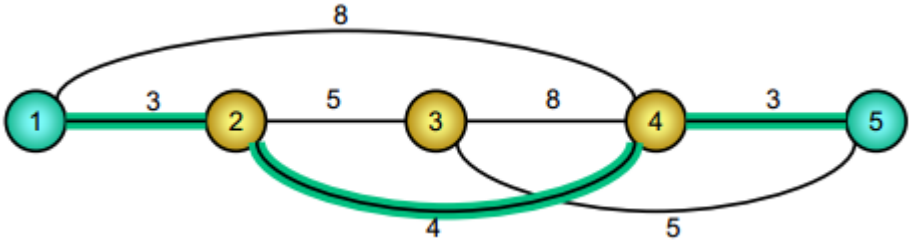
Dùng thuật toán Bellman-Ford tìm đường đi ngắn nhất từ s đến t: <s=u_0, u_1,…,u_{q-1},u_q=t). Dọc trên đường đi Bellman-Ford với mỗi cạnh (u_{i-1},u_i) ta bỏ đi cung (u_{i-1},u_i).

Với mỗi cạnh (u,v) của đồ thị, nếu cả cung (u,v) và (v,u) đều được duy trì đến bước này thì bỏ luôn cả hai, những cung còn lại chỉ đường đi từ t về s, bằng cách lật ngược chiều ta có lời giải.

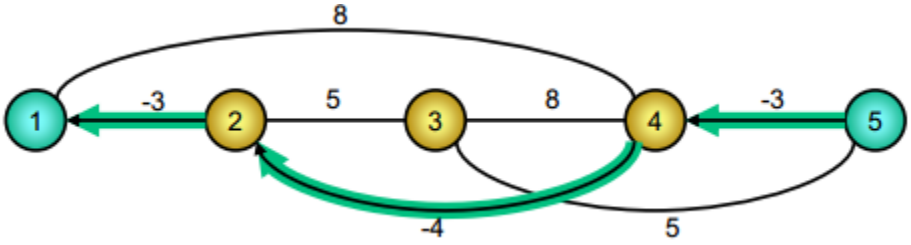
Ví dụ: đồ thị ban đầu với 2 đỉnh s=1 và t=5



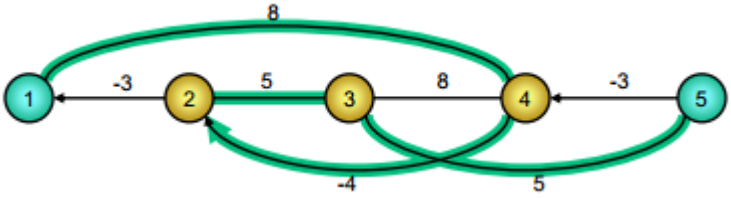
Đường đi Dijkstra: 1, 2, 4, 5 với độ dài 10



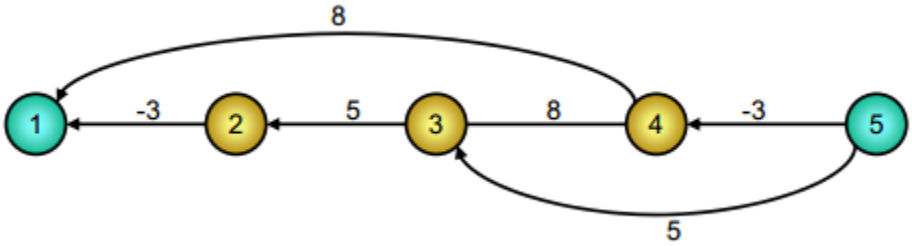
Bỏ đi các cung (1,2); (2,4); (4,5). Đặt lại trọng số các cung ngược chiều đi: c[2,1]=-3; C[4,2]=-4; C[5,4]=-3



