TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BỘ MÔN TOÁN RỜI RẠC

_____*_____

BÀI TẬP LỚN

Đề tài số 10:

Bài toán tìm đường đi ngắn nhất trên đồ thị

Giáo viên hướng dẫn: Huỳnh Thanh Bình

Sinh viên thực hiện: Nguyễn Duy Thành - 20102737 - CNTT1-K55

Vũ Văn Hiệp - 20101545 - CNTT2-K55

Bạch Văn Hải - 20101464 - CNTT1-K55

Hà Nội, tháng 4 – 2012

Mục lục

Lời nói đầu……..…….....3

Phần một: Bài toán tìm đường đi ngắn nhất trên đồ thị và các thuật toán thông dụng…….……..……..4

Chương I: Phát biểu bài toán…….……...4

1. Các khái niệm cơ bản……..…….....4

2. Phát biểu bài toán…….…….....5

3. Nhận xét……..…….....5

Chương II: Các thuật toán giải bài toán tìm đường đi ngắn nhất trên đồ thị……6

1. Thuật toán tìm kiếm theo chiều rộng ứng dụng tìm đường đi ngắn nhất trên đồ thị theo số cạnh.....6

2. Thuật toán Ford-Bellman……..……....8

3. Thuật toán Dijkstra……...…….……..11

4. Thuật toán Floyd-Warshall……..…….……..13

5. Thuật toán A*……...…….……16

Phần hai: Giới thiệu về Project minh họa các thuật toán tìm đường đi ngắn nhất trên đồ thị bằng giao diện đồ họa……..……....17

Phần ba: Kết luận…….……..…….....18

Tài liệu tham khảo..…….....19

Lời nói đầu

Trong lý thuyết đồ thị, **Bài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đồ thị liên thông** là một trong những bài toán có ứng dụng thực tế to lớn nhất. Từ bài toán này, người ta đã phát triển thành nhiều bài toán khác như: Bài toán chọn hành trình tiết kiệm nhất về thời gian, chi phí hay khoảng cách trên một mạng lưới giao thông; Bài toán lập lịch thi công các công đoạn của một công trình lớn, Bài toán lực chọn con đường truyền tin với chi phí thấp nhất trong mạng lưới thông tin, Bài toán tìm lời dịch tốt nhất cho một câu nói … Đến nay, chúng ta đã có rất nhiều thuật toán để giải **Bài toán tìm đường đi ngắn nhất** với khả năng áp dụng và độ phức tạp khác nhau. Trong bài báo cáo này, chúng em xin trình bày về 6 phương pháp giải quyết bài toán này kèm theo project minh họa việc giải bài toán bằng giao diện đồ họa.

Dù đã rất cố gắng nhưng do khả năng còn hạn chế nên không thể không có sai sót, rất mong được cô tận tình chỉ bảo. Xin chân thành cảm ơn!

Nhóm sinh viên

Nguyễn Duy Thành

Vũ Văn Hiệp

Bạch Văn Hải

Phần một: Bài toán tìm đường đi ngắn nhất trên đồ thị và các thuật toán thông dụng

Chương I: Phát biểu bài toán

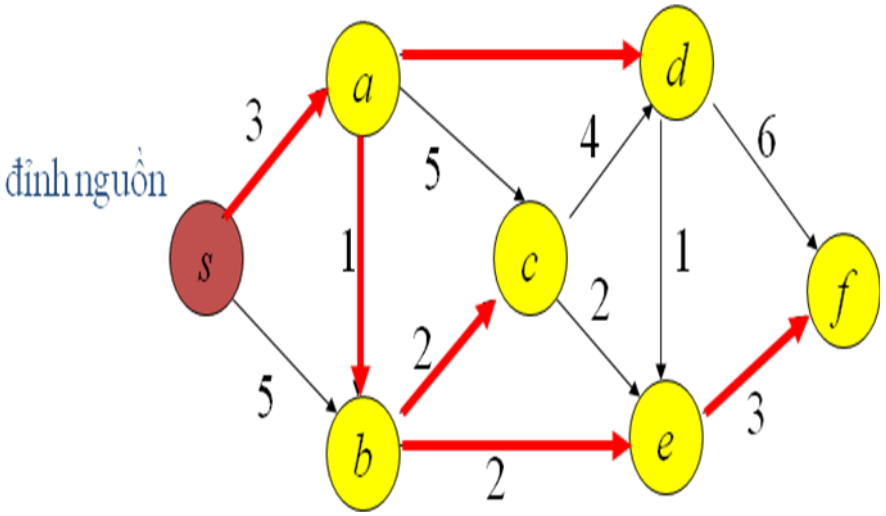
1. Các khái niệm cơ bản

Xét đồ thị có hướng G= (V, E) với |V|=n, |E|=m. Các cung (u, v) của đồ thị được gắn trọng số được xác định bởi hàm w(u, v) có giá trị thực còn được gọi là độ dài của cạnh (u, v). Quy ước w(u, v) = ∞ nếu (u, v) $\notin$ E.

Nếu P = $v_0 \to v_1 \to v_2 \to \ldots \to v_k$ là một đường đi trên G, thì **độ dài** của nó là tổng các trọng số trên các cung của nó:

$$w(P) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

Đường đi ngắn nhất từ đỉnh u đến đỉnh v là đường đi có độ dài ngắn nhất trong số các đường đi từ u đến v, còn được gọi là khoảng cách từ u đến v ký hiệu là δ(u, v).



2. Phát biểu bài toán

Bài toán tìm đường đi ngắn nhất trên đồ thị được phát biểu dưới dạng tổng quát:

"Trên đồ thị liên thông G = (V, E), tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát s ϵ V đến một đỉnh đích t ϵV. Đường đi như vậy gọi là đường đi ngắn nhất từ s đến t, độ dài của nó ký hiệu là d(s, t) ."

3. Nhận xét

- Nếu không tồn tại đường đi từ s đến t thì coi d(s, t) = ∞.
- Nếu mọi chu trình trên đồ thị đều có độ dài dương, thì đường đi ngắn nhất không có đỉnh nào lặp lại (gọi là đường đi cơ bản).
- Nếu trong đồ thị có chu trình âm thì không thể xác định khoảng cách giữa một số cặp đỉnh, nên chúng ta không xét ở đây.
- Đường đi ngắn nhất luôn có thể tìm trong số các đường đi đơn.
- Mọi đường đi ngắn nhất trong đồ thị đều đi qua không quá n-1 cạnh, với n là số đỉnh.
- Ta có thể dùng tính chất sau để chứng minh tính đúng đắn của một thuật toán tìm đường đi ngắn nhất: Mọi đường đi con của đường đi ngắn nhất cũng là đường đi ngắn nhất.
- Giả sử P là đường đi ngắn nhất từ s đến v: P = $s \to \ldots \to u \to v$, khi đó ta có $\delta(s, v) = \delta(s, u) + w(u, v)$.

Chương II: Các thuật toán giải bài toán tìm đường đi ngắn nhất trên đồ thị

1. Thuật toán tìm kiếm theo chiều rộng ứng dụng tìm đường đi ngắn nhất trên đồ thị theo số cạnh

Xét đồ thị có hướng hoặc vô hướng không trọng số G = (V, E) có n đỉnh và m cạnh, khi đó ta chỉ quan tâm xem giữa hai đỉnh bất kỳ của đồ thị có cạnh hay không. Ta quy ước w(u, v) = 1 nếu có cạnh nối u và v, ngược lại w(u, v) = 0. Bài toán trong trường hợp này trở thành tìm đường đi ngắn nhất trên đồ thị theo số cạnh. Khi đó, ta có thể dùng thuật toán tìm kiếm (duyệt) đồ thị theo chiều rộng (BFS) để giải bài toán vì BFS(s) cho phép ta thăm tất cả các đỉnh thuộc cùng một thành phần liên thông với s. Phía dưới là thủ tục được viết dưới dạng mã giả mô tả thuật toán:

```
Procedure BFS(s)
begin
    Queue = ∅;                          //Khởi tạo hàng đợi rỗng
    for v ∈ V do                        //Khởi tạo mảng In_queue
        In_queue[v] = false;
    Enqueue(Queue, v);                  //Kết nạp v vào hàng đợi
    In_queue[v] = true;                 //Đánh dấu v đã ở trong hàng đợi
    while (Queue ≠ ∅) do
```

```
begin
        Dequeue(Queue, p);              //Lấy p ra khỏi hàng đợi
        Visit(p);                       //Thăm đỉnh p
        for  u ∈ Adjacency[p] do        //Tìm trong các đỉnh kề với p
            if  (In_queue[u]==false)  then //Nếu đỉnh chưa được xét
            begin
                Enqueue(Queue, u);
                In_queue[u] = true;
                Before[u] = p;          //Lưu đường đi
            end
    end
end
```

Nhận xét:

- Sau khi thực hiện thủ tục này, nếu In_queue[t] = true thì tồn tại đường đi từ s đến t, ngược lại không có đường đi.
- Biến Before[u] dùng để ghi nhận đỉnh đi trước đỉnh u trong đường đi từ s đến t.
- Độ phức tạp tính toán của thuật toán là O(n+m).

2. Thuật toán Ford-Bellman

Thuật toán Ford-Bellman được sử dụng trong việc tìm đường đi ngắn nhất từ một đỉnh s đến tất cả các đỉnh còn lại. Nó có một ưu điểm rất hay đó là có thể làm việc với đồ thị có trọng số các cung tùy ý.

Thuật toán được xây dựng dựa trên tư tưởng sau: Xét đồ thị có hướng G = (V, E) có n đỉnh với s là đỉnh xuất phát và a[u, v], u, v $\in$ V là ma trận trọng số. Giả thiết đồ thị không có chu trình âm, ta gọi d[v] là khoảng cách từ s đến v. Ta khởi tạo d[s] = 0 và d[v] = ∞ với v $\neq$ s. Xét mọi cạnh (u, v) của đồ thị, nếu có cạnh nào thỏa mãn bất đẳng thức d[u] + a[u, v] < d[v] thì ta gán d[v] = d[u] + a[u, v] để tối ưu hóa d[v]. Tức là nếu độ dài đường đi từ s tới v lại lớn hơn tổng độ dài đường đi từ s đến u cộng với chi phí đi từ u đến v thì ta hủy bỏ đường đi từ s đến v đang có và thay vào đó là con đường từ s đến u và từ u đến v. Thuật toán sẽ kết thúc khi không thể tối ưu thêm bất kỳ d[v] nào nữa. Dưới đây là mã giả mô tả thuật toán:

```
procedure Ford_Bellman
begin
    for v ∈ V do
    begin
        d[v] = a[s, v];
        Before[v] = s;
    end
    d[s] = 0;
```

```
for  k = 1 to n-1 do
    for v ∈ V\{s} do
        for u ∈ V do
            if  (d[v] > d[u] + a[u, v])
            begin
                d[v] > d[u] + a[u, v];
                Before[v] = u;
            end
end
```

Chứng minh tính đúng đắn của thuật toán:

- Sau khi khởi tạo d[s] = 0 và d[v ≠ s] = ∞, dãy d[v] chính là độ dài ngắn nhất của đường đi từ s đến v không quá 0 cạnh.
- Tại lần lặp thứ k, d[v] bằng độ dài đường đi ngắn nhất từ s đến v không quá k cạnh. Dễ thấy tính chất sau: đường đi từ s đến v qua không quá k + 1 cạnh sẽ phải được thành lập bằng cách lấy đường đi từ s đến một đỉnh u nào đó qua không quá k cạnh, rồi đi qua cung (u, v) để tới v. Vì vậy, độ dài đường đi ngắn nhất từ s đến v qua không quá k+1 cạnh sẽ là nhỏ nhất (Nguyên lý tối ưu Bellman).
- Sau lần lặp thứ n-1, ta có d[v] bằng độ dài đường đi ngắn nhất từ s tới v qua không quá n-1 cạnh. Vì đồ thị không có chu trình âm nên sẽ có một đường đi ngắn nhất từ s đến v là đường đi cơ bản, tức là d[v] là độ dài đường đi ngắn nhất từ s đến v.

Nhận xét:

- Độ phức tạp của thuật toán là $O(n^3)$.
- Ta có thể tăng hiệu quả thuật toán bằng cách chấm dứt vòng lặp theo k khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến d[v] nào bị thay đổi giá trị.
- Đối với đồ thị thưa, ta nên sử dụng ma trận kề Adjacency[v] trong vòng lặp trong cùng theo u vì sẽ giúp độ phức tạp thuật toán chỉ còn là O(n.m):

```
for u ∈ Adjacency[v] do
    if  (d[v] > d[u] + a[u, v])
    begin
        d[v] > d[u] + a[u, v];
        Before[v] = u;
    end
```

- Thuật toán More Bellman: đây là một cải tiến của Ford-Bellman. Ý tưởng của nó là thay vì tìm kiếm cặp đỉnh (u, v) thỏa mãn bất đẳng thức để tối ưu đường đi từ s tới v, ta sẽ lưu các đỉnh u vào hàng đợi Queue, rồi lần lượt lấy các đỉnh trong Queue ra để tính d[v]. Từ đỉnh u ta tính d[v] cho các đỉnh v là lân cận của u, nếu nhận được giá trị tốt hơn hiện tại thì cập nhật lại d[v] và kiểm tra xem v có nằm trong Queue chưa. Nếu v không nằm trong Queue thì nạp nó vào còn không thì bỏ qua. Việc cài đặt thuật toán này khá phức tạp nên không tiện trình bày tại đây.

3. Thuật toán Dijkstra

Trong trường hợp trọng số trên các cung là không âm, thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị làm việc hiệu quả hơn so với thuật toán Ford-Bellman.

Ý tưởng của thuật toán là gán cho các đỉnh các nhãn tạm thời cho biết cân trên của đọ dài đường đi ngắn nhất từ s đến đỉnh đó. Các nhãn được biến đổi theo một thủ tục lặp. Mỗi bước lặp sẽ có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh nào đó trở thành nhãn cố định thì nó sẽ cho ta không phải là cận trên mà là độ dài đường đi ngắn nhất từ đỉnh s đến đỉnh nó.

Sau đây là mã giả mô tả thuật toán:

```
procedure Dijkstra
begin
    for   v ∈ V  do            //Khởi tạo
   begin
        d[v] = a[s,v];
        Before[v] = s;
    end;
    d[s] = 0;
    S = {s};                   //S là tập các đỉnh có nhãn cố định
    T = V \ {s};               //T là tập các đỉnh có nhãn tạm thời
    while  T ≠∅ do
```

```
begin
    Tìm các đỉnh u ∈ T thỏa mãn d[u] = min{d[z]: z ∈ T};
    T = T \ {u};
    S:= S ∪{u};        //Cố định nhãn đỉnh u
    for v ∈ T do       //Gán nhãn lại cho các đỉnh trong T
        if (d[v] > d[u] + a[u, v]) then
        begin
            d[v] = d[u] + a[u, v];
            Before[v] = u ;
        end
end
end
```

Chứng minh tính đúng đắn của thuật toán:

- Giả sử ở một bước lặp nào đó các nhãn cố định cho ta độ dài các đường đi ngắn nhất từ s đến các đỉnh có nhãn cố định, ta sẽ chứng minh rằng ở lần lặp tiếp theo nếu đỉnh u* thu được nhãn cố định thì d[u*] chính là độ dài đường đi ngắn nhất từ s đến u*.
- Nếu gọi S_1 là tập các đỉnh có nhãn cố định, S_2 là tập các nhãn tạm thời ở bước lặp đang xét. Kết thúc mỗi bước lập nhãn tạm thời d[v] cho ta độ dài đường đi ngắn nhất từ s đến v chỉ qua những đỉnh nằm hoàn toàn trong tập S_1. Giả sử đường đi ngắn nhất từ s đến u* không nằm trọn

trong tập S_1, tức là nó đi qua ít nhất một dỉnh của tập S_2. Gọi $z \in S_2$ là đỉnh đầu tiên như vậy trên đường đi này. Do trong số trên các cung là không âm nên đoạn đường từ z đến u* có độ dài L>0 và d[z] < d[u*] – L < d[u*]. Bất đẳng thức này mâu thuẫn với cách xác định đỉnh u* là đỉnh có nhãn tạm thời nhỏ nhất. Vậy đường đi ngắn nhất từ s đến u* phải nằm trọn trong S_1 và d[u*] là độ dài đường đi.

- Tại lần lặp đầu tiên S_1 = {s}, tại các lần lặp tiếp theo ta lần lượt thêm vào S_1 một đỉnh u*. Do đó giả thiết d[v] là đường đi ngắn nhất từ s đến v với mọi $v \in S_1$ là đúng với lần lặp đầu tiên. Áp dụng quy nạp suy ra thuật toán Dijkstra cho ta đường đi ngắn nhất từ s đến mọi đỉnh của đồ thị.

Nhận xét:

- Đơn giản, dễ cài đặt.
- Độ phức tạp thuật toán là $O(n^2)$.
- Có thể cải tiến Dijkstra bằng cách dùng cấu trúc đống Heap, khi đó ta có thuật toán Dijkstra Heap với độ phức tạp O(mlogn). Do cài đặt phức tạp nên không trình bày trong báo cáo.

4. Thuật toán Floyd-Warshall

Ta có thể giải bài toán tìm đường đi ngắn nhất giữa tất cả các đỉnh của đồ thị bằng cách sử dụng n lần thuật toán Ford-Bellman (độ phức tạp trở thành $O(n^4)$) hoặc thuật toán Dijkstra (độ phức tạp trở thành $O(n^3)$). Tuy nhiên, với trường hợp tổng quát, ta phải sử dụng thuật toán Floyd-Warshall.

Tư tưởng của thuật toán này là với mọi đỉnh k của đồ thị được xét theo thứ tự từ 1 đến n, xét mọi cặp đỉnh (u, v), ta làm cực tiểu hóa d[u, v] theo công thức: d[u, v] = min(d[u, v], d[u,k] + d[k, v]). Tức là nếu đường đi từ u đến v đang có lại dài hơn đường đi từ u đến k cộng cới đường đi từ k đến v thì ta hủy bỏ đường đi từ u đến v hiện tại và thay vào đó là hai đường từ u đến k và từ k đến v.

Sau đây là mã giả mô tả thuật toán:

```
procedure Floyd_Warshall
begin
    for  i = 1 to n do
        for j = 1 to n do
        begin
            d[i, j] = a[i, j];
            Before[i, j] = I;
        End
    for  k = 1 to n do
        for  i = 1 to n do
            for  j = 1 to n do
                if  (d[i, j] > d[i, k] + d[k, j]) then
                begin
                    d[i, j] = d[i, k] + d[k, j];
```

Before[i, j] = Before[k, j];

end

end

Chứng minh tính đúng đắn của thuật toán:

- Gọi $a^k[u, v]$ là độ dài đường đi ngắn nhất từ u đến v mà chỉ đi qua các đỉnh trung gian thuộc tập {1, 2, …, k}, khi đó với k = 0 thì $a^0[u, v] = a[u, v]$.
- Giả sử ta đã tính được các $a^{k-1}[u, v]$ thì $a^k[u, v]$ sẽ được xây dựng như sau:
  - Nếu không đi qua đỉnh k tức là chỉ đi qua các đỉnh trung gian thuộc tập {1, 2, …, k-1}, khi đó $a^k[u, v] = a^{k-1}[u, v]$.
  - Nếu có đi qua đỉnh k thì đường đi đó sẽ bao gồm hai đường đi từ u đến k và từ k đến v, hai đường này chỉ đi qua các đỉnh trung gian thuộc tập {1, 2, …, k-1}, khi đó ta có $a^k[u, v] = a^{k-1}[u, k] + a^{k-1}[k, v]$.
  - Vì ta muốn $a^k[u, v]$ là cực tiểu nên suy ra $a^k[u, v] = \min(a^{k-1}[u, v], a^{k-1}[u, k] + a^{k-1}[k, v])$.
- Cuối cùng, ta có $a^n[u, v]$ là đường đi ngắn nhất từ u đến v mà chỉ đi qua các đỉnh trung gian thuộc tập {1, 2, …, n}.

Nhận xét:

- Khi cài đặt thuật toán, ta không có khái niệm $a^k[u, v]$ mà sẽ thao tác trực tiếp với a[u, v]. a[u, v] tại bước tối ưu thứ k sẽ được tính toán để bằng $\min(a[u, v]), a^{k-1}[u, k] + a^{k-1}[k, v])$.
- Độ phức tạp thuật toán là $O(n^3)$.

5. Thuật toán a*

Phần hai: Giới thiệu về Project minh họa các thuật toán tìm đường đi ngắn nhất trên đồ thị bằng giao diện đồ họa

Phần ba: Kết luận

Tài liệu tham khảo

Boss14420@gmail.com