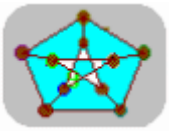
Lý Thuyết Đồ Thị
Đại Học Khoa Học Tự Nhiên Tp.HCM
227 Nguyễn Văn Cừ, Q5

GVHD: Lê Ngọc Thành
E-mail: lnthanh@fit.hcmuns.edu.vn
http://courses.cs.hcmuns.edu.vn



Tìm đường đi ngắn nhất với Dijkstra

1.Thuật toán Dijkstra

Cho G=(X,E) là một đồ thị có trọng không âm gồm n đỉnh. Thuật toán Dijkstra được dùng để tìm đường đi ngắn nhất từ đỉnh i đến j cho trước.
Gọi L là ma trận trọng lượng (với qui ước L_{hk} = +∞ nếu không có cạnh nối từ đỉnh h đến đỉnh k).
Ta sử dụng thêm hai mảng để lưu vết của quá trình tìm đường đi:

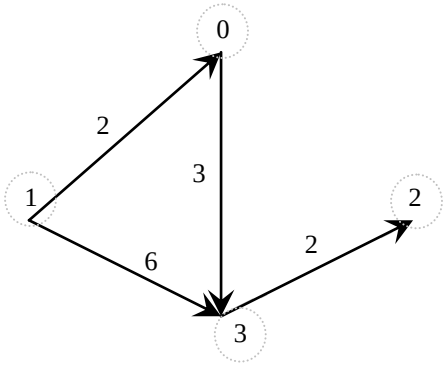
- Dodai[…] : lưu độ dài từ đỉnh đầu i đến các đỉnh trong đồ thị.
- Nhan[…] : lưu đỉnh liền trước nó trên đường đi.

Bước 1: Gán T:= X và gán các nhãn:
Dodai[i] = 0;
Dodai[k] = +∞ với $\forall k \in X \setminus \{i\}$;
Nhan[k] = -1; $\forall k \in X$;
Bước 2: Nếu j $\notin$ T thì dừng và giá trị Dodai[j] là độ dài đường đi ngắn nhất từ i đến j và Nhan[j] lưu đỉnh nằm ngay trước j trên đường đi đó.
Bước 3: Chọn đỉnh v $\in$ T sao cho Dodai[v] nhỏ nhất và gán T:= T\{v}.
Bước 4: $\forall k \in T$ và có cạnh nối từ v đến k:
Nếu Dodai[k] > Dodai[v] + L_{vk} thì
Dodai[k] = Dodai[v] + L_{vk};
Nhan[k] = v;
Cuối nếu
Cuối với mọi
Trở về bước 2

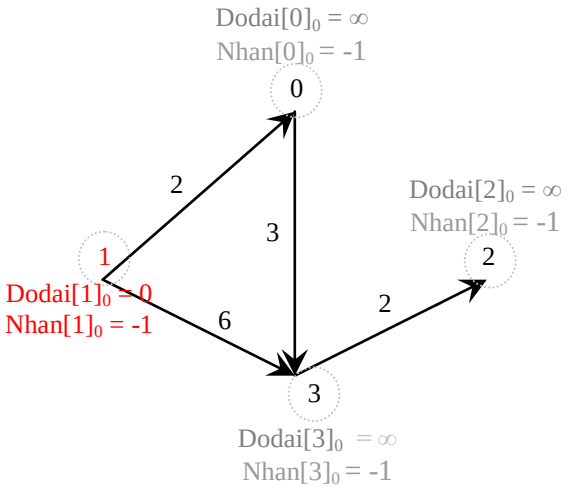
Chú ý: Khi thuật toán dừng, nếu Dodai[j] = +∞ thì không tồn tại đường đi từ i đến j, nếu ngược lại thì Dodai[j] là độ dài đường đi ngắn nhất.

2. Ví dụ Dijkstra

Cho đồ thị sau:



Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2 trong đồ thị.



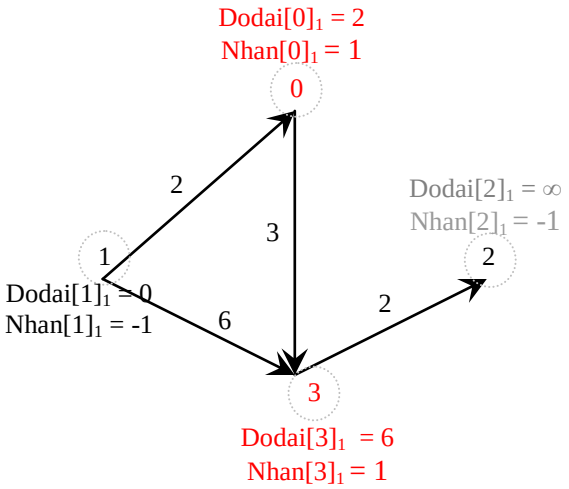
Bước 1: khởi tạo

Khởi tạo đỉnh 1 với độ dài min hiện tại là 0, và nhãn đỉnh trước là chính nó (hoặc -1, điều này không quan trọng). Các đỉnh còn lại đều được gán độ dài min là ∞. Dưới đây là bảng mô tả

T[]\step	0
0	0
1	1
2	2
3	3

Dodai[]\step	0
0	∞
1	0
2	∞
3	∞

Nhan[]\step	0
0	-1
1	-1
2	-1
3	-1



Bước 2: đỉnh 2 vẫn thuộc T ta sang bước kế tiếp.

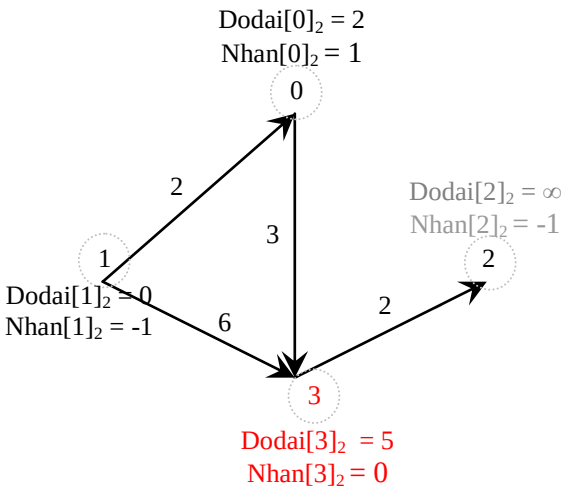
Bước 3: chọn đỉnh có độ dài nó nhỏ nhất. Ở đây là đỉnh 1, ta loại đỉnh này ra.

T[]\step	0	1
0	0	0
1	1	~~1~~
2	2	2
3	3	3

Bước 4: Các đỉnh còn lại đều có Dodai[] cực đại nên ta cập nhật lại như sau

Dodai[]\step	0	1
0	∞	2
1	0	0
2	∞	∞
3	∞	6

Nhan[]\step	0	1
0	-1	1
1	-1	-1
2	-1	-1
3	-1	1



Bước 2 (lần 2): đỉnh 2 vẫn thuộc T ta sang bước kế tiếp.

Bước 3 (lần 2): trong **T**, chọn đỉnh có độ dài nhỏ nhất. Ở đây là đỉnh 0, nên:

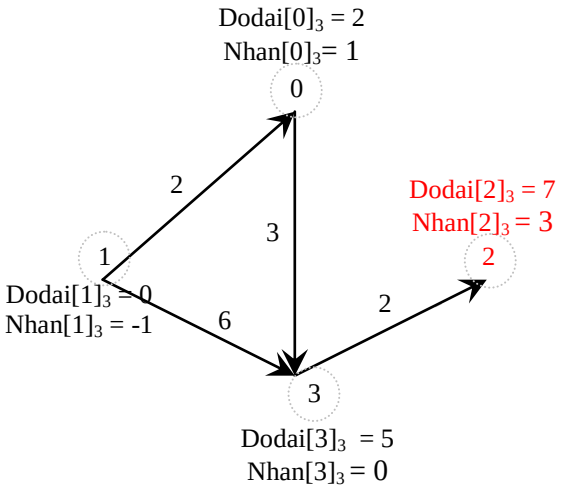
T[]\step	0	1	2
0	0	0	~~0~~
1	1		
2	2	2	2
3	3	3	3

Bước 4 (lần 2): tính độ dài từ đỉnh 0 vừa xét ở trên đến các đỉnh còn lại trong T.

Đỉnh số 3 có chi phí mới là 2+3=5 nhỏ hơn chi phí cũ (6), vì vậy ta cập nhật lại đỉnh này.

Dodai[]\step	0	1	2
0	∞	2	2
1	0	0	0
2	∞	∞	∞
3	∞	6	5

Nhan[]\step	0	1	2
0	-1	1	1
1	-1	-1	-1
2	-1	-1	-1
3	-1	1	0



Bước 2 (lần 3): đỉnh 2 vẫn thuộc T ta sang bước kế tiếp.

Bước 3 (lần 3): lấy đỉnh 3 ra khỏi **T** vì có độ dài nhỏ nhất:

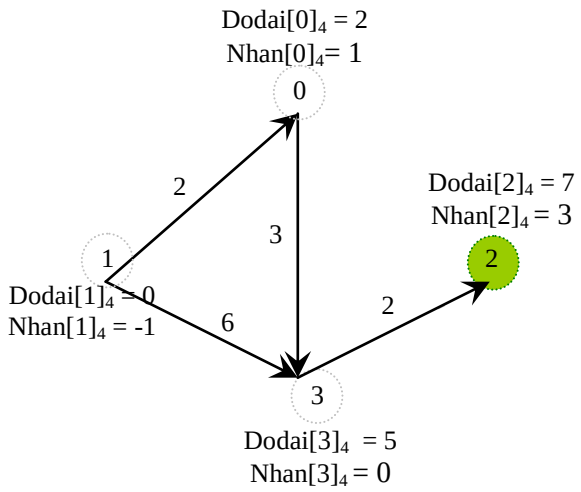
T[]\step	0	1	2	3
0	0	0		
1	1			
2	2	2	2	2
3	3	3	3	~~3~~

Bước 4 (lần 3): tương tự ta cũng tính độ dài từ đỉnh 3 đến đỉnh còn lại trong T.

Đỉnh số 2 có chi phí mới là 5+2=7 < ∞.

Dodai[]\step	0	1	2	3
0	∞	2	2	2
1	0	0	0	0
2	∞	∞	∞	7
3	∞	6	5	5

Nhan[]\step	0	1	2	3
0	-1	1	1	1
1	-1	-1	-1	-1
2	-1	-1	-1	3
3	-1	1	0	0



Bước 2 (lần 4): đỉnh 2 vẫn thuộc T.

Bước 3 (lần 4): lấy đỉnh 2 ra khỏi **T** vì có độ dài nhỏ nhất, lúc này

T[]\step	0	1	2	3	4
0	0	0			
1	1				
2	2	2	2	2	~~2~~
3	3	3	3		

Lúc này đỉnh 2 không còn nằm trong T nữa, chúng ta kết thúc thuật toán.

3. Hướng dẫn cài đặt Dijkstra

```
#define MAX 100
#define VOCUC -1 //có thể sử dụng trị MAXINT (32767)

typedef struct {
        int n;
        int L[MAX][MAX]; // ma trận trọng số
}GRAPH;

int nT;                                 //số đỉnh của đồ thị
int T[MAX];                             // mảng đánh dấu tập hợp
int Dodai[MAX];                         //độ dài từ đỉnh đầu đến các đỉnh trong đồ thị
int Nhan[MAX];                          //đỉnh liền trước trên đường đi

/*Khởi tạo các thông số cho thuật toán*/
void Init(GRAPH g, int dinhDau)
{
        nT = g.n;
        for (i…)
        {
              T[i] = 1;      //tất cả các đỉnh đều nằm trong T
              Dodai[i] = VOCUC;
              Nhan[i] = -1;
        }
        // khởi tạo cho đỉnh đầu
        Dodai[dinhDau] = 0;
}

…
/*Thuật toán Dijstra*/
void DijkstraAlg(GRAPH g,int dinhDau, int dinhCuoi)
{
        //Khoi tao cac thong so cho thuat toan
        ….
        // Trong khi đỉnh cuối vẫn trong T
```

```
        while {T[j] ==…)
        {
                // Tìm đỉnh có độ dài nhỏ nhất trong T
                min = VOCUC; v = -1;
                for (i…)
                    if (T[i] != … && min >= ….)
                    {
                        min = …;
                        v = i;
                    }
                //Nếu không tìm thấy, dừng thuật toán, kết luận không có đương đi từ ….
                if….break;
                //Loại v ra khỏi tập T
                T[v]= 0;

                //Duyệt các đỉnh có cạnh nối từ v đến (chú ý: đồ thị có hướng)
                for (k….)
                        //Chú ý: tùy thuộc vào trị VOCUC, ta có những phép so sánh khác nhau
                        if (g…. && …. > ….)
                        {
                                Dodai[k] = ….;
                                Nhan[k] = v;
                        }
        }
        …
}

void PrintScreen()
{
        //Sử dụng mảng Dodai và Nhan để xuất ra đường đi và chi phí
        ….
}
```