Đường đi Bellman-Ford: 1, 4, 2, 3, 5 với độ dài 14



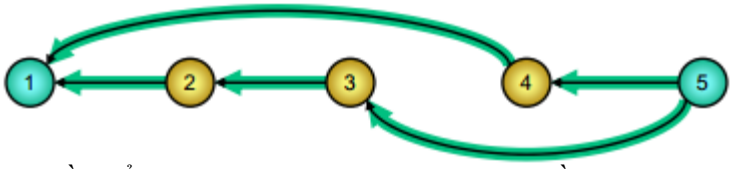
Bỏ các cung: (1,4); (4,2); (2,3); (3,5)



Bỏ nốt cạnh (3,4) ta có 2 đường đi từ 5 về 1:

+ 5, 4, 1

+ 5, 3, 2, 1



Đối chiếu lên đồ thị ban đầu, tổng độ dài 2 đường đi tìm được bằng độ dài đường đi Dijkstra cộng với độ dài đường đi Bellman-Ford=24. Lật ngược thứ tự các đỉnh trong hai đường đi trên ta được cặp đường đi từ 1 đến 5 có tổng độ dìa nhỏ nhất cần tìm.

Bài 2: HỆ RÀNG BUỘC

Cho n biến số nguyên $v_1, v_2,\ldots,v_n$ và một tập m ràng buộc. Mỗi ràng buộc được biểu diễn bởi 3 số nguyên i, j, c có dạng $v_j-v_i<=c$ (c là số nguyên).

Hãy tìm cách gán giá trị nguyên nằm trong phạm vi [a,b] cho các biến $v_1, v_2,\ldots,v_n$ để thỏa mãn tất cả m ràng buộc đã cho.

Dữ liệu vào: Từ tệp văn bản SDC.INP bao gồm:

+ Dòng đầu chứa 4 số nguyên dương n, m,a, b với n<=1000, m<=10000, a<b<=10^6;

+ Mỗi dòng trong m dòng tiếp theo chứa 3 số nguyên i, j, c tương ứng với một ràng buộc (1<=i, j<=n; |c|<=10^6)

Dữ liệu ra: ghi vào tệp văn bản SDC.OUT bao gồm:

+ Dòng 1 ghi từ YES nếu có phương án thực hiện, ghi từ NO nếu không có phương án.

+ Trong trường hợp có phương án thực hiện, dòng 2 ghi n giá trị của $v_1, v_2,\ldots,v_n$ tìm được.

Các số trên dòng của tệp SDC.INP và SDC.OUT được ghi cách nhau ít nhất 1 dấu cách.

Ví dụ:

SDC.INP	SDC.OUT
3 3 1 4 1 2 5 2 3 -2 3 1 -1	YES 1 4 2

Hướng dẫn

Dựng đồ thị G=(V,E) với ***n*** đỉnh ứng với ***n*** biến, mỗi ràng buộc dạng $v_j-v_i\le c_{ij}$ ứng với một cung nối từ đỉnh i đến đỉnh j với trọng số c_{ij}. Bổ sung thêm 1 đỉnh 0 nối tới mọi đỉnh khác bằng cung có trọng số bằng 0.

+ Nếu đồ thị có chu trình âm thì không tồn tại cách gán giá trị cho các biến nằm thỏa mãn các ràng buộc.

+ Nếu đồ thị không có chu trình âm, thuật toán FordBellman có thể áp dụng để xác định d[v] là đường đi ngắn nhất từ 1 đến v. (v=1..n)

Từ bất đẳng thức tam giác: $\forall(i,j)\in E: d[j]\le d[i]+c_{ij}$ Ta có: $d[j]-d[i]\le c_{ij}$

Tức là các giá trị d[.] thỏa mãn tất cả các ràng buộc. Việc cuối cùng là công thêm các d[.] cùng 1 hằng số để các giá trị này rơi vào đoạn [a,b].

Nếu $d[.]_{max}-d[.]_{min}$ >b-a thì không tồn tại cách gán giá trị theo yêu cầu