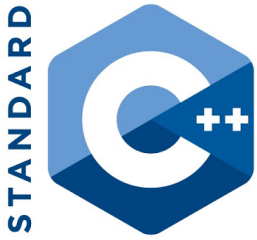
THUẬT TOÁN

Thuật toán Dijkstra tìm đường đi ngắn nhất

23/11/2015 18,276 Views
8 Min Read



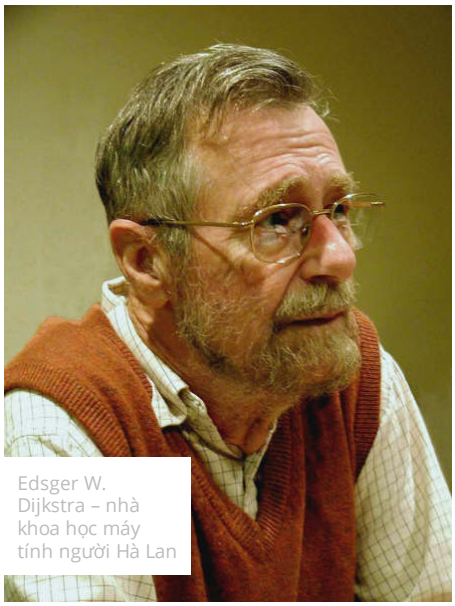
Xin chào mọi người!

Vậy là đã sắp cuối tháng 11, sắp đến 1 mùa thi cho các bạn sinh viên rồi. Mình thi cuối tháng 12 nên nhẹ lo hơn (bù lại không được đi chơi Noel :((buồn lắm chứ, vả lại cũng có gấu đâu mà đi =))). Không những thế còn thi kéo dài qua tết dương lịch nữa chứ, thật là dã man con ngan mà...

Hôm nay mình có sử dụng Google Map, thấy nó tìm đường thật là bá đạo, nhưng liệu Google làm sao biết được đường đi ngắn nhất từ điểm A đến điểm B trong số vô vàn các con đường kia? Có rất nhiều cách giải quyết cho bài toán này, nhưng hôm nay mình sẽ giới thiệu về thuật toán Dijkstra tìm đường đi ngắn nhất (Dijkstra's Shortest Path Algorithm).

1. Khái niệm

Giải thuật Dijkstra, mang tên của 1 nhà khoa học máy tính người Hà Lan **Edsger W. Dijkstra**, là một thuật toán giải quyết bài toán đường đi ngắn nhất trong một đồ thị có hướng **không có cạnh trọng số âm**. Ứng dụng lớn nhất của thuật toán này là trong công nghệ Hệ thống định vị toàn cầu (GPS).



Edsger W. Dijkstra – nhà khoa học máy tính người Hà Lan

Cho 1 đồ thị có hướng **G = (V, E)** với các cạnh có trọng số không âm, có dữ liệu nhập vào là ma trận trọng số **L** và 2 đỉnh **x, y** cho trước. Việc ta cần làm là

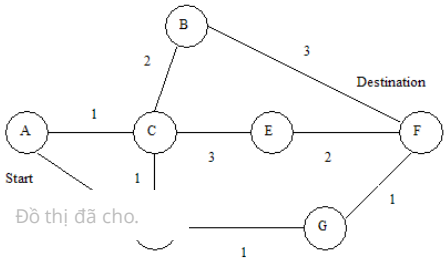
tìm đường đi ngắn nhất từ **x** đến **y** trong đồ thị **G**.

Việc chúng ta cần làm là chỉ ra đỉnh **v** bất kì sao cho **x -> v** là đường đi ngắn nhất. Ta gọi **length[v]** là giá trị đường đi ngắn nhất từ **x -> v**, có thể hiểu **length[v]** là giá trị đường đi ngắn nhất trong các đường đi từ đỉnh **x** qua các đỉnh trong tập hợp S (nếu có) rồi đến **v**.

Thuật toán:

1. Khởi tạo các mảng **n** phần tử: **label, length, prev**.
   Gán **label[k] = 1, length[k] = -1 (inf), prev[k] = -1** với **k** chạy từ **0 -> n – 1**.
   Gán **length[first] = 0**
2. Chọn đỉnh **v** trong mảng sao cho **length[k]** là nhỏ nhất. Sau đó gán **label[k] = 0** (Đã đánh dấu)
3. Tạo vòng lặp với biến chạy **k**, xét nếu **label[k] = 1** (Chưa đánh dấu) và có đường đi từ **v -> k**:
   Nếu **length[k] > length[v] +** trọng số từ **v -> k** hoặc **length[k] = inf**, có nghĩa là nếu ta tìm được 1 đường từ **v -> k** là nhỏ nhất, hoặc là chưa tìm được đường nào ngắn nhất (inf) =>
   Gán **length[k] = length[v] +** trọng số **v -> k**, **prev[k] = v** (Tạo vết chân đỉnh trước đó).
4. Nếu **label[last] = 0** (Đã đánh dấu đỉnh đến), kết thúc vòng lặp. Nếu không thì quay lại bước 2.

VD: Ta có 1 đồ thị như sau



Đồ thị đã cho.

Ta cần chỉ ra đường đi ngắn nhất từ đỉnh **A** tới **F**. Vậy các bước sẽ như thế nào?

A	B	C	D	E	F	G	Ghi chú
					inf	inf	Khởi tạo
					inf	inf	Tìm khoảng cách từ A đến các đỉnh kề. Đỉnh C là đỉnh có đường đi ngắn nhất => Đánh dấu C
					inf	inf	Tìm khoảng cách từ C đến các đỉnh kề. Đỉnh D là đỉnh có đường đi ngắn nhất từ A => Đánh dấu D
					inf	(3, D)	Tìm khoảng cách từ D đến các đỉnh kề. Đỉnh B là đỉnh có đường đi ngắn nhất từ A => Đánh dấu B
					6, B)	**(3, D)***	Tìm khoảng cách từ B đến các đỉnh kề. Đỉnh G là đỉnh có đường đi ngắn nhất từ A => Đánh dấu G
					,4, G)		Tìm khoảng cách từ G đến các đỉnh kề. Đỉnh E là đỉnh có đường đi ngắn nhất từ A => Đánh dấu E
					(4, G)*		Tìm khoảng cách từ E đến các đỉnh kề. Đỉnh F là đỉnh có đường đi ngắn nhất từ A => Đánh dấu F (Kết thúc)

Bảng thống kê các bước thực hiện.

Về nguyên lý, thuật toán Dijkstra giống như việc chạy thi đua vậy. Từ điểm đến ban đầu, ta sẽ mỗi người chạy đến điểm kết thúc theo các đường đi khác nhau. Nếu người nào chạy tới đích trước (tìm min và đánh dấu điểm kết thúc sớm nhất) thì ta xuất ra các đỉnh mà người đó đã đi qua.

2. Cài đặt thuật toán

Input trong bài toán có dạng:

Mã nguồn:

```
7 // n
1 6 // first và last
0 0 1 4 0 0 0
0 0 1 0 0 4 0
1 1 0 2 3 0 0
4 0 2 0 0 0 1
0 0 3 0 0 2 0
0 4 0 0 2 0 1
0 0 0 1 0 1 0
```

Đầu tiên ta cần phải tạo 1 Class:

Mã nguồn:

```
class Dijkstra
{
private:
    int n;
    int **mat; // Graph
    int firstVer, lastVer;
    int* label;
    int* length;
    int* prev;
    bool createPath();
public:
    Dijkstra() {}
    Dijkstra(string filePat
    void findMinPath(string
    ~Dijkstra();
};
```

Xây dựng Constructor cho class

Mã nguồn:

```
Dijkstra::Dijkstra(string f
{
    ifstream fi(filePath);

    // Đọc số đỉnh trong fi
    fi >> n;

    // Đọc đỉnh đầu và cuối
    fi >> firstVer >> lastV

    // Ta cần giảm đi 1 cho
    // Vì mảng bắt đầu từ 0
    firstVer--;
    lastVer--;

    // Cấp phát động
    label = new int[n];
    length = new int[n];
    prev = new int[n];
    mat = new int*[n];
    for (int i = 0; i < n;

    // Khởi tạo
    for (int i = 0; i < n;
    {
        label[i] = 1;
        length[i] = -1; //
        prev[i] = -1;
    }
    length[firstVer] = 0;

    // Đọc Graph matrix
    for (int i = 0; i < n;
    {
        for (int j = 0; j <
        {
            fi >> mat[i][j]
        }
    }

    fi.close();
}
```

Kế tiếp, ta tạo 1 method createPath() để tìm đường đi ngắn nhất

Mã nguồn:

```
bool Dijkstra::createPath()
{
    // Chừng nào đỉnh lastV
    while (label[lastVer] =
    {
        int min = -1;
        int vertex = -1;

        // Tìm min length
        for (int i = 0; i <
        {
            if (label[i] ==
            {
                min = lengt
                vertex = i;
            }
        }

        // Nếu ta không tìm
        if (min == -1)
        {
```

```
                return false;
            }

            // Đánh dấu đỉnh ve
            length[vertex] = mi
            label[vertex] = 0;


            for (int i = 0; i <
            {
                // Nếu đỉnh chu
                if (label[i] ==
                {
                    // Nếu đườn
                    if (length[
                    {
                        length[
                        // Tạo
                        prev[i]
                    }
                }
            }
        }
        return true;
    }
```

Cuối cùng ta tạo 1 method để xuất ra đường đi ngắn nhất

Mã nguồn:

```
// Ta cũng có thể dùng stac
void Dijkstra::findMinPath(
{
    ofstream fo(filePath);
    bool pathExists = this-
    if (!pathExists) fo <<
    else
    {
        // Dò ngược từ đỉnh
        int k = lastVer;
        while (k != firstVe
        {
            fo << (k + 1) <
            // Tìm ngược lạ
            k = prev[k];
        }
        fo << (firstVer + 1
    }
}
```

Vậy là ta đã giải quyết thành công bài toán tìm đường đi ngắn nhất bằng thuật toán Dijkstra. Thật đơn giản phải không nào?

Trên đây là bài viết về Tìm đường ngắn nhất bằng thuật toán Dijkstra. Cảm ơn các bạn đã chú ý theo dõi! Xin chào và hẹn gặp lại ở những bài tiếp theo

Tags thuat_toan

You may also like

About the author VIEW ALL POSTS

Võ Hoài Sơn

Tính tình bất định
Chọc vào là bịnh
Rất yêu lập trình
Luôn code hết mình
Mình hiện đang là sinh viên của trường ĐH Khoa học tự nhiên TPHCM. Bản thân rất thích code, kiêm luôn cả mần thơ nên thường hơi hâm hâm dở dở. Ngoài ra chém gió, chém chuối, chém trái cây các kiểu cũng là sở trường của mình. Rất mong được làm quen với các bạn :D

Add Comment

2 bình luận Sắp xếp theo Mới nhất

Thêm bình luận...

Phuc Trinh ·
CEO & Founder tại Trường Sĩ Quan Thông Tin
Good. Hoc xong quen sach @@
Thích · Phản hồi · 28 Tháng 12 2017 21:47

Duy Đạt ·
Student Officer tại FPT University HCM
Cảm ơn bạn, bài rất hay và dễ hiểu

Xem nhiều nhất

Bài viết

LẬP TRÌNH C#
LinQ là gì?
16/12/2016 859 Views

THỦ THUẬT
Thủ thuật debug trong Visual Studio (Phần 2)
28/11/2016 577 Views

THỦ THUẬT
Thủ thuật debug trong Visual Studio (Phần 1)
28/11/2016 1,977 Views

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG