HƯỚNG DẪN THỰC HÀNH
TÌM ĐƯỜNG ĐI NGẮN NHẤT – DIJKSTRA

1 Thuật toán Dijkstra

Dijkstra là thuật toán tìm đường đi ngắn nhất (ĐĐNN) giữa hai đỉnh của một đồ thị có trọng số. Trong phần hướng dẫn này, ta sẽ xem xét thuật toán trên đồ thị đơn, có trọng số không âm.

Cho đồ thị đơn $G = (V, E)$ có trọng số không âm gồm N đỉnh. ĐĐNN từ s đến g sẽ được xác định như sau:

- Gọi W là ma trận trọng số của đồ thị G, $W[i][j] = +\infty$ nếu không có cạnh nối từ đỉnh thứ i đến đỉnh thứ j.
- $ChiPhi$ là mảng lưu giữ trọng lượng đường đi ngắn nhất từ s đến các đỉnh khác trong đồ thị. $ChiPhi[k]$ là trọng lượng ĐĐNN từ s đến đỉnh thứ k trong đồ thi G.
- $Nhan$ là mảng lưu vết của ĐĐNN. $Nhan[k]$ sẽ lưu giữ đỉnh phía trước đỉnh k trong ĐĐNN.

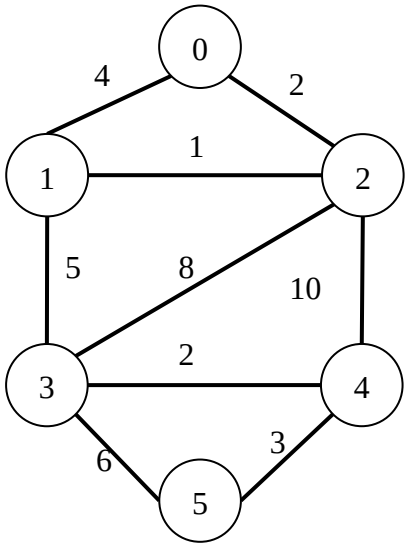
```
Bước 1: Gán T:= V và thực hiện gán nhãn
For i:= 0 to N-1
   ChiPhi[i] = +∞;
   Nhan[i] = -1; //-1 nghĩa là không có đỉnh nào trước nó.
ChiPhi[s] = 0;              // xuất phát từ s.

Bước 2:  Trong khi g ∈ T:
Chọn đỉnh v ∈ T sao cho L[v] là nhỏ nhất;
T:= T\{v};
For tất cả các đỉnh k ∈ T và có cạnh nối từ v đến k
   If ChiPhi[k] > ChiPhi[v] + W[v][k] then
        ChiPhi[k] = ChiPhi[v] + W[v][k];
        Nhan[k] = v;
   End if
End For
```

Bước 4: Xuất phát từ Nhan[g], lần ngược trở lại để tìm đường đi ngắn nhất từ s đến g.

Chú ý: khi thuật toán dừng, nếu $ChiPhi[g] = +\infty$ thì không tồn tại đường đi từ s đến g. Ngược lại thì $ChiPhi[g]$ chính là trọng lượng của ĐĐNN.

2 Ví dụ

Cho đồ thị G như hình sau:



Dưới đây minh họa các bước chạy thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh thứ 0 đến đỉnh thứ 5

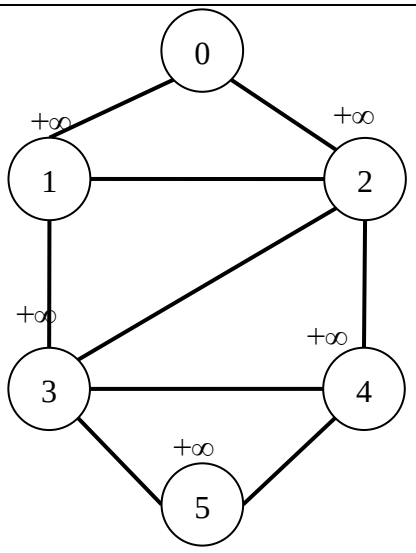
Gán nhãn:
T = {0, 1, 2, 3, 4, 5}

ChiPhi[]

0	+∞	+∞	+∞	+∞	+∞

Nhan[]

-1	-1	-1	-1	-1	-1

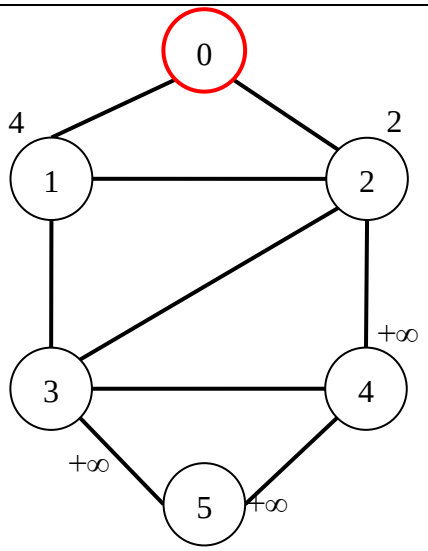


v = 0
T = {1, 2, 3, 4, 5}
ChiPhi[]

0	**4**	**2**	+∞	+∞	+∞

Nhan[]

-1	**0**	**0**	-1	-1	-1

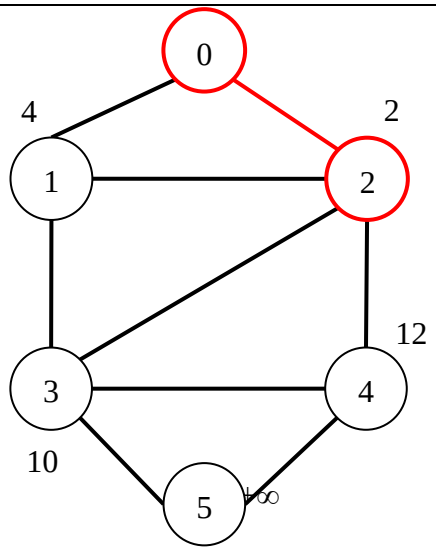


v = 2
T = {1, 3, 4, 5}
ChiPhi[]

0	**3**	2	**10**	**12**	+∞

Nhan[]

-1	**2**	0	**2**	**2**	-1

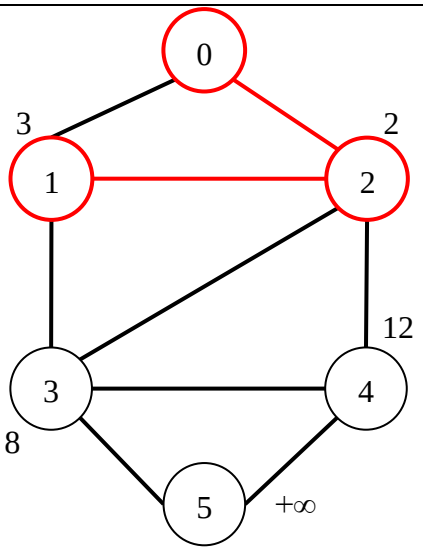


v = 1
T = {3, 4, 5}
ChiPhi[]

0	3	2	**8**	12	+∞

Nhan[]

-1	2	0	**1**	2	-1

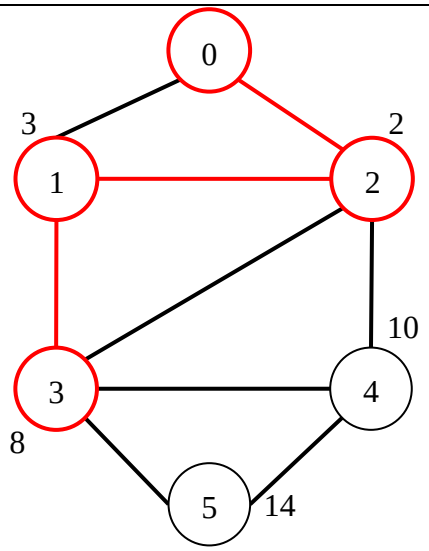


v = 3
T = {4, 5}
ChiPhi[]

0	3	2	8	**10**	**14**

Nhan[]

-1	2	0	1	**3**	**3**

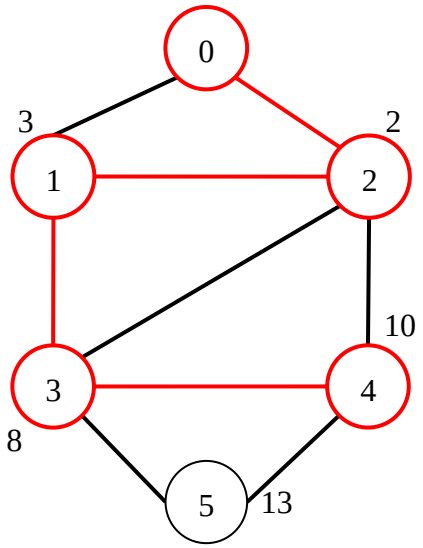


v = 4
T = {5}
ChiPhi[]

0	3	2	8	10	**13**

Nhan[]

-1	2	0	1	3	**4**

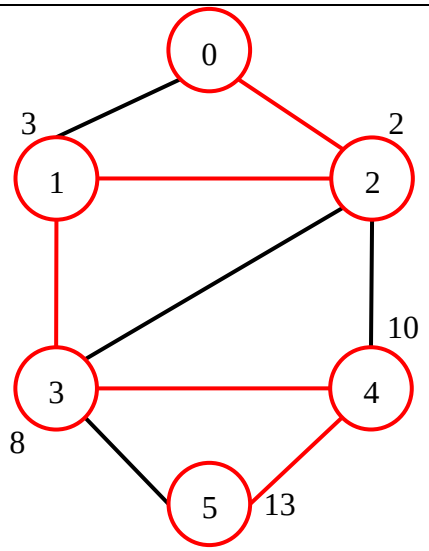


v = 5
T = {}
ChiPhi[]

0	3	2	8	10	13

Nhan[]

-1	2	0	1	3	4



3 Hướng dẫn cài đặt

```
void Graph::TimDuongDiNganNhat(int s, int g, int Nhan[], int ChiPhi[])
{
    int soDinh = this->adjacentMatrix.size;
    // Bước 1: Khởi tạo các giá trị cho ChiPhi, Nhan và T
    bool* T = new bool[soDinh];
    for(int i = 0;i < soDinh; i++)
    {
        T[i] = true;    // Đỉnh i thuộc T.
        ChiPhi[i] = VOCUC;
        Nhan[i] = -1;
    }
    // Gán chi phí của đỉnh s là 0.
    ChiPhi[s] = 0;

    // Bước 2:
    // Trong khi đỉnh g vẫn chưa thuộc T.
    while(....)
    {
        // 2.1 Tìm đỉnh thuộc T có chi phí nhỏ nhất.
        int minDinh = -1;
        for(int i = 0; i < soDinh; i++)
            // Nếu i là đỉnh chưa thuộc T và có chi phí nhỏ hơn minDinh.
            // thì minDinh là i.
            if(.... && (minDinh == -1 || ....))
              minDinh = i;

       // 2.2 Cập nhật ChiPhi và Nhan theo đỉnh vừa tìm được
       if(minDinh != -1)
       {
           // Loại minDinh ra khỏi T.
           T[minDinh] = ...;
           for(int k = 0; k < soDinh; k++)
           {
               // Nếu k vẫn thuộc T và có cạnh nối từ k đến minDinh
               if(.... && .....)
                   if(ChiPhi[k] > ChiPhi[minDinh] + this->adjacentMatrix.weights[minDinh][k])
                   {
                       // Cập nhật chi phí của đỉnh k.
                       ChiPhi[k] = ....;
                       // Cập nhật đỉnh trước đỉnh k
                       Nhan[k] = ....;
                   }
           }
       }
    }
    // Giải phóng bộ nhớ cho T.
    delete T;

}
```

Khi đó, đường đi ngắn nhất và chi phí lần lượt được lưu giữa trong mảng Nhan[] và ChiPhi[].

Sinh viên có thể định nghĩa biến $VOCUC$ bằng một giá trị lớn. Ví dụ:
$\#define\ VOCUC\ 100000$