# Bài tập về nhà - Thuật toán Phân tán

Thành viên:

1. Hà Xuân Thiện - 24520031
2. Trần Quang Trường - 24521901

Ngày 30 tháng 10 năm 2025

## Mục lục

# 1 Subtask 1 ($n, q \leq 1000$)

## 1.1 Mô tả Thuật Toán

**Quan sát 1: Ta có thể biến đổi bài toán thành bài toán đồ thị.**

Cụ thể: Ta coi mỗi máy tính là một đỉnh của đồ thị. Nếu có thể truyền dữ liệu trực tiếp giữa hai máy tính $u$ và $v$ thì tồn tại một cạnh $(u, v)$ có trọng số $|u - v|$. Lúc này, bài toán trở thành tìm kiếm đường đi có tổng trọng số nhỏ nhất (đường đi tối ưu) đi từ $s$ đến $t$.

**Định nghĩa:**

- Độ dài đường đi là số lượng cạnh đi qua của đường đi đó.
- Gọi $E$ là tập cạnh của đồ thị. Nếu cạnh $(u, v)$ thuộc đồ thị ta nói $(u, v) \in E$, ngược lại $(u, v) \notin E$.

**Nhận xét 1: Nếu $s \neq t$ và $(s, t) \in E$ thì độ dài đường đi tối ưu bằng 1.**

*Chứng minh:*

1. Xét đường đi trực tiếp đi từ $s$ đến $t$, chi phí bằng $|s - t|$.
2. Xét đường đi thông qua một đỉnh trung gian $u$ ($(s, u)$, $(u, t) \in E$), thì chi phí bằng $|s - u| + |u - t|$.

   Đặt $a = s - u$, $b = u - t$. Theo bất đẳng thức tam giác:

   $$|a + b| \leq |a| + |b| \Rightarrow |s - u + u - t| \leq |s - u| + |u - t| \Rightarrow |s - t| \leq |s - u| + |u - t|$$

3. Xét đường đi thông qua hai đỉnh trung gian $u, v$ ($(s, u)$, $(u, v)$, $(v, t) \in E$), chi phí bằng $|s - u| + |u - v| + |v - t|$. Theo bất đẳng thức tam giác:

   $$|s - t| \leq |s - u| + |u - v| + |v - t|$$

4. Tương tự, với mọi đường đi qua $k$ đỉnh trung gian $u_1, \ldots, u_k$:

   $$|s - t| \leq |s - u_1| + \sum_{i=1}^{k-1} |u_i - u_{i+1}| + |u_k - t|$$

   **Kết luận:** đường đi tối ưu là đường đi đi trực tiếp từ $s$ đến $t$ với chi phí $|s - t|$. Độ dài đường đi tối ưu bằng 1.

**Nhận xét 2: Độ dài đường đi tối ưu đi từ $s$ đến $t$ không quá 2.**

*Chứng minh:*

**Trường hợp 1.** Nếu $s = t$: Đường đi tối ưu có chi phí $|s - t| = 0$. Độ dài đường đi tối ưu bằng 0.

**Trường hợp 2.** Nếu $s \neq t$ và $(s, t) \in E$: Theo nhận xét 1, độ dài đường đi tối ưu bằng 1.

**Trường hợp 3.** Nếu $s \neq t$ và $(s, t) \notin E$:

- Gọi tập giao thức là $\mathcal{P} = \{B, G, R, Y\}$.
- Mỗi đỉnh chứa một tập giao thức $|\mathcal{P}_i| = 2$, $\mathcal{P}_i \in \mathcal{P}$.
- Nếu $(u, v) \in E$ thì $\mathcal{P}_u \cap \mathcal{P}_v \neq \varnothing$, ngược lại $\mathcal{P}_u \cap \mathcal{P}_v = \varnothing$.
- Từ nhận xét 1, ta có chứng minh đường đi tối ưu luôn có dạng:

$$\{u_1, u_2, \ldots, u_k\}$$

  với $k > 2, s = u_1, t = u_k$ và $|\mathcal{P}_i \cap \mathcal{P}_{i+1}| = 1, \forall i < k$ - thỏa mãn điều kiện:

  - Với mọi $i < k, j \in [i+2, k]$ thì $(u_i, u_j) \notin E$, hay $\mathcal{P}_i \cap \mathcal{P}_j = \varnothing$.

  Xét đường đi tối ưu $\{s = u_1, u_2, u_3 = t\}$:

  - Để tồn tại, cần có $u_2$ sao cho:

$$|\mathcal{P}_s \cap \mathcal{P}_{u_2}| = 1, \quad |\mathcal{P}_{u_2} \cap \mathcal{P}_t| = 1$$

  - Suy ra: $\mathcal{P}_{u_2} = \{a, b\}$ với $a \in \mathcal{P}_s$, $b \in \mathcal{P}_t$.

  $\Rightarrow$ Do đó, tồn tại đường đi tối ưu có độ dài bằng 2 **khi và chỉ khi** tồn tại $u_2 = \{a, b\}$ sao cho $a \in \mathcal{P}_s, b \in \mathcal{P}_t$.

  Xét đường đi $\{s = u_1, u_2, u_3, u_4 = t\}$:

  - Để tồn tại đường đi thỏa mãn, cần tồn tại $u_2$, $u_3$ sao cho:

$$\begin{aligned}
&|\mathcal{P}_s \cap \mathcal{P}_{u_2}| = 1,\\
&|\mathcal{P}_{u_2} \cap \mathcal{P}_{u_3}| = 1,\\
&|\mathcal{P}_{u_3} \cap \mathcal{P}_t| = 1,\\
&\mathcal{P}_s \cap \mathcal{P}_{u_3} = \varnothing,\\
&\mathcal{P}_s \cap \mathcal{P}_t = \varnothing,\\
&\mathcal{P}_{u_2} \cap \mathcal{P}_t = \varnothing.
\end{aligned}$$

  - Xét $\mathcal{P}_s = \{a, b\}$.
    $\Rightarrow \mathcal{P}_{u_3} = \{c, d\}$ (do $\mathcal{P}_s \cap \mathcal{P}_{u_3} = \varnothing$).
    $\Rightarrow \mathcal{P}_t = \{c, d\}$ (do $\mathcal{P}_s \cap \mathcal{P}_t = \varnothing$).
    $\Rightarrow$ $\mathcal{P}_{u_3} \cup \mathcal{P}_t = \{c, d\}$, hay $|\mathcal{P}_{u_3} \cup \mathcal{P}_t| = 2$, mâu thuẫn với điều kiện $|\mathcal{P}_{u_3} \cap \mathcal{P}_t| = 1$.
    $\Rightarrow$ Do đó, không tồn tại đường đi tối ưu độ dài bằng 3 thỏa mãn điều kiện.

  Chứng minh tương tự, không tồn tại đường đi tối ưu có mọi độ dài lớn hơn 3.

  $\Rightarrow$ Do đó, nếu tồn tại đường đi từ $s$ đến $t$, thì độ dài tối ưu $= 2$.

## 1.2 Cài Đặt Thuật Toán

Ta biểu diễn $\mathcal{P}_u$ dưới dạng một mask 4 bit $ms_u$.

Nếu $\mathcal{P}_u \cap \mathcal{P}_v = \varnothing$ thì $ms_u \& ms_v = 0$. Ngược lại, $ms_u \& ms_v \neq 0$.

```python
code_str = "BGRY"
idx = {c: 1 << i for i, c in enumerate(code_str)}
def get_idx(c):
    return idx[c]
ms = [get_idx(s[i][0]) | get_idx(s[i][1]) for i in range(n)]
```

Với mỗi truy vấn $(u, v)$:

- Nếu $u = v$: đáp án $= 0$.
- Nếu $ms_u \& ms_v \neq 0$: đáp án $= |u - v|$.
- Ngược lại: đáp án $= \min_k |k-u|+|k-v|$ với điều kiện $ms_u \& ms_k \neq 0$ và $ms_v \& ms_k \neq 0$. Nếu không tồn tại $k$ thỏa: in $-1$.

```python
INF = 1000000000
for i in range(q):
    u, v = map(int, input().split())
    u = u - 1
    v = v - 1
    ans = INF
    if u == v:
        ans = 0
    elif ms[u] & ms[v]:
        ans = abs(u-v)
    else:
        for k in range(n):
            if ms[k] & ms[u] and ms[k] & ms[v]:
                ans = min(ans, abs(u - k), abs(k - v))
    print(ans)
```

## 1.3 Phân Tích Độ Phức Tạp

### 1.3.1 Độ phức tạp thời gian.

Ở bước **tiền xử lý**, ta cần xây dựng mảng $ms_u$ với số phép thao tác là $2 \cdot n$. Do đó, độ phức tạp là $\Theta(n)$.

Ở mỗi truy vấn, hai trường hợp đầu chỉ mất một thao tác thực hiện, nhưng với trường hợp thứ ba, ta mất $n$ thao tác để duyệt mọi phần tử trong mảng, do đó mất độ phức tạp $\Theta(n)$.

Tổng độ phức tạp **xử lý các truy vấn** là $\Theta(q \cdot n)$.

Do đó, **độ phức tạp thời gian** của bài toán là

$$\langle \Theta(n), \Theta(q \cdot n) \rangle \quad \text{hay} \quad \Theta(n + q \cdot n).$$

### 1.3.2 Độ phức tạp không gian.

Ở bước **tiền xử lý**, ta cần tạo một mảng $ms_u$, do đó mất độ phức tạp không gian là $\Theta(n)$.

Ở bước **xử lý truy vấn**, ta không tạo thêm gì, nên độ phức tạp không gian là $\Theta(1)$.

Vì vậy, **độ phức tạp không gian** của bài toán là

$$\langle \Theta(n), \Theta(1) \rangle \quad \text{hay} \quad \Theta(n).$$

# 2 Subtask 2 ($n, q \leq 10^5$)

## 2.1 Mô tả Thuật Toán

Không mất tính tổng quát, giả sử $s \leq t$.

**Nhận xét 4:** Nếu $(s, t) \notin E$ và tồn tại đường đi tối ưu, thì đường đi tối ưu $\{s, u, t\}$ có $u$ là đỉnh gần $s$ nhất hoặc gần $t$ nhất về bên trái hoặc phải.

Gọi:

$$L[x] = \max\{i < s \mid |\mathcal{P}_i \cap \mathcal{P}_s| = 1\}, \quad R[x] = \min\{i > s \mid |\mathcal{P}_i \cap \mathcal{P}_s| = 1\}$$

Khi đó $u \in \{L[s], R[s], L[t], R[t]\}$.

*Chứng minh:*

Xét 3 trường hợp của $u$:

- **Nếu $u \leq s$ thì:**

  $$s - L[s] \leq s - i \quad \forall i \leq s,\ |\mathcal{P}_i \cap \mathcal{P}_s| = 1.$$

  Do đó,

  $$\Rightarrow u = L[s].$$

- **Nếu $s \leq u \leq t$ thì:**

  Mọi $u$ thỏa mãn đều tạo ra đường đi tối ưu.

  $$\Rightarrow u = R[s] \text{ hoặc } u = L[t] \text{ đều thỏa mãn.}$$

- **Nếu $t \leq u$ thì:**

  $$R[t] - t \leq i - t \quad \forall t \leq i,\ |\mathcal{P}_t \cap \mathcal{P}_i| = 1.$$

  Do đó,

  $$\Rightarrow u = R[t].$$

Suy ra,

$$u \in \{L[s],\ R[s],\ L[t],\ R[t]\}.$$

## 2.2 Cài Đặt Thuật Toán

Để xây dựng $L[x], R[x]$, ta duy trì mảng $last[4]$ lưu hai vị trí cuối cùng có bit thứ $b$ bật và mask của hai vị trí là khác nhau.

Ban đầu $L[x] = -1, R[x] = n, \forall x$.

```
last = [(-1, -1) for i in range(4)]
L = [-1 for i in range(n)]
R = [n for i in range(n)]
```

Tại vị trí $i$, nếu bit $b$ bật thì:

$$p = last[b] \text{ sao cho } ms_p \neq ms_i$$

Sau đó:

$$L[i] = \max(L[i], p), \quad R[p] = \min(R[p], i)$$

và cập nhật $i$ vào vị trí cuối cùng có bit thứ $b$ hay $last[b]$.

```python
for i in range(n):
    for b in range(4):
        if ms[i] >> b & 1:
            p = last[b][last[b][0] != -1
                        and ms[i] == ms[last[b][0]]]
            if p != -1:
                L[i] = max(L[i], p)
                R[p] = min(R[p], i)

            if last[b][0] == -1 or ms[last[b][0]] == ms[i]:
                last[b] = (i, last[b][1])
            else:
                last[b] = (i, last[b][0])
```

Khi đã có $L[x], R[x]$, mỗi truy vấn:

- Nếu $s = t$ hoặc $(s, t) \in E$: xử lý như Subtask 1.
- Ngược lại: duyệt $u \in \{L[s], R[s], L[t], R[t]\}$ và lấy min $|u - s| + |u - t|$ nếu thỏa mãn điều kiện $ms_u \& ms_s \neq 0$ và $ms_u \& ms_t \neq 0$. Nếu không tìm được $u$ thỏa mãn, in -1.

```python
INF = 1000000000
for i in range(q):
    u, v = map(int, input().split())
    u = u - 1
    v = v - 1
    ans = -1
    if ms[u] & ms[v]:
        ans = abs(u-v)
    else:
        ans = INF
        if L[u] != -1:
            ans = min(ans, abs(u - L[u]) + abs(L[u] - v))
        if R[u] != n:
            ans = min(ans, abs(u - R[u]) + abs(R[u] - v))
        if L[v] != -1:
            ans = min(ans, abs(v - L[v]) + abs(L[v] - u))
        if R[v] != n:
            ans = min(ans, abs(v - R[v]) + abs(R[v] - u))
        if ans == INF:
            ans = -1
    print(ans)
```

## 2.3 Phân Tích Độ Phức Tạp

### 2.3.1 Độ phức tạp thời gian.

**Ở bước tiền xử lý:**

- Đầu tiên, ta cần xây dựng mảng $ms_u$ với số phép thao tác là $2 \cdot n$.
- Tiếp theo, để khởi tạo mảng $L[x]$ và $R[x]$, ta duyệt qua $n$ phần tử. Với mỗi phần tử, ta kiểm tra tại 2 bit bật (số phép thao tác là 4), và tại mỗi bit bật ta kiểm tra tối đa 2 vị trí trong $last[b]$ để tìm $p$ thỏa mãn (số thao tác tại mỗi bit bật là 2).

Do đó, tổng số phép thao tác để khởi tạo mảng $L[x], R[x]$ là

$$(4 + 2 \cdot 2) \times n = 8n.$$

Suy ra, tổng độ phức tạp thời gian để tiền xử lý là $\Theta(n)$.

**Ở bước xử lý truy vấn:**

- Với mỗi truy vấn, hai trường hợp đầu chỉ mất 1 thao tác thực hiện.
- Với trường hợp thứ ba, ta chỉ duyệt 4 phần tử và kiểm tra điều kiện thỏa mãn.

Do đó, số phép thao tác là 4 cho mỗi truy vấn, hay độ phức tạp thời gian để thực hiện mỗi truy vấn là $\Theta(1)$.

Suy ra, độ phức tạp để thực hiện toàn bộ $q$ truy vấn là $\Theta(q)$.

**Độ phức tạp thời gian của bài toán là**

$$\langle \Theta(n), \Theta(q) \rangle \text{ hay } \Theta(n + q).$$

### 2.3.2 Độ phức tạp không gian.

**Ở bước tiền xử lý:**

- Ta cần tạo mảng $ms_u$, $L[x]$, $R[x]$ nên mất độ phức tạp không gian là $\Theta(n)$.
- Mảng $last[4]$ với mỗi phần tử lưu 2 vị trí nên mất độ phức tạp không gian là $\Theta(1)$.

Do đó, độ phức tạp không gian ở bước tiền xử lý là $\Theta(n)$.

**Ở bước xử lý truy vấn:** Ta không tạo thêm gì nên độ phức tạp không gian là $\Theta(1)$.

**Độ phức tạp không gian của bài toán là**

$$\langle \Theta(n), \Theta(1) \rangle \text{ hay } \Theta(n).$$

# Lưu ý

Phần này note một số chỗ mình chưa giải thích trong phần mục đích.

1. **Tại sao lại là $|\mathcal{P}_i \cap \mathcal{P}_{i+1}| = 1, \forall i < k$ mà không phải là $\mathcal{P}_i \cap \mathcal{P}_{i+1} \neq \varnothing, \forall i < k$**

   - Bởi vì nếu chỉ cần điều kiện kia thì độ dài đường đi tối ưu không nhất thiết bằng 2.

     $\{s, u_2, u_3, t\}$ vẫn thỏa mãn nếu chỉ cần với điều kiện $\mathcal{P}_i \cap \mathcal{P}_{i+1} \neq \varnothing, \forall i < k$.
   - Một cách chứng minh khác. Mục đích ở đây là tìm $k$ nhỏ nhất mà vẫn thỏa mãn đường đi tối ưu.

     Ta ưu tiên xét $\{s, u_3, t\}$ và bỏ qua $\{s, u_2, u_3, t\}$ nếu $|\mathcal{P}_s \cap \mathcal{P}_{u_2}| = 2$ vì lúc này $s$ có thể đến $u_3$ vẫn đạt tối ưu mà không cần qua $u_2$ (Theo nhận xét 1).

2. **Tại sao phải lưu 2 vị trí cuối cùng có bit $b$ bật thay vì một vị trí**

   - Tại vị trí $i$ bất kì, ta cần tìm $p = last[b]$ sao cho $ms_p \neq ms_i$ nếu bit $b$ bật.

     Nếu như chỉ lưu 1 vị trí cuối cùng, thì $p$ có thể rơi vào trường hợp $ms_p = ms_i$.

     Do đó, ta cần lưu 2 vị trí để đảm bảo có thể tồn tại một vị trí trong đó $ms_p \neq ms_i$.