**Đồ thị**

**I. Chu trình Euler:**

* Chu trình đơn chứa tất cả các cạnh của đồ thị G được gọi là chu trình Eluer (Eulerian ciruit – Euler ciruit).
* Đường đi Euler trong G là đường đi đơn chứa mọi cạnh của G (Euler path).
* Đồ thì có chu trình Euler được gọi là đồ thị Euler (Euler graph).
* Đồ thị có đường đi Eluer được gọi là đồ thị nửa Euler (Semi-euler graph).
* Một đa đồ thị liên thông có chu trình Euler khi và chỉ khi mỗi đỉnh của nó đều có bậc chẵn.
* Đồ thị vô hướng liên thông G là nữa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

Thuật toán tìm chi trình Euler

* Bước khởi tạo:
  + Tìm chu trình C trong G
  + Loại khỏi G các cạnh trong chu trình C
* Bước lặp: Trong khi G khác rỗng thực hiện các bước sau:
  + 3.1: Tìm chu trình C’ trong G có đỉnh bắt đầu thuộc C.
  + 3.2: Loại bỏ khỏi G các đỉnh cô lập và các cạnh của C’.
  + 3.3: Chèn vào C chu trình C, ở vị trí thích hợp.
* Kết thúc thuật toán ta có C chình là chu trình Euler cần tìm.

Thuật toán Fleury tìm chu trình Euler:

* Bước 1: Chọn một đỉnh u tùy ý để bắt đầu.
* Bước 2: Chọn một cạnh để đi tiếp, chỉ chọn cạnh cầu khi nào không còn lựa chọn nào khác.
* Bước 3: Đánh dấu cạnh đó đã đi qua cho biết ta không thể quay lại cạnh đó.
* Bước 4: Đi theo cạnh đó đến đỉnh tiếp theo.
* Bước 5: Lặp lại bước 2 cho đến khi nào mọi cạnh đều đã được duyệt.

**II. Đường đi Hamilton:**

1. Định nghĩa:

* Đường đi x0,x1,x2, .. ,xn-1,xn trong đồ thị G=(V,E) được gọi là đường đi Hamilton (Hamiltonia path, Hamilton path) nếu V = { x0,x1,x2, .. ,xn-1,xn } và xi != xj, với 0 <= i < j <= n.
* Chu trình x0,x1,…,xn-1, xn,x0 (n > 1) trong đồ thị G = (V, E) được gọi là chu trình Hamilton nếu x0,x1,x2, .. ,xn-1,xn là đường đi Hamilton.
* Đồ thị được gọi là đồ thị Hamilton nếu nó chứa chu trình Hamilton và gọi là nửa Hamilton nếu nó chứa đường đi Hamilton.

**III. Bài toán đường đi ngắn nhất**

* Định nghĩa đường đi có trọng số:

Cho đồ thị G = (V, E) là đồ thị có trọng số và trọng số mỗi cạnh là w(e). Với G’ là một đồ thị con của G thì trọng số của G’ được định nghĩa là:

w(G’) =

* + Nếu G’ là đường đi hay chu trình thì w(G’) gọi là độ dài của G’
  + Nếu G’ là một mạch (chu trình có các đỉnh không lặp lại) và w(G’) < 0 thì ta gọi G’ là mạch âm.
* Phát biểu bài toán:
  + Cho đồ thị G = (V, E) là một đồ thị có trọng số và s, t ∊ V. Gọi P là tập hợp tất cả các đường đi từ s tới t.

Xét bài toán:

Tìm p0 ∊ P sao cho p0 = min{w(p): p ∊ P}

* + Bài toán này gọi là bài toán đường đi ngắn nhất (shorted path problem) và p0 gọi là đường đi ngắn nhất (shorted path) từ s đến t.
* Nguyên lý Bellman:
  + Hầu hết các thuật toán tìm đường đi ngắn nhất đều đặt trên cơ sở trên nguyên lý Bellman. Đây là nguyên lý tổng quát hóa cho bài toán tối ưu hóa rời rạc, được nhà toán học người Mỹ Richard Ernest Bellman (1920 -1984) đưa ra vào năm 1953. Nguyên lý này còn được gọi là nguyên lý quy hoạch động Bellman.
  + Đối với trường hợp bài toán đường đi ngăn nhất thì có thể trình bày nguyên lý này như sau:
    - Giả sử P là đường di ngắn nhất từ đỉnh u đến đỉnh v và k là một đỉnh nằm trên đường đi P.
    - Giả sử P = P1 (+) P2 với P1 là đường đi con của P từ u đến k và P2 là đường đi con của P từ k đến v.
    - Nguyên lý Bellman nói rằng P1 cũng là đường đi ngắn nhất từ u đến k, vì nếu có một đường đi khác là P1 ‘ từ u đến k có trọng số nhỏ hơn P1 thì P1’ (+) P2 là đường đi từ u đến v mà có trọng số nhỏ hơn P, điều này mâu thuẫn tới tính ngắn nhất của P.
  + Điều kiện tồn tại lời giải:
    - Gọi P là một đường đi từ u đến v, giả sử P chứa một mạch µ. Có 2 trường hợp sau đây:
      * Nếu L(µ) >= 0 thì có thể cải tiến đường đi P bằng cách bỏ đi mạch µ.
      * Nếu L(µ) < 0 thì không tồn tại đường đi ngắn nhất từ đỉnh u đến v vì nếu quay vòng tại µ càng nhiều vòng thì trọng lượng đường đi P càng nhỏ đi, tức là L(P) 🡪 -∞.
* Thuật toán Dijkstra:
  + Dữ liệu ra/vào:
    - Xét đồ thị G = (V, E) có ***trọng số giả sử không âm.***
      * Dữ liệu nhập vào cho thuật toán là ma trận trọng lượng D và hai đỉnh u, v cho trước.
      * Dữ liệu xuất là đường đi ngắn nhất từ u đến v.
  + Thuật toán Dijkstra
    - Bước 1. Gán T = V và gán các nhãn:  
      L[u] = 0; L[k] = +∞, Với mọi k ∊ V\{u};

Prev[k] = -1, với mọi k ∊ V.

* + - Bước 2. Nếu v !∊ T thì dừng lại và giá trị L[v] chính là độ dài đường đi ngắn nhất u đến v và Prev[v] là đỉnh nằm ngay đước v trên đường đi đó.
    - Bước 3. Chọn đỉnh i ∊ sao cho L[i] nhỏ nhất và gán T = T\{i}.
    - Bước 4.
      * Với mọi k ∊ T và từ đỉnh i (ở bước 3) đến đỉnh k có cạnh nối:  
        nếu L[k] > L[i] + Dik thì Gán L[k] = L[i] + Dik và Prev[k] = i
      * Trở về bước 2.
  + Ví dụ Dijkstra
    - Ví dụ 1:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T | S | 2 | 3 | 4 | 5 | T |
| L | 0 | +∞ | +∞ | +∞ | +∞ | +∞ |
| Prev | -1 | -1 | -1 | -1 | -1 | -1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  | 2 | 3 | 4 | 5 | T |
| L | 0 | 6 | +∞ | 4 | +∞ | +∞ |
| Prev | -1 | S | -1 | s | -1 | -1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  | 2 | 3 |  | 5 | T |
| L | 0 | 6 | 7 | 4 | 13 | +∞ |
| Prev | -1 | S | 4 | s | 4 | -1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  |  | 3 |  | 5 | T |
| L | 0 | 6 | 7 | 4 | 13 | +∞ |
| Prev | -1 | S | 4 | s | 4 | -1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  |  |  |  | 5 | T |
| L | 0 | 6 | 7 | 4 | 9 | 9 |
| Prev | -1 | S | 4 | s | 3 | 3 |

3

4

2

3

9

2

3

5

6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  |  |  |  |  | T |
| L | 0 | 6 | 7 | 4 | 9 | 9 |
| Prev | -1 | S | 4 | s | 3 | 3 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | S | 2 | 3 | 4 | 5 | t |
| T |  |  |  |  |  | T |
| L | 0 | 6 | 7 | 4 | 9 | 9 |
| Prev | -1 | S | 4 | s | 3 | 3 |

* Thuật toán Bellman:
  + hay còn gọi là Bellman – Ford do Bellman và Ford đưa ra.
  + Thuật toán này tìm đường đi ngắn nhất từ một đỉnh của đồ thị đến mỗi đỉnh khác nếu đồ thị không có mạch âm.
  + Nếu phát hiện đồ thị có mạch âm thì thuật toán dừng. Dữ liệu nhập cho thuật toán là ma trận trọng lượng D.
  + Thuật toán Bellman:
    - Cho trước đỉnh x ∊ V.
    - Bước 1. Khởi tạo:
      * Π(0,x) = 0; π(0,j) = +∞, với mọi i != x; k = 1.
    - Bước 2. Với mỗi i ∊ V ta đặt:
      * Π(k,i) = min({π(k-1,i)} U {π(k-1,j) + Dji})
    - Bước 3.
      * Nếu π(k,i) = π(k-1, i) với mọi i ∊ V thì π(k, i) chính là độ dài đường đi ngắn từ x đến i.
      * Ngược lại nếu k < n thì tăng k = k + 1 và trở lại bước 2;
      * Nếu k = n thì dừng vì từ x đi tới được một mạch âm.