TOÁN RỜI RẠC
Bài toán cây khung nhỏ nhất
Đường đi ngắn nhất trên đồ thị

NGUYỄN HẢI TRIỀU[1]

[1]Bộ môn Kỹ thuật phần mềm,

Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, February 2022

Tổng quan

1. **Cây khung nhỏ nhất của đồ thị trọng số**
2. Đường đi ngắn nhất trên đồ thị

Định nghĩa 1.1 (Minimum Spanning Tree)

Giả sử $G = (V, E)$ là một đồ thị đơn vô hướng liên thông gồm n đỉnh và có trọng số cạnh. Cây khung nhỏ nhất của G là cây khung có tổng trọng số nhỏ nhất.

Cây khung nhỏ nhất của đồ thị có nhiều ứng dụng trong đời sống:

- xây dựng hệ thống Metro trong thành phố
- xây dựng một mạng máy tính với chi phí nối mạng là nhỏ nhất

Hai thuật toán xác định cây khung nhỏ nhất trên đồ thị là **Kruskal** và Kruskal. Các thuật toán này đều có độ phức tạp thời gian $O(n^2)$.

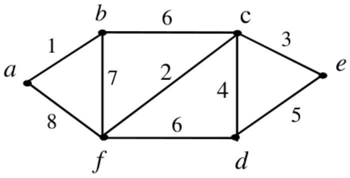
Thuật toán Kruskal

Cho đồ thị n đỉnh, liên thông, có trọng số $G = (V, E)$:

1. $T = \emptyset$
2. sắp xếp các cạnh của đồ thị theo thứ tự không giảm theo trọng số
3. lần lượt ghép các cạnh theo thứ tự trong danh sách thêm vào T sao cho không tạo ra chu trình trong T và đủ $n-1$ cạnh.

Algorithm 1: Thuật toán Kruskal

```
Input   : Đồ thị G(V, E)
Output: T - Cây khung nhỏ nhất
Function Kruskal-MST
    Initial  : T = ∅
    for i ← 1 to n − 1 do
        Chọn e là cạnh có trọng số nhỏ nhất trong E
        if T ∪ {e} không chứa chu trình then
            T := T ∪ {e}
            E := E \ e
        end
    end
end
```



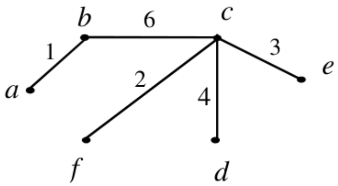
(G)

Hình 1: Sử dụng thuật toán Kruskal, xác định cây khung nhỏ nhất của đồ thị G

Ví dụ hình 1

- lập danh sách các cạnh theo thứ tự tăng dần trọng số $DS = \{ab, fc, ce, cd, de, bcfd, bf, af\}$.
- lần lượt ghép các cạnh ở đầu danh sách vào T sao cho không tạo thành chu trình và **đủ 5 cạnh**
- cây khung nhỏ nhất của đồ thị đã cho là

$$T = \{ab, fc, ce, cd, bc\}$$



(T)

Thuật toán Prim

Cho đồ thị n đỉnh, liên thông, có trọng số $G = (V, E)$.

1. $W = u$, $T = \emptyset$, trong đó u là đỉnh bất kỳ thuộc V
2. Lập $E^* = \{vw|vw \in E; v \in W; w \notin W\}$
3. Xét $e = minE^* = v^*w^*$

$$W = W \cup \{w^*\}; T = T \cup \{e\}$$

4. Nếu $W = V \Rightarrow T$ chứa cạnh của cây khung nhỏ nhất. Nếu $W < V \Rightarrow$ quay lại bước 2

Algorithm 2: Thuật toán Prim

Input : Đồ thị $G(V, E)$

Output: T - Cây khung nhỏ nhất

Function `Prim-MST`

  Initial : $T = \emptyset$

  for $i \leftarrow 1$ **to** $n-2$ **do**

    Chọn e là cạnh có trọng số nhỏ nhất trong E và liên thuộc với T

    if $T \cup \{e\}$ *không chứa chu trình* **then**

      $T := T \cup \{e\}$

      $E := E \setminus e$

    end

  end

end

Xét lại đồ thị G ở hình 1. Kết quả tính toán của thuật toán Prim cho đồ thị G xuất phát từ đỉnh a được ghi lại theo như bảng dưới đây và cây khung nhỏ nhất của G là $T = \{ab, bc, cf, cd, ce\}$, $l(T) = 16$.

Bước lặp	a	b	c	d	e	f	T
1	-	$\boldsymbol{a,1}$	$a,\ \infty$	$a,\ \infty$	$a,\ \infty$	$a,8$	$T = \{\}$
2	-	-	$\boldsymbol{b,6}$	$a,\ \infty$	$a,\ \infty$	$b,7$	ab
3	-	-	-	$c,4$	$c,3$	$\boldsymbol{c,2}$	bc
4	-	-	-	$c,4$	$\boldsymbol{c,3}$	-	cf
5	-	-	-	$\boldsymbol{c,4}$	-	-	ce
6	-	-	-	-	-	-	cd

Bảng 1: Ví dụ thuật toán Prim cho hình 1

Lưu ý: những đỉnh không kề với a sẽ được gán trọng số là ∞ để loại bỏ các cạnh không liên thuộc. Nếu có nhiều đỉnh có trọng số nhỏ nhất bằng nhau $\Rightarrow$ chọn một đỉnh tùy ý trong số đó làm đỉnh chọn $\Rightarrow$ *một đồ thị có thể có nhiều cây khung nhỏ nhất.*

So sánh thuật toán Kruskal và Prim

Prim

- chọn các cạnh có trọng số nhỏ nhất liên thuộc với các đỉnh thuộc cây khung T đang xây dựng và không tạo ra chu trình
- *đặc biệt ở bước trung gian các cạnh được bổ sung vào cây khung T luôn tạo thành một cây con*

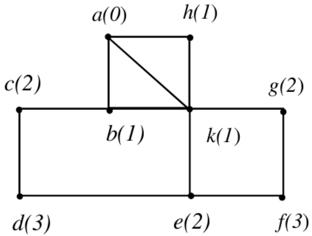
Kruskal

- chọn cạnh có trọng số nhỏ nhất trong danh sách các cạnh mà không tạo ra chu trình
- *ở bước trung gian, các cạnh của T có thể là một rừng*

- Bài toán xác định đường đi ngắn nhất trong thành phố, quy hoạch hệ thống giao thông, lập lịch thi công công trình...
- Hai thuật toán cơ bản tìm đường đi ngắn nhất giữa các đỉnh trên đồ thị, đó là **thuật toán Dijkstra** (1959) và **thuật toán Floyd** (1963).

Đường đi ngắn nhất theo số cạnh của đồ thị

Cho một đồ thị đơn vô hướng liên thông. Tìm đường đi ngắn nhất theo số cạnh từ đỉnh a đến đỉnh b của đồ thị. Có thể giải quyết bài toán này bằng cách vận dụng thuật toán BFS.



Ví dụ 2.1

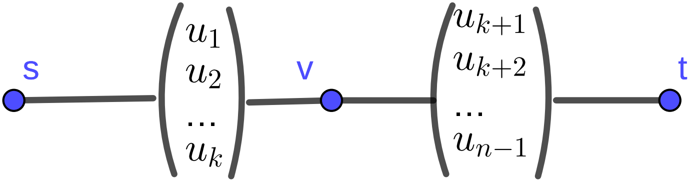
Tìm đường đi ngắn nhất từ đỉnh a đến đỉnh d của đồ thị ở hình trên.

Đường đi ngắn nhất trên đồ thị trọng số

Điều kiện để tìm đường đi ngắn nhất từ đỉnh $s \to t$ có lời giải

1. Phải tồn tại đường đi từ s tới t
   - Đồ thị vô hướng liên thông hoặc đồ thị có hướng liên thông mạnh
   - Đồ thị vô hướng, trong đó s, t thuộc một thành phần liên thông
   - Đồ thị có hướng có đường đi từ s tới t
2. Đồ thị không chứa chu trình âm
   - Đồ thị vô hướng không có cạnh âm
   - Đồ thị có hướng không có chu trình âm

Ý tưởng



- Tại đỉnh cần xét, ta thử đường đi qua các đỉnh trung gian để đến đích
- Cập nhật đường đi ngắn nhất nếu tìm được đường đi tốt hơn đường đi tốt nhất đã biết.
- Thông tin cần lưu:
  - Độ dài đường đi ngắn nhất từ $s \to t$ đã biết
  - Đỉnh nằm trước v trên đường đi ngắn nhất đã biết.

Thuật toán Dijkstra

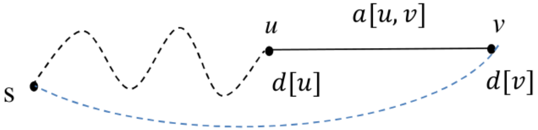
Cho đồ thị đơn vô hướng liên thông G có trọng số dương. Tìm đường từ đỉnh s đến t của G sao cho tổng trọng số của đường đi này là nhỏ nhất.

Dijkstra-nhà khoa học máy tính người Hà Lan đã đề xuất thuật toán tìm đường đi ngắn nhất từ một đỉnh đến tất các các đỉnh còn lại trong đồ thị vô hướng có trọng số dương.

Ký hiệu

- $d[t]$ là nhãn của đỉnh t, độ dài ngắn nhất của đường đi từ s đến t
- $a[s,t]$ trọng số cạnh/cung từ s đến t

Thuật toán Dijkstra



Hình 2: Đường đi ngắn nhất từ đỉnh s đến đỉnh v thông qua u

Theo như hình trên, nhãn của đỉnh v là $d[v]$ sẽ được gán nhãn mới là:

$$d[v] := min(d[v], d[u] + a[u, v]) \tag{1}$$

Nghĩa là, ở mỗi bước gán nhãn cho đỉnh, *nếu nhãn mới lớn hơn hoặc bằng nhãn cũ thì giữa nguyên nhãn cũ, ngược lại thì thay nhãn cũ bằng nhãn mới.*

Thuật toán Dijkstra

Mô tả thuật toán Dijkstra

1. khởi tạo nhãn cho các đỉnh theo quy tắc: đỉnh xuất phát s được gắn nhãn bằng 0, các đỉnh kề với s được gán bằng trọng số tương ứng, các đỉnh còn lại được gán nhãn là $+\infty$
2. chọn đỉnh có nhãn nhỏ nhất (đỉnh này đã được cực tiểu hóa theo công thức 1) trong các đỉnh được gán nhãn ở bước đầu tiên, giả sử là đỉnh u
3. tiến hành gán lại nhãn và cực tiểu hóa nhãn cho các đỉnh còn lại theo u
4. thuật toán kếtc thúc khi tất các các đỉnh đã được chọn và đã được cực tiểu hoá nhãn

Đường đi ngắn nhất trong đồ thị là không duy nhất do nhiều đỉnh có thể có nhãn bé nhất bằng nhau.

Thuật toán Dijkstra

Algorithm 3: Thuật toán Dijkstra

Input : Ma trận kề đồ thị $G = (V, E)$ và đỉnh xuất phát s

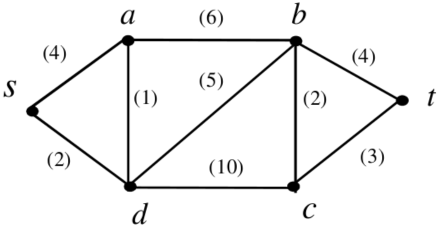
Output: Đường đi ngắn nhất từ s đến các đỉnh còn lại

```
Function Dijkstra
    for v ∈ V do                                  // Khởi tạo nhãn cho các đỉnh
        d[v] := a[s, v]
        truoc[v] := s                             // đỉnh trước v trên đường đi là s
    end
    d[s] := 0
    H := V \ {s}                                  // H-tập hợp chứa các đỉnh có nhãn tạm thời
    while H ≠ ∅ do
        d[u] := arg min{d[z] : z ∈ H}             // Chọn u là đỉnh có nhãn bé nhất
        H := H \ {u}
        for v ∈ V do                              // Cực tiểu hóa nhãn đỉnh v theo u trong H
            if d[v] > d[u] + a[u, v] then
                d[v] := d[u] + a[u, v]
                truoc[v] := u                     // Vết của đường đi ngắn nhất
            end
        end
    end
end
```

Thuật toán Dijkstra

- thuật toán Dijkstra sử dụng $n - 1$ lần cực tiểu hóa nhãn
- mỗi lần cực tiểu hóa nhãn mất nhiều nhất n phép toán so sánh nhãn
- độ phức tạp thời gian tính toán của thuật toán Dijkstra là $O(n^2)$
- thuật toán Dijkstra cũng có thể áp dụng cho đồ thị có hướng, chỉ cần lưu ý đến hướng của cung

Thuật toán Dijkstra



Hình 3

Ví dụ 2.2

Tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại của đồ thị trên hình 3 bằng thuật toán Dijkstra.

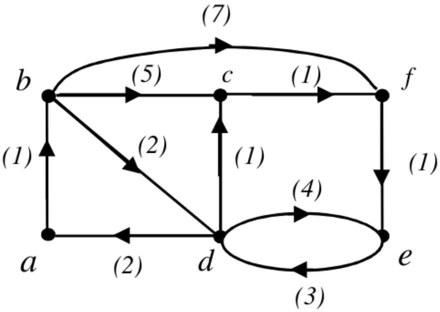
Thuật toán Dijkstra

Bảng 2: Đường đi ngắn nhất trong ví dụ 2.2

Bước lặp	s	a	b	c	d	t
1	$^*s,0$	$s,4$	s,∞	s,∞	$^*s,2$	s,∞
2	-	$^*d,3$	$d,7$	$d,12$	-	s,∞
3		-	$^*d,7$	$d,12$		s,∞
4			-	$^*b,9$		$b,11$
5						$^*b,11$

Mỗi ô trong bảng gồm hai thông số, thông số đầu chỉ tên đỉnh trung gian của đường đi từ s đến đỉnh tương ứng của ô, thông số sau là nhãn của đỉnh đó.

Thuật toán Dijkstra



Hình 4

Ví dụ 2.3

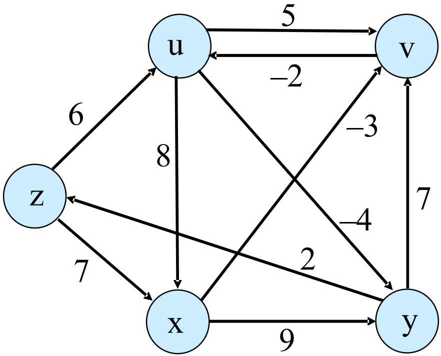
Xác định cây đường đi cho đồ thị có hướng G với trọng số các cung như trên hình 4.

Thuật toán Dijkstra

Bảng 3: Đường đi ngắn nhất từ đỉnh a đến tất cả các đỉnh còn lại của đồ thị G trong ví dụ 2.3.

Bước lặp	a	b	c	d	e	f
1	$^*a,0$	$^*a,1$	a,∞	a,∞	a,∞	a,∞
2	-	-	$b,6$	$^*b,3$	a,∞	$b,8$
3			$^*d,4$	-	$d,7$	$b,8$
4			-		$d,7$	$^*c,5$
5					$^*f,6$	-

Thuật toán Dijkstra



Ví dụ 2.4

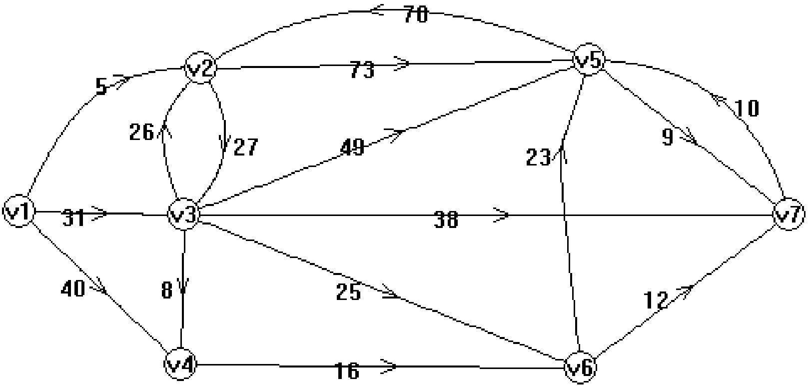
Tìm đường đi ngắn nhất từ z đến các đỉnh còn lại bằng thuật toán Dijkstra

Thuật toán Dijkstra

Ví dụ 2.5 (biểu diễn thực tế)

Cho đồ thị có trọng số $G = (V, E)$, $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ *xác định bởi ma trận trọng số* D*. Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ* v_1 *đến các đỉnh* $v_2, v_3, v_4, v_5, v_6, v_7$.

$$D = \begin{bmatrix} 0 & 5 & 31 & 40 & \infty & \infty & \infty \\ \infty & 0 & 27 & \infty & 73 & \infty & \infty \\ \infty & 26 & 0 & 8 & 49 & 25 & 38 \\ \infty & \infty & \infty & 0 & \infty & 16 & \infty \\ \infty & 70 & \infty & \infty & 0 & \infty & 9 \\ \infty & \infty & \infty & \infty & 23 & 0 & 12 \\ \infty & \infty & \infty & \infty & 10 & \infty & 0 \end{bmatrix}$$



Hình 5: Đồ thị minh họa cho ma trận kề trong ví dụ 2.5.

Thuật toán Floyd

So sánh với thuật toán Dijkstra

- Dijkstra tìm đường đi ngắn nhất từ một đỉnh đến các đỉnh còn lại.
- Floyd tìm đường đi ngắn nhất *giữa các cặp đỉnh của đồ thị* và có thể *áp dụng cho cả đồ thị có hướng không chứa chu trình và trọng số cạnh có thể âm.*
- Thuật toán Floyd có độ phức tạp thời gian tính toán $O(n^3)$

Thuật toán Floyd

Mô tả thuật toán Floyd

Xét đồ thị vô hướng có trọng số $G = (V, E)$, $|V| = n$ và $A = [a_{ij}], (i, j = \overline{1, n})$ là ma trận kề trọng số của G. Từ ma trận A, thuật toán Floyd cho *kết quả là hai ma trận*:

- Ma trận $D = [d_{ij}], (i, j = \overline{1, n})$, trong đó d_{ij} là đường đi ngắn nhất từ i đến j. Đây là ma trận đối xứng.
- Ma trận $P = [p_{ij}], (i, j = \overline{1, n})$ cho biết vết của đường đi ngắn nhất, trong đó p_{ij} *ghi nhận đỉnh đi trước đỉnh* j trong đường đi ngắn nhất từ i đến j.

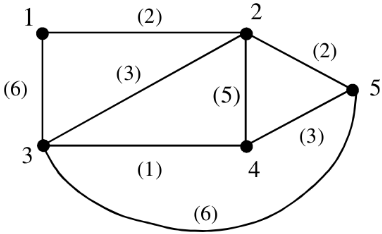
Thuật toán Floyd

Algorithm 4: Thuật toán Floyd

Input : Ma trận kề $A = [a_{ij}]$
Output: Ma trận $D = [d_{ij}]; P = [p_{ij}]$

Function Floyd
    for $i \leftarrow 1$ **to** n **do**
        for $j \leftarrow 1$ **to** n **do** `// Khởi tạo ma trận D, P`
            $d[i,j] := a[i,j]$
            $p[i,j] := i$
        end
    end
    for $k \leftarrow 1$ **to** n **do**
        for $i \leftarrow 1$ **to** n **do**
            for $j \leftarrow 1$ **to** n **do**
                if $(d[i,j] > d[i,k] + d[k,j])$ **then**
                    $d[i,j] := d[i,k] + d[k,j]$ `// Cực tiểu hóa nhãn`
                    $p[i,j] := p[k,j]$ `// vết của đường đi`
                end
            end
        end
    end
end

Thuật toán Floyd



$$A = \begin{bmatrix} 0 & 2 & 6 & \infty & \infty \\ 2 & 0 & 3 & 5 & 2 \\ 6 & 3 & 0 & 1 & 6 \\ \infty & 5 & 1 & 0 & 3 \\ \infty & 2 & 6 & 3 & 0 \end{bmatrix}$$

Hình 6: Đồ thị G

Ví dụ 2.6

Áp dụng thuật toán Floyd tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị vô hướng có trọng số G và ma trận kề A tương ứng.

Thuật toán Floyd

$$D = \begin{bmatrix} 0 & 2 & 5 & 6 & 4 \\ 2 & 0 & 3 & 4 & 2 \\ 5 & 3 & 0 & 1 & 4 \\ 6 & 4 & 1 & 0 & 3 \\ 4 & 2 & 4 & 3 & 0 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 1 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 & 2 \\ 2 & 3 & 3 & 3 & 4 \\ 2 & 3 & 4 & 4 & 4 \\ 2 & 5 & 4 & 5 & 5 \end{bmatrix}$$

Hình 7: Kết quả thực hiện của thuật toán Floyd cho đồ thị G.

Gợi ý ví dụ 2.6

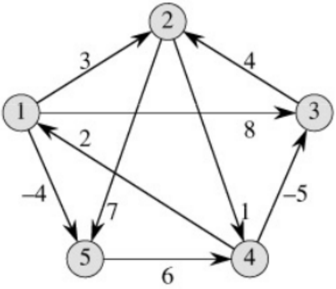
Giả sử cần xác định đường đi ngắn nhất từ đỉnh **1** đến đỉnh **4**:

- $d_{14} = 6 \Rightarrow$ độ dài của đường đi ngắn nhất từ đỉnh **1** đến đỉnh **4** là 6
- $p_{14} = 3; p_{13} = 2; p_{12} = 1 \Rightarrow$ vết của đường đi ngắn nhất từ đỉnh **1** đến đỉnh **4** là $1 \to 2 \to 3 \to 4$.

Thuật toán Floyd

Ví dụ 2.7

Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh trong đồ thị sau bằng thuật toán Floyd.



Tài liệu tham khảo

- Đ.N. An
  Giáo Trình Toán Rời Rạc. *Trường ĐH Nha Trang, (2021).*
- Giáo trình Toán rời rạc
  Giáo trình Toán Rời Rạc. *Trường ĐHSP Huế. (2003), 22-35.*
- N.T. Nhựt
  Bài giảng Toán Rời Rạc. *Trường ĐH KHTN Tp.HCM. (2011).*