**Qui hoạch động - Thuật toán Bellman-Ford**

Cho một đồ thị có hướng, mỗi cạnh có trọng số bất kỳ (có thể âm).

* Bài toán 1 tìm các đường đi ngắn nhất một đích đến: Cho trước 1 đỉnh đích đến t, tìm các đường đi ngắn nhất từ mỗi đỉnh đến đỉnh đích t đó
* Bài toán 2 tìm các đường đi ngắn nhất một đỉnh xuất phát: Cho trước 1 đỉnh xuất phát s, tìm các đường đi ngắn nhất từ đỉnh xuất phát s đó đến mỗi đỉnh khác.



Lưu ý: Bài toán tìm đường đi ngắn nhất trên đồ thị có hướng, có trọng số. Nhưng ở đây cho phép trọng số của các cạnh có thể âm, do đó không thể giải được theo thuật toán Dijkstra.



Chu trình âm:









Ở đây chúng ta sử dụng phương pháp Qui hoạch động, tức là giải các bài toán nhỏ trước, chồng chéo nhau, lưu kết quả lại để trợ giúp giải các bài toán lớn hơn dần và cho đến bài toán ban đầu.

Giả sử xét bài toán 1.











* Đối với Bài toán 1 tìm các đường đi ngắn nhất một đích đến t: các bài toán nhỏ là với mỗi đỉnh v, tìm độ dài đường đi ngắn nhất d(i, v) từ v đến t mà chỉ sử dụng tối đa i cạnh (i = 0, 1, …, n – 1, ở đây n là số đỉnh).
* Đối với Bài toán 2 tìm các đường đi ngắn nhất một đỉnh xuất phát s: các bài toán nhỏ là với mỗi đỉnh v, tìm độ dài đường đi ngắn nhất d(i, v) từ s đến v mà chỉ sử dụng tối đa i cạnh (i = 0, 1, …, n – 1, ở đây n là số đỉnh).

Vì d(i,v) ở bước thứ i có thể ghi đè lên d(i-1,v), nên chúng ta dùng mảng một chiều d(v). Và để tìm lại đường đi chúng ta lưu

* Đỉnh w nằm sau v trên đường đi ngắn nhất v đến t đang xét đối với Bài toán 1 gọi là succ(v) = w.
* Đỉnh w nằm trước v trên đường đi ngắn nhất s đến v đang xét đối với Bài toán 2 gọi là prev(v) = w.

Thuật toán Bellman-Ford chạy duyệt tối đa n – 1 bước (vì một chu đường đi n cạnh qua n đỉnh sẽ có một chu trình, có thể loại chu trình đó ra), mỗi bước tương ứng với số cạnh được sử dụng cho các đường đi đang tìm. Ở mỗi bước chúng ta xét đỉnh v, mà d(v) ở bước trước đó được cập nhật:

**Thủ tục Relax** (thư dãn): xét một đỉnh v, mà d(v) được làm tốt lên ở bước trước (nếu không được làm tốt lên ở bước trước, thì cũng sẽ không được xét ở bước sau, vì nó đã được xét ngay sau bước nó được cập nhật). Sau đó xét từng cạnh e (đi vào nó, đối với bài toán 1 và đi ra từ nó đối với bài toán 2) với đỉnh kia là u và xem d(u) có tốt lên không, nếu tốt lên thì cập nhật và ghi nhớ u được cập nhật.

* Đối với Bài toán 1 tìm các đường đi ngắn nhất một đích đến t. Xét các cạnh (u, v) đi vào v:

nếu d(u) > c(u, v) + d(v)

thì cập nhật d(u): = c(u, v) + d(v) và succ(u) = v

* Đối với Bài toán 2 tìm các đường đi ngắn nhất một đỉnh xuất phát s. Xét các cạnh (v, u) đi ra từ v:

nếu d(u) > d(v) + c(v, u)

thì cập nhật d(u): = d(v) + c(v, u) và prev(u) = v

Thực hiện tối đa n -1 bước, nhưng nếu ở một bước nào đó không có giá trị d(u) nào được cập nhật thì dừng. Khi đó

* Đối với Bài toán 1: d(v) là độ dài đường đi ngắn nhất từ v đến t và đường đi đó là: v, succ(v), succ(succ(v)), …., t
* Đối với Bài toán 2: d(v) là độ dài đường đi ngắn nhất từ s đến v và đường đi đó là: s, … , prev(prev(v), prev(v), v

**Phát hiện chu trình âm**: nếu đồ thị có chu trình âm, thì nó sẽ không có đường đi ngắn nhất.

1. **Single destination Shortest paths – Các đường đi ngắn nhất một đích đến**



Đích là đỉnh 6: Tìm các đường đi ngắn nhất từ mỗi đỉnh đến đỉnh 6.

Khởi tạo, giả sử M là một số dương rất lớn, N = Null.

s(i) là đỉnh sau i nằm trên đường đi ngắn nhất hiện thời từ đỉnh i đến đỉnh đích.

**Ví dụ 1:** Tìm các đường đi ngắn nhất từ mỗi đỉnh đến đỉnh 6.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| d(i), s(i) | Bước 0 | 1 | 2 | 3 | 4 |
| d(0);s(0) | M;Null |  | 29;4 | ~~28;1~~, 26;4 | 25;4 |
| d(1);s(1) | M;N |  | 23;2 | (22, 7) |  |
| d(2);s(2) | M;N | 11;6 |  |  |  |
| d(3);s(3) | M;N | 9;6 |  |  |  |
| d(4);s(4) | M;N | 20;6 | 17;5 | 16;5 |  |
| d(5);s(5) | M;N | 13;6 | 12;2 |  |  |
| \*d(6);s(6) | ~~M;N~~, 0;N |  |  |  |  |
| d(7);s(7) | M;N |  | 18;2 |  |  |

Bước 4 không thay đổi gì. Các đường đi ngắn nhất từ các đỉnh đến đỉnh đích.

d(0) = 25: 0 -> 4-> 5 -> 2 -> 6

d(1) = 22: 1 -> 7 -> 2 ->6

d(2) = 11: 2->6

d(3) = 9: 3->6

d(4) = 16: 4 -> 5 -> 2 -> 6

d(5) = 12: 5->2->6

d(6) = 0: 6-> 6

d(7) = 18: 7->2->6

1. **Single Source Shortest paths - Các đường đi ngắn nhất một đỉnh đi**

Shortest – Paths (V, E, c, s)

For each node v in V

d(v) := M

prev(v) := Null

d(s) := 0

For i = 1 to n - 1

For each node w in V

if (d(w) thay đổi ở bước trước)

for each (w,v) in E

if d(v) > d(w) + c (w,v)

d(v) := d(w) + c (w,v)

prev(v) = w

Nếu không có d(v) nào thay đổi ở bước i, dừng

**Ví dụ 2:** Tìm các đường đi ngắn nhất từ đỉnh xuất phát 0 đến mỗi đỉnh khác.

0 là đỉnh xuất phát. Khởi tạo, giả sử M là một số dương rất lớn, N = Null.

p(i) là đỉnh đứng trước đỉnh i, nằm trên đường đi ngắn nhất hiện thời từ đỉnh xuất phát s đến đỉnh i.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| d(i), p(i) | Bước 0 | 1 | 2 | 3 | 4 | **5** |
| \*d(0);p(0) | ~~M;N~~,0;N |  |  |  |  |  |
| d(1);p(1) | M;Null | 5;0 |  |  |  |  |
| d(2);p(2) | M; N |  | ~~17;~~1, ~~15;7,~~ | 14;5 |  |  |
| d(3);p(3) | M; N |  | ~~20;1~~, | ~~18;2~~,17;2 | 17;2 |  |
| d(4);p(4) | M; N | 9;0 |  |  |  |  |
| d(5);p(5) | M; N |  | 13;4 |  |  |  |
| d(6);p(6) | M; N |  | ~~29;4~~, | ~~26;2~~,25;2 |  |  |
| d(7);p(7) | M; N | 8;0 |  |  |  |  |

Chú ý ở bước 3, do d(3) được cập nhật ở bước trước và bước này update(3) = true, nên sau 2 ta vẫn chọn 3, d(6)= 29, d(3) = 18, c(3,6)=9, 29 > 18+9=27, nên cập nhật d(6)=27, s(6)=3.

Ở bước 5, không có d(v) được cập nhật, thuật toán dừng

Các đường di ngắn nhất tìm được là:

d(0) = 0: 0 -> 0

d(1) = 5: 0 -> 1

d(2) = 14: 0->4->5->2

d(3) = 17: 0->4->5->2->3

d(4) = 9: 0 -> 4

d(5) = 13: 0->4->5

d(6) = 25: 0->4->5->2 -> 6

d(7) = 8: 0->7





**Ví dụ 3**: Cho đồ thị sau. Giải

* Bài toán 1 với đỉnh đích là đỉnh z.
* Bài toán 2 với đỉnh xuất phát là đỉnh s.

