Disjoint Set Union (DSU) – Hợp Tập Hợp Rời Rạc

Nguyễn Quản Bá Hồng*

Ngày 22 tháng 8 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Disjoint Set Union (DSU) – Hợp Tập Hợp Rời Rạc.*
  PDF: URL: .pdf.
  TEX: URL: .tex.
- .
  PDF: URL: .pdf.
  TEX: URL: .tex.

Mục lục

1 Introduction to Disjoint Set Union – Nhập Môn Hợp Tập Hợp Rời Rạc 1
1.1 Data Structure Disjoint Set Union – Cấu trúc dữ liệu Disjoint Set Union 2

2 Miscellaneous 2

1 Introduction to Disjoint Set Union – Nhập Môn Hợp Tập Hợp Rời Rạc

Resources – Tài nguyên.

1. Algorithms for Competitive Programming/disjoint set union.
2. Benjamin Qi, Andrew Wang, Nathan Gong, Michael Cao. USACO Guide/Disjoint Set Union.

   Abstract. The Disjoint Set Union (DSU) data structure, which allows you to add edges to a graph & test whether 2 vertices of a graph are connected.

   – Cấu trúc dữ liệu Disjoint Set Union (DSU), cho phép bạn thêm các cạnh vào đồ thị & kiểm tra xem 2 đỉnh của đồ thị có được kết nối hay không.
3. Ngô Quang Nhật. VNOI Wiki/Disjoint Set Union.
4. Wikipedia/disjoint-set data structure.

Disjoint Set Union (abbr., DSU) là 1 cấu trúc dữ liệu hữu dụng, thường xuất hiện trong các kỳ thi Lập trình Thi Đấu, & có thể được dùng để quản lý 1 cách hiệu 1 tập hợp của các tập hợp.

Bài toán 1. *Cho 1 đồ thị $G = (V, E)$ có $|V| = n \in \mathbb{N}^\star$ đỉnh, ban đầu không có cạnh nào, $E = \emptyset$. Ta cần xử lý 2 loại truy vấn:*

1. *Thêm 1 cạnh giữa 2 đỉnh $x, y \in V$ trong đồ thị, i.e., $E = E \cup \{\{x, y\}\}$ nếu G là đồ thị vô hướng & $E = E \cup \{(x, y)\}$ nếu G là đồ thị có hướng.*
2. *In ra `yes` nếu như 2 đỉnh x, y nằm trong cùng 1 thành phần liên thông. In ra `no` nếu ngược lại.*

*A scientist- & creative artist wannabe, a mathematics & computer science lecturer of Department of Artificial Intelligence & Data Science (AIDS), School of Technology (SOT), UMT Trường Đại học Quản lý & Công nghệ TP.HCM, Hồ Chí Minh City, Việt Nam.
E-mail: nguyenquanbahong@gmail.com & hong.nguyenquanba@umt.edu.vn. Website: https://nqbh.github.io/. GitHub: https://github.com/NQBH.

1.1 Data Structure Disjoint Set Union – Cấu trúc dữ liệu Disjoint Set Union

Cho tiện, ta đánh số n đỉnh của đồ thị G bởi $1, 2, \ldots, n$ (trường hợp n đỉnh được dán nhãn bởi $v_1, v_2, \ldots, v_n$ hoàn toàn tương tự vì ta có thể làm việc trên chỉ số i của v_i), khi đó $V = [n]$. Giả sử G có $c :=$ `num_connected_component` $\in \mathbb{N}^\star$ (số thành phần liên thông) $C_1, C_2, \ldots, C_c$ với $\{C_i\}_{i=1}^c$ là 1 phân hoạch của $V = [n]$, i.e.:

$$\bigcup_{i=1}^{c} C_i = [n],\ C_i \cap C_j = \emptyset,\ \forall i, j \in [c],\ i \neq j.$$

Nếu ta coi mỗi đỉnh của đồ thị $G = (V, E)$ là 1 phần tử & mỗi thành phần liên thông (connected component) trong đồ thị là 1 tập hợp, truy vấn 1 sẽ trở thành gộp 2 tập hợp lần lượt chứa phần tử x, y thành 1 tập hợp mới & truy vấn 2 trở thành việc hỏi 2 phần tử x, y có nằm trong cùng 1 tập hợp không.

Để tiện tính toán & lý luận về mặt toán học cho riêng cấu trúc dữ liệu DSU, sau đây là 1 định nghĩa lai Toán–Tin mang tính cá nhân của tác giả [NQBH], hoàn toàn không chính thống trong Lý thuyết Đồ thị:

Định nghĩa 1 (Chỉ số thành phần liên thông)**.** *Cho đồ thị vô hướng $G = (V, E)$ với $V = [n]$. Gọi $C(i) \subset [n]$ là thành phần liên thông của $G = (V, E)$ chứa đỉnh $i \in [n]$ & gọi chỉ số của thành phần liên thông chứa đỉnh i là* $\mathrm{cid}(i)$*, i.e., $i \in C_{\mathrm{cid}(i)} \equiv C(i)$, với hàm* $\mathrm{cid} : [n] \to [c]$ *được gọi là* hàm chỉ số liên thông.

Với định nghĩa 1, ta có ngay

$$\begin{cases} i \in C(i) = C_{\mathrm{cid}(i)} \subset [n], \\ i, j \text{ are connected},\ \forall j \in C(i). \end{cases} \forall i \in [n].$$

Ở đây, ta coi mỗi đỉnh của đồ thị tự liên thông với chính nó theo nghĩa đỉnh đó đến được (reachability) chính đỉnh đó thông qua 1 đường đi có độ dài 0, được gọi là 1 *đường đi tầm thường*, chứ không phải theo nghĩa khuyên (loop).

Lemma 1 (A characterization of connectedness – 1 đặc trưng hóa của tính liên thông)**.** *Cho đồ thị vô hướng $G = (V, E)$. (i) 2 đỉnh trên 1 đồ thị G không liên thông với nhau, i.e., không có đường đi nào trên G nối chúng khi & chỉ khi chúng thuộc 2 thành phần liên thông khác nhau, i.e.,*

$$i, j \textit{ are not connected} \Leftrightarrow C(i) \neq C(j) \Leftrightarrow C(i) \cap C(j) = \emptyset \Leftrightarrow \mathrm{cid}(i) \neq \mathrm{cid}(j),\ \forall i, j \in [n].$$

(ii) 2 đỉnh trên đồ thị G liên thông với nhau, i.e., có đường đi trên trên G nối chúng khi & chỉ khi chúng cùng thuộc 1 thành phần liên thông, i.e.:

$$i, j \textit{ are connected} \Leftrightarrow C(i) = C(j) \Leftrightarrow C(i) \cap C(j) \neq \emptyset \Leftrightarrow \mathrm{cid}(i) = \mathrm{cid}(j),\ \forall i, j \in [n].$$

Với truy vấn 1, giả sử 2 đỉnh $i, j \in [n]$ chưa có cạnh nối chúng trực tiếp, i.e., $\{i, j\} \notin E$. Có 2 trường hợp xảy ra:

- Trường hợp 1: Giả sử i, j cùng thuộc 1 thành phần liên thông, theo Lem. 1, có $C(i) = C(j)$, $\mathrm{cid}(i) = \mathrm{cid}(j)$ nên ta chỉ cần thêm cạnh $\{i, j\}$ vào tập cạnh E: $E \leftarrow E \cup \{\{i, j\}\}$ hay `edge_list.append({i, j})` mà không cần cập nhật c tập liên thông $\{C_i\}_{i=1}^c$ hay hàm chỉ số liên thông $\mathrm{cid}(\cdot)$.
- Trường hợp 2: Giả sử i, j thuộc 2 thành phần liên thông khác nhau, theo 1, có $C(i) \neq C(j)$, $\mathrm{cid}(i) \neq \mathrm{cid}(j)$

2 Miscellaneous