



Simple Code C Java (<http://simplecodecjava.blogspot.com/>)

Lập trình thật đơn giản



(https://pub.accesstrade.vn/deep_link/4348611980423676010?url=http%3A%2F%2Fmytour.vn%2F)

[Bài mới \(http://simplecodecjava.blogspot.com/\)](http://simplecodecjava.blogspot.com/)

[Thuật toán \(http://simplecodecjava.blogspot.com/2015/09/thuat-toan.html\)](http://simplecodecjava.blogspot.com/2015/09/thuat-toan.html)

[Tải xuống](#) ▼

[Bài tập mẫu \(http://simplecodecjava.blogspot.com/2015/09/bai-tap-mau.html\)](http://simplecodecjava.blogspot.com/2015/09/bai-tap-mau.html)

(<https://www.facebook.com/simplecodecjava>)

(<https://plus.google.com/115027304727901283630>)

(<https://twitter.com/SimpleCodeCJava>)



[Thuật toán] Tìm đường đi ngắn nhất Dijkstra

3:44 PM (2015-10-25T15:44:00+07:00) Posted by Admin Blog (<https://plus.google.com/115027304727901283630>) No (<https://www.blogger.com/comment.g?blogID=9046094762067454426&postID=2991691592852221416>) Comments

Xét đồ thị $G = \langle V, E \rangle$: trong đó $|V| = n$, $|E| = m$. Với mỗi cạnh $(u, v) \in E$, ta đặt tương ứng với nó một số thực $A_{\langle u, v \rangle}$ được gọi là trọng số của cạnh. Ta sẽ đặt $A_{\langle u, v \rangle} = \infty$ nếu $(u, v) \notin E$. Nếu dãy v_0, v_1, \dots, v_k là một đường đi trên G thì tổng của tất cả các cạnh (v_{i-1}, v_i) được gọi là độ dài của đường đi.


Bài toán tìm đường đi ngắn nhất trên đồ thị dưới dạng tổng quát có thể được phát biểu dưới dạng sau: Tìm đường đi ngắn nhất từ một đỉnh xuất phát $s \in V$ (đỉnh nguồn) đến đỉnh cuối $t \in V$ (đỉnh đích). Đường đi như vậy được gọi là đường đi ngắn nhất từ s đến t , độ dài của đường đi $d(s, t)$ được gọi là khoảng cách ngắn nhất từ s đến t (trong trường hợp tổng quát $d(s, t)$ có thể âm). Nếu như không tồn tại đường đi từ s đến t thì độ dài đường đi $d(s, t) = \infty$. Nếu như mỗi chu trình trong đồ thị đều có độ dài dương thì trong đường đi ngắn nhất sẽ không có đỉnh nào bị lặp lại, đường đi như vậy được gọi là đường đi cơ bản. Nếu như đồ thị tồn tại một chu trình nào đó có độ dài âm, thì đường đi ngắn nhất có thể không xác định, vì ta có thể đi qua chu trình âm đó một số lần đủ lớn để độ dài của nó nhỏ hơn bất kỳ một số thực cho trước nào.

1. Thuật toán gán nhãn.

Có rất nhiều thuật toán khác nhau được xây dựng để tìm đường đi ngắn nhất. Nhưng tư tưởng chung của các thuật toán đó có thể được mô tả như sau:

Từ ma trận trọng số $A_{\langle u, v \rangle}$, $u, v \in V$, ta tìm cận trên $d[v]$ của khoảng cách từ s đến tất cả các đỉnh $v \in V$. Mỗi khi phát hiện thấy $d[u] + A_{\langle u, v \rangle} < d[v]$ thì cận trên $d[v]$ sẽ được làm tốt lên bằng cách gán $d[v] = d[u] + A_{\langle u, v \rangle}$. Quá trình sẽ kết thúc khi nào ta không thể làm tốt hơn lên được bất kỳ cận trên nào, khi đó $d[v]$ sẽ cho ta giá trị ngắn nhất từ đỉnh s đến đỉnh v . Giá trị $d[v]$ được gọi là nhãn của đỉnh v . Ví dụ dưới đây thể hiện tư tưởng trên bằng một thuật toán gán nhãn tổng quát như sau:

Ví dụ. Tìm đường đi ngắn nhất từ đỉnh **A** đến đỉnh **I** trên đồ thị hình sau:

 (https://dl.dropboxusercontent.com/u/79544280/SimpleCodeJava/Blogger/Image/150427_dijkstra.png)

Thuật toán tìm đường đi ngắn nhất Dijkstra

- Bước 1.** Gán cho nhãn đỉnh A là 0;
- Bước 2.** Trong số các cạnh (cung) xuất phát từ A, ta chọn cạnh có độ dài nhỏ nhất, sau đó gán nhãn cho đỉnh đó bằng nhãn của đỉnh A cộng với độ dài cạnh tương ứng. Ta chọn được đỉnh C có trọng số AC = 5, nhãn $d[C] = 0 + 5 = 5$.
- Bước 3.** Tiếp đó, trong số các cạnh (cung) đi từ một đỉnh có nhãn là A hoặc C tới một đỉnh chưa được gán nhãn, ta chọn cạnh sao cho nhãn của đỉnh cộng với trọng số cạnh tương ứng là nhỏ nhất gán cho nhãn của đỉnh cuối của cạnh. Như vậy, ta lần lượt gán được các nhãn như sau: $d[B] = 6$ vì $d[B] < d[C] + |CB| = 5 + 4$; $d[E] = 8$; Tiếp tục làm như vậy cho tới khi đỉnh I được gán nhãn đó chính là độ dài đường đi ngắn nhất từ A đến I. Thực chất, nhãn của mỗi đỉnh chính là đường đi ngắn nhất từ đỉnh nguồn tới nó.

Quá trình có thể được mô tả như trong bảng dưới đây.



Bước	Đỉnh được gán nhãn	Nhãn các đỉnh	Đỉnh đã dùng để gán nhãn
Khởi tạo	A	0	A
1	C	0+5 = 5	A
2	B	0+6 = 6	A
3	E	0+8 = 8	A
4	D	5+4 = 9	C
5	F	5+7 = 13	B
6	H	8+6 = 14	E
7	G	9+6 = 15	D
8	I	15 + 3 = 18	G

Như vậy, độ dài đường đi ngắn nhất từ A đến I là 18. Đường đi ngắn nhất từ A đến I qua các đỉnh: A-> C-> D -> G -> I.

2. Thuật toán Dijkstra.

Thuật toán tìm đường đi ngắn nhất từ đỉnh **s** đến các đỉnh còn lại được Dijkstra đề nghị áp dụng cho trường hợp đồ thị có hướng với trọng số không âm. Thuật toán được thực hiện trên cơ sở gán tạm thời cho các đỉnh. Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó. Các nhãn này sẽ được biến đổi (tính lại) nhờ một thủ tục lặp, mà ở mỗi bước lặp một số đỉnh sẽ có nhãn không thay đổi, nhãn đó chính là độ dài đường đi ngắn nhất từ s đến đỉnh đó.

Bạn cũng có thể xem thêm thuật toán Floyd - Tìm đường đi ngắn nhất ở [đây](http://simplecodejava.blogspot.kr/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html)

(<http://simplecodejava.blogspot.kr/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html>).

Thuật toán có thể được mô tả bằng thủ tục Dijkstra như sau:

```

1 void Dijkstra(void)
2 /*Đầu vào G=(V, E) với n đỉnh có ma trận trọng sốA[u,v]≥0; s∈V */
3 /*Đầu ra là khoảng cách nhỏ nhất từ s đến các đỉnh còn lại d[v]: v∈V*/
4 /*Truoc[v] ghi lại đỉnh trước v trong đường đi ngắn nhất từ s đến v*/
5 {
6     /* Bước 1: Khởi tạo nhãn tạm thời cho các đỉnh*/
7     for (v∈V) {
8         d[v] = A[s, v];
9         truoc[v] = s;
10    }
11    d[s] = 0; T = V\{s}; /*T là tập đỉnh có nhãn tạm thời*/
12    /* Bước lặp */
13    while (T != ∅) {
14        Tìm đỉnh u∈T sao cho d[u] = min{ d[z] | z∈T };
15        T = T\{u}; /*cố định nhãn đỉnh u*/;
16        For(v∈T) { /* Gán lại nhãn cho các đỉnh trong T*/
17            If(d[v] > d[u] + A[u, v]) {
18                d[v] = d[u] + A[u, v];
19                truoc[v] = u;
20            }
21        }
22    }
23 }
```

Chương trình cài đặt thuật toán Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh khác của đồ thị có hướng với trọng số không âm được thực hiện như sau:

```

1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 #define MAX 50
```

```

5 #define TRUE 1
6 #define FALSE 0
7 #define VOCUNG 10000000
8 int n;//số đỉnh của đồ thị.
9 int s;//đỉnh đầu.
10 int t;//đỉnh cuối
11 char chon;
12 int truoc[MAX];//mảng đánh dấu đường đi.
13 int d[MAX];//mảng đánh dấu khoảng cách.
14 int Matrix[MAX][MAX];//ma trận trọng số
15 int chuaxet[MAX];//mảng đánh dấu đỉnh đã được gán nhãn.
16
17 void Init(void){
18     freopen("DIJKSTRA.IN","r", stdin);
19     cin>>n;
20     cout<<"Số đỉnh : "<< n<<endl;
21     cin>>s>>t;//nhập đỉnh đầu và đỉnh cuối của đồ thị.
22     //nhập ma trận của đồ thị.
23     for (int i = 1; i <= n; i++){
24         for (int j = 1; j <= n; j++){
25             cin>>Matrix[i][j];
26             if (Matrix[i][j] == 0) Matrix[i][j] = VOCUNG;
27         }
28     }
29 }
30 void Result(void){
31     cout<<"Duong di ngan nhat tu "<<(char)(s+'A'-1)<<" den "<<(char)(t + 'A' -1)<<" la
32     cout<<(char)(t + 'A' - 1)<<"<=";//in đỉnh cuối dưới dạng char.
33     int i = truoc[t];
34     while (i != s){
35         cout<<(char)(i + 'A' -1)<<"<=";//in ra kết quả dưới dạng char.
36         i = truoc[i];
37     }
38     cout<<(char)(s+'A' -1);//in đỉnh đầu dưới dạng char.
39     cout<<endl<<"Do dai duong di la : "<< d[t];
40 }
41 void Dijkstra(void){
42     int u, minp;
43     //khởi tạo nhãn tạm thời cho các đỉnh.
44     for (int v = 1; v <= n; v++){
45         d[v] = Matrix[s][v];
46         truoc[v] = s;
47         chuaxet[v] = FALSE;
48     }
49     truoc[s] = 0;
50     d[s] = 0;
51     chuaxet[s] = TRUE;
52     //bước lặp
53     while (!chuaxet[t]) {
54         minp = VOCUNG;
55         //tìm đỉnh u sao cho d[u] là nhỏ nhất
56         for (int v = 1; v <= n; v++){
57             if ((!chuaxet[v]) && (minp > d[v])){
58                 u = v;
59                 minp = d[v];
60             }
61         }
62         chuaxet[u] = TRUE;// u là đỉnh có nhãn tạm thời nhỏ nhất
63         if (!chuaxet[t]){
64             //gán nhãn lại cho các đỉnh.
65             for (int v = 1; v <= n; v++){
66                 if ((!chuaxet[v]) && (d[u] + Matrix[u][v] < d[v])){
67                     d[v] = d[u] + Matrix[u][v];
68                     truoc[v] = u;
69                 }
70             }
71         }
72     }
}

```

```

73 }
74 void main(void){
75     Init();
76     Dijkstra();
77     Result();
78     getch();
79 }

```

File DIJKSTRA.IN được tải về tại [đây](https://www.dropbox.com/s/rwiqm1a8i7fkkk3/DIJKSTRA.IN?dl=0) (<https://www.dropbox.com/s/rwiqm1a8i7fkkk3/DIJKSTRA.IN?dl=0>).

Output của chương trình.

```

1 So dinh: 9
2 Duong di ngan nhat tu A den I la
3 I<=G<=D<=C<=A
4 Do dai duong di la: 18

```

Thích

4

Cám ơn bạn đã đọc bài viết này. Hãy chia sẻ bài viết và bình luận ý kiến của bạn ở bên dưới.

Share this

g+ Google (<https://plus.google.com/share?url=http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html>)

f Facebook (<https://www.facebook.com/sharer/sharer.php?u=http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html>)

t Twitter (<https://twitter.com/intent/tweet?text=%5BThu%E1%BA%ADt%20to%C3%A1n%5D%20T%C3%ACm%20%C4%91%C6%B0%E1%BB%9Dng%20%C4%91i%20ng%E1%BA%AFn%20nh%E1%BA%A5t%20Dijkstra%20%20Simple%20Code%20C%20Java&url=http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html>)

+ More



Chào mừng bạn đến với SimpleCodeJava Blog - Mục đích của chúng tôi khi thành lập blog này là muốn chia sẻ những kiến thức và kinh nghiệm lập trình mà chúng tôi đã học được với mong muốn giúp đỡ mọi người, giúp bạn rút ngắn được thời gian tìm hiểu cũng như việc giải quyết những vấn đề trong lập trình C và Java.

Bài liên quan

🔗 [\[Thuật toán\] Tìm đường đi ngắn nhất Dijkstra](http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html) (<http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-dijkstra.html>)

Tag : [Dijkstra](http://simplecodejava.blogspot.com/search/label/Dijkstra) (<http://simplecodejava.blogspot.com/search/label/Dijkstra>) , [Thuật toán](http://simplecodejava.blogspot.com/search/label/Thu%E1%BA%ADt%20to%C3%A1n) (<http://simplecodejava.blogspot.com/search/label/Thu%E1%BA%ADt%20to%C3%A1n>) , [thuật toán tìm đường đi ngắn nhất](http://simplecodejava.blogspot.com/search/label/thuat%20toan%20tim%20duong%20di%20ngan%20nh%E1%BA%A5t)



(<http://simplecodejava.blogspot.com/search/label/thu%E1%BA%ADt%20to%C3%A1n%20t%C3%ACm%20%C4%91%C6%B0%E1%BB%9Dng%20%C4%91i%20ng%E1%BA%AFn%20nh%E1%BA%A5t>)

← [Newer Post](http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-floyd.html) (<http://simplecodejava.blogspot.com/2015/10/thuat-toan-tim-uong-i-ngan-nhat-floyd.html>) [Home](#)
(<http://simplecodejava.blogspot.com/2015/10/thuat-toan-prim-tim-cay-khung-nho-nhat.html>)
odecjava.blogspot.com/)

0 bình luận

Sắp xếp theo Cũ nhất



Thêm bình luận...

[Plugin bình luận của Facebook](#)

0 Comment to "[Thuật toán] Tìm đường đi ngắn nhất Dijkstra "

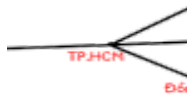
[Post a Comment](https://www.blogger.com/comment.g?blogID=9046094762067454426&postID=2991691592852221416) (<https://www.blogger.com/comment.g?blogID=9046094762067454426&postID=2991691592852221416>)

Subscribe to: [Post Comments \(Atom\)](#)

<http://simplecodejava.blogspot.com/feeds/2991691592852221416/comments/default>

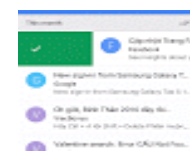
RECENT ★ WEEKLY

COMMENT

 [Đồ thị] Lý thuyết đồ thị - Đường đi, Chu trình, Đồ thị liên thông

(<http://simplecodejava.blogspot.com/2016/04/do-thi-duong-di-chi-trinh-lien-thong.html>)

Đồ thị: là một cấu trúc dữ liệu rời rạc bao...



[Android] Swipe RecyclerView cho Android
(<http://simplecodejava.blogspot.com/2016/02/android-swipe-recyleview-cho-android.html>)

Sau một thời gian sử dụng sử dụng Gmail và...

[Android] Thư viện TimePicker & DatePicker đẹp cho android. ^



([http://simplecodej.ava.blogspot.com/2016/02/android-thu-vien-timepicker-](http://simplecodej.ava.blogspot.com/2016/02/android-thu-vien-timepicker-datepicker.html)

[datepicker.html](http://simplecodej.ava.blogspot.com/2016/02/android-thu-vien-timepicker-datepicker.html))

Qua một thời gian sử dụng Google calendar mình...



[Java] Copy dữ liệu sang file khác trong Java
([http://simplecodej.ava.blogspot.com/2016/01/java-copy-](http://simplecodej.ava.blogspot.com/2016/01/java-copy-du-lieu-sang-file-khac-trong-java.html)

[du-lieu-sang-file-khac-trong.html](http://simplecodej.ava.blogspot.com/2016/01/java-copy-du-lieu-sang-file-khac-trong-java.html))

Bài viết sẽ trình bày cách copy dữ liệu từ một...



[Java] Lấy IP trong Java
([http://simplecodej.ava.blogspot.com/20](http://simplecodej.ava.blogspot.com/2016/01/java-lay-ip-trong-java.html)

[16/01/java-lay-ip-trong-java.html](http://simplecodej.ava.blogspot.com/2016/01/java-lay-ip-trong-java.html))

Bài viết sẽ hướng dẫn bạn lấy địa chỉ IP từ...



[Java] Chuyển từ hệ thập phân sang hệ nhị phân.
([http://simplecodej.ava.blogspot.com/2016/01/java-](http://simplecodej.ava.blogspot.com/2016/01/java-chuyen-tu-he-thap-phan-sang-he-nhi-phan.html)

[chuyen-tu-he-thap-phan-sang-he-nhi.html](http://simplecodej.ava.blogspot.com/2016/01/java-chuyen-tu-he-thap-phan-sang-he-nhi-phan.html))

Có 3 cách để chuyển một số từ hệ thập phân...



[Java] Sự khác nhau giữa HashMap và Hashtable
([http://simplecodej.ava.blogspot.com/2016/01/java-su-](http://simplecodej.ava.blogspot.com/2016/01/java-su-khac-nhau-giua-hashmap-va-hashtable.html)

[khac-nhau-giua-hashmap-va.html](http://simplecodej.ava.blogspot.com/2016/01/java-su-khac-nhau-giua-hashmap-va-hashtable.html))

Sự khác nhau giữa HashMap và Hashtable ? Đây...



[Java] Tạo file trong Java
([http://simplecodej.ava.blogspot.com/20](http://simplecodej.ava.blogspot.com/2015/12/java-tao-file-trong-java.html)

[15/12/java-tao-file-trong-java.html](http://simplecodej.ava.blogspot.com/2015/12/java-tao-file-trong-java.html))

Bài viết này sẽ trình bày cách tạo một file trong...



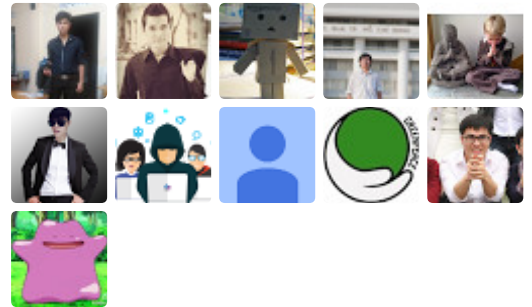
FOLLOW US ON FACEBOOK



GOOGLE+ FOLLOWERS

Admin Blog

Add to circles



13 have me in circles

[View all](#)

FOLLOW BY EMAIL

Email address...

Submit

