圖論

Koying

2024-08-15

目錄

- 前言
- 圖的名詞
- 圖的儲存
- 圖的遍歷
- 二分圖
- 拓撲排序
- DAG 上的 DP

前言

前言

- 相信大家都或多或少聽過圖論這個名詞，但可能不知道這實際上是什麼
- 光聽圖論可能會覺得它是一個很複雜的理論，很難親近
- 但其實圖論沒有那麼複雜，甚至可以套用到很多生活中的例子
- 這堂課會帶大家認識什麼是圖論、圖論有甚麼內容、以及圖論的基本概念與實作

前言

- 此外，圖論的許多概念會跳脫大家以往使用陣列的儲存方式
- 對於第一次接觸的人可能會覺得很難想像
- 簡易大家可以準備紙筆，將圖畫在紙上可能會比較方便思考

什麼是圖論？

- 在數學層面，圖論是組合數學的分支，與群論、矩陣論、拓撲學相關

什麼是圖論？

- 在數學層面，圖論是組合數學的分支，與群論、矩陣論、拓撲學相關
- 在資訊層面，圖論是指**圖的理論**，並不是指圖學的圖片

什麼是圖論？

- 在數學層面，圖論是組合數學的分支，與群論、矩陣論、拓撲學相關
- 在資訊層面，圖論是指**圖的理論**，並不是指圖學的圖片
- 是更接近於關係圖的概念

什麼是圖論？

- 在數學層面，圖論是組合數學的分支，與群論、矩陣論、拓撲學相關
- 在資訊層面，圖論是指**圖的理論**，並不是指圖學的圖片
- 是更接近於關係圖的概念
- 起源於著名的柯尼斯堡七橋問題

什麼是圖論？

- 在數學層面，圖論是組合數學的分支，與群論、矩陣論、拓撲學相關
- 在資訊層面，圖論是指**圖的理論**，並不是指圖學的圖片
- 是更接近於關係圖的概念
- 起源於著名的柯尼斯堡七橋問題
- 歐拉被認為是圖論的創始人

圖論的基本概念

一張圖的組成

- 圖論由點 (Vertex, V) 與邊 (Edge, E) 組成

一張圖的組成

- 圖論由點 (Vertex, V) 與邊 (Edge, E) 組成
- 兩點之間會由邊互相連接

一張圖的組成

- 圖論由點 (Vertex, V) 與邊 (Edge, E) 組成
- 兩點之間會由邊互相連接
- 由許多點以及邊所組成的集合 G 就是一張圖，可以表示為 $\mathrm{G} = (\mathrm{V}, \mathrm{E})$

一張圖可以用來表示甚麼？

- 在圖論中，一張圖可以表示很多東西

一張圖可以用來表示甚麼？

- 在圖論中，一張圖可以表示很多東西
- 點經常代表著一個人、物品或是城市
- 而邊經常代表人事物之間的關係，或是兩座城市之間的道路

一張圖可以用來表示甚麼？

- 在圖論中，一張圖可以表示很多東西
- 點經常代表著一個人、物品或是城市
- 而邊經常代表人事物之間的關係，或是兩座城市之間的道路
- 只要是有關係的人事物，都可以用圖論表示出來

一張圖可以用來表示甚麼？

- 在圖論中，一張圖可以表示很多東西
- 點經常代表著一個人、物品或是城市
- 而邊經常代表人事物之間的關係，或是兩座城市之間的道路
- 只要是有關係的人事物，都可以用圖論表示出來
  - 像是我被 Colten 打爆，就可以用 Colten -> Koying 的方式呈現出來

名詞解釋

與點相關的名詞

- 相鄰 (adjacent)：若有一條邊連結 u, v ，則我們稱 u, v 相鄰

與點相關的名詞

- 相鄰 (adjacent)：若有一條邊連結 u, v ，則我們稱 u, v 相鄰
- 度數 (degree)：與一個點相連接的邊數

度數

- 入度 (in-degree)：指向節點的邊數，計為 $\deg^+(\mathrm{v})$

度數

- 入度 (in-degree)：指向節點的邊數，計為 $\deg^+(\mathrm{v})$
- 出度 (out-degree)：由節點往外指的邊數，計為 $\deg_-(\mathrm{v})$

度數

- 入度 (in-degree)：指向節點的邊數，計為 $\deg^+(v)$
- 出度 (out-degree)：由節點往外指的邊數，計為 $\deg_-(v)$
- 一張圖 $G = (E, V)$，$\sum_{v \in V} \deg(v) = 2 \mid E \mid$

度數

- 入度 (in-degree)：指向節點的邊數，計為 $\deg^+(\mathrm{v})$
- 出度 (out-degree)：由節點往外指的邊數，計為 $\deg_-(\mathrm{v})$
- 一張圖 $\mathrm{G} = (\mathrm{E}, \mathrm{V})$，$\sum_{\mathrm{v} \in \mathrm{V}} \deg(\mathrm{v}) = 2 \mid \mathrm{E} \mid$
- 孤立點 (isolated vertex) ：$\deg(\mathrm{v}) = 0$
- 葉節點 (leaf vertex)：$\deg(\mathrm{v}) = 1$

度數

- 入度 (in-degree)：指向節點的邊數，計為 $\deg^+(\mathrm{v})$
- 出度 (out-degree)：由節點往外指的邊數，計為 $\deg_-(\mathrm{v})$
- 一張圖 $\mathrm{G} = (\mathrm{E}, \mathrm{V})$，$\sum_{\mathrm{v} \in \mathrm{V}} \deg(\mathrm{v}) = 2 \mid \mathrm{E} \mid$
- 孤立點 (isolated vertex) ：$\deg(\mathrm{v}) = 0$
- 葉節點 (leaf vertex)：$\deg(\mathrm{v}) = 1$
- 偶點 (even vertex)：$2 \mid \deg(\mathrm{v})$
- 奇點 (odd vertex)：$1 \mid \deg(\mathrm{v})$

與邊相關的名詞

1. 權重 (weigt)：計為 $\mathrm{w}(\mathrm{u},\mathrm{v})$，可以是邊的距離、價值、流量等等
   - 負邊 (negative edge)：$\mathrm{w}(\mathrm{u},\mathrm{v}) < 0$
2. 重複邊 (multipul edge)：兩條一樣的邊
3. 路徑 (path)：由一個點到另一個點所途經的邊
   - 簡單路徑 (simple path)：路徑上每個點只走過一次
   - 最短路徑 (shortest path)：權重和最短的路徑
4. 環、迴路 (cycle)：一條起終點相同的路徑
   - 自環 (self-cycle)：自己連到自己的環
   - 負環 (negative cycle)：權重和為負的環
5. 連通 (connected)

連通

1. 無向圖：
   - 連通：存在一條路徑能從從 a 走到 b
   - 連通分量：一個集合內的點互相連通
2. 有向圖：
   - 強連通：a, b 兩兩互相可達
   - 強連通分量：集合內的點對都兩兩互相可達
   - 弱連通：將有向邊改成無向邊會變為連通

與圖相關的名詞

1. 簡單圖：沒有自環或重邊的圖
2. 有向圖、無向圖、混和圖
3. 子圖：由一張圖內的某些點、邊組成的圖
4. 稀疏圖、稠密圖：若 $|\mathrm{E}|$ 遠小於 $|\mathrm{V}|^2$ 稱為稀疏圖，反之為稠密圖
5. 有向無環圖：沒有環的有向圖
6. 樹：n 個節點，$\mathrm{n}-1$ 條邊的有向無環圖

與樹相關的名詞

- 有根樹、無根樹：是否有指定根結點
- 適用於有根樹與無根樹的名詞：
  1. 森林 (forest)：每個連通塊都是樹的圖
  2. 生成樹 (spanning tree)：一個連通無向圖的子圖，在圖的邊集裡選 $\mathrm{n}-1$ 條邊將所有頂點連通，會滿足是一棵樹
  3. 葉節點 (leaf node)：無根樹上度數不超過 1 的點 / 有根樹上沒有子結點的點
- 適用於有根樹的名詞
  1. 父親 (parent node)：一個結點到根結點路徑上的第二個點，根結點沒有父親
  2. 祖先 (ancestor)：一個結點到根結點路徑上除了自身以外的所有結點
  3. 子結點 (child node)：若 u 是 v 的父親，則 v 是 u 的子結點
  4. 深度 (depth)：根結點到某個點路徑上的點數
  5. 高度 (height)：所有結點深度的最大值
  6. 子樹 (subtree)：由某個點以及其子孫所組成的樹

一筆畫問題

有一張無向圖，求有沒有辦法在每條邊不經過兩次的情況下，將所有點跟邊都走過一遍

- 首先對於一個圖的走訪有兩種情況：有起終點跟沒有起終點
- 不難發現，在有起終點的情況下，除了起終點以外，每個點入點的次數跟出點的次數一樣，也就是每個點的度數都會是偶數
- 而起點的入點次數會是出點次數 +1，終點的出點次數會是入點次數 +1，也就是起終點的度數會是奇數
- 所以我們可以得到一個結論：要滿足能夠一筆畫畫完的圖，其奇點的數量必為 0/2

如何表示圖

如何表示圖

- 如果要表示一個點的話，只需要用一個陣列來紀錄就好了

如何表示圖

- 如果要表示一個點的話，只需要用一個陣列來紀錄就好了
- 但是要怎麼表示邊就有點麻煩了

如何表示圖

- 如果要表示一個點的話，只需要用一個陣列來紀錄就好了
- 但是要怎麼表示邊就有點麻煩了
- 以下介紹三種常見的存圖方法

如何表示圖

- 如果要表示一個點的話，只需要用一個陣列來紀錄就好了
- 但是要怎麼表示邊就有點麻煩了
- 以下介紹三種常見的存圖方法
  1. 直接儲存
  2. 相鄰矩陣
  3. 相鄰串列
- 接下來的講義中，會以 n 表示點數，m 表示邊數

直接儲存

- 我們可以使用跟點一樣的方式，直接把點的資訊以 pair 或是 tuple 陣列之類的方式存起來

直接儲存

- 我們可以使用跟點一樣的方式，直接把點的資訊以 pair 或是 tuple 陣列之類的方式存起來
- 查詢一條邊是否存在：$\mathcal{O}(\mathrm{m})$
- 遍歷一個點的出邊：$\mathcal{O}(\mathrm{m})$
- 遍歷整張圖：$\mathcal{O}(\mathrm{nm})$
- 空間複雜度：$\mathcal{O}(\mathrm{m})$

直接儲存

- 我們可以使用跟點一樣的方式，直接把點的資訊以 pair 或是 tuple 陣列之類的方式存起來
- 查詢一條邊是否存在：$\mathcal{O}(\mathrm{m})$
- 遍歷一個點的出邊：$\mathcal{O}(\mathrm{m})$
- 遍歷整張圖：$\mathcal{O}(\mathrm{nm})$
- 空間複雜度：$\mathcal{O}(\mathrm{m})$
- 可以發現到這種方法最大的缺點在遍歷整張圖時的複雜度，所以不常出現在需要遍歷的情況中
- 參考程式：build_graph_array.cpp

相鄰矩陣

- 這種方法會使用一個二維的表格，以 $G[i][j]$ 代表點 (i, j) 之間是否有邊連接

相鄰矩陣

- 這種方法會使用一個二維的表格，以 $\mathrm{G[i][j]}$ 代表點 (i, j) 之間是否有邊連接
- 如果有邊權的話通常會以 pair 來儲存

相鄰矩陣

- 這種方法會使用一個二維的表格，以 $\mathrm{G[i][j]}$ 代表點 (i, j) 之間是否有邊連接
- 如果有邊權的話通常會以 pair 來儲存
- 查詢一條邊是否存在：$\mathcal{O}(1)$
- 遍歷一個點的出邊：$\mathcal{O}(\mathrm{n})$
- 遍歷整張圖：$\mathcal{O}(\mathrm{n}^2)$
- 空間複雜度：$\mathcal{O}(\mathrm{n}^2)$

相鄰矩陣

- 這種方法會使用一個二維的表格，以 $\mathrm{G[i][j]}$ 代表點 (i, j) 之間是否有邊連接
- 如果有邊權的話通常會以 pair 來儲存
- 查詢一條邊是否存在：$\mathcal{O}(1)$
- 遍歷一個點的出邊：$\mathcal{O}(\mathrm{n})$
- 遍歷整張圖：$\mathcal{O}(\mathrm{n}^2)$
- 空間複雜度：$\mathcal{O}(\mathrm{n}^2)$
- 這種作法 $\mathrm{n} > 5000$ 就不能用了，所以很少出現在實際操作上
- 參考程式：build_graph_2darray.cpp

相鄰串列

- 相鄰矩陣會導致複雜度爆炸的根本原因在於他會遍歷太多根本沒有邊連接的點 (例如 $1, 5$ 之間沒有邊連接，但我們還是會遍歷到 $\mathrm{G}[1][5]$)

相鄰串列

- 相鄰矩陣會導致複雜度爆炸的根本原因在於他會遍歷太多根本沒有邊連接的點 (例如 $1, 5$ 之間沒有邊連接，但我們還是會遍歷到 $\mathrm{G}[1][5]$)
- 為了解決這個問題，我們可以只儲存哪些點就好

相鄰串列

- 相鄰矩陣會導致複雜度爆炸的根本原因在於他會遍歷太多根本沒有邊連接的點 (例如 $1, 5$ 之間沒有邊連接，但我們還是會遍歷到 $\mathrm{G}[1][5]$)
- 為了解決這個問題，我們可以只儲存哪些點就好
- 但是這樣我們就沒有辦法知道陣列要開多大了呀

相鄰串列

- 相鄰矩陣會導致複雜度爆炸的根本原因在於他會遍歷太多根本沒有邊連接的點 (例如 $1, 5$ 之間沒有邊連接，但我們還是會遍歷到 $G[1][5]$)
- 為了解決這個問題，我們可以只儲存哪些點就好
- 但是這樣我們就沒有辦法知道陣列要開多大了呀
- 簡單，那就用可以動態開點的陣列線段樹就好啦

相鄰串列

- 相鄰矩陣會導致複雜度爆炸的根本原因在於他會遍歷太多根本沒有邊連接的點 (例如 $1, 5$ 之間沒有邊連接，但我們還是會遍歷到 $\mathrm{G}[1][5]$)
- 為了解決這個問題，我們可以只儲存哪些點就好
- 但是這樣我們就沒有辦法知道陣列要開多大了呀
- 簡單，那就用可以動態開點的陣列線段樹就好啦
- 實作上我們會開 $\mathrm{n}+1$ 個 vector (vector<int> G[n + 1])，以 $\mathrm{G}[\mathrm{i}]$ 代表 i 相鄰的點有哪些
- 參考程式：build_graph_vector.cpp

相鄰串列

- 查詢一條邊 u, v 是否存在：$\mathcal{O}(\deg^+(\mathrm{u}))$ (若有經過排序那就會變成 $\mathcal{O}(\log \deg^+(\mathrm{u}))$)
- 遍歷一個點 u 的出邊：$\mathcal{O}(\deg^+(\mathrm{i}))$
- 遍歷整張圖：$\mathcal{O}(\mathrm{m})$
- 空間複雜度：$\mathcal{O}(\mathrm{m})$

相鄰串列

- 查詢一條邊 u, v 是否存在：$\mathcal{O}(\deg^+(u))$ (若有經過排序那就會變成 $\mathcal{O}(\log \deg^+(u))$)
- 遍歷一個點 u 的出邊：$\mathcal{O}(\deg^+(i))$
- 遍歷整張圖：$\mathcal{O}(m)$
- 空間複雜度：$\mathcal{O}(m)$
- 這樣的複雜度幾乎可以滿足所有的使用情況，因此最常被拿來使用在圖論相關的題目裡

相鄰串列

- 查詢一條邊 u, v 是否存在：$\mathcal{O}(\deg^+(u))$ (若有經過排序那就會變成 $\mathcal{O}(\log \deg^+(u))$)
- 遍歷一個點 u 的出邊：$\mathcal{O}(\deg^+(i))$
- 遍歷整張圖：$\mathcal{O}(m)$
- 空間複雜度：$\mathcal{O}(m)$
- 這樣的複雜度幾乎可以滿足所有的使用情況，因此最常被拿來使用在圖論相關的題目裡
- 接下來的內容大部分都會以這種方式來儲存一張圖

圖的遍歷

DFS 深度優先搜尋

- 想必大家在聽過搜尋的課之後都對遞迴枚舉有一定的熟悉吧

DFS 深度優先搜尋

- 想必大家在聽過搜尋的課之後都對遞迴枚舉有一定的熟悉吧
- 該如何將遞迴枚舉轉換成 DFS 呢？
- 先來複習一下遞迴枚舉的步驟：

DFS 深度優先搜尋

- 想必大家在聽過搜尋的課之後都對遞迴枚舉有一定的熟悉吧
- 該如何將遞迴枚舉轉換成 DFS 呢？
- 先來複習一下遞迴枚舉的步驟：

1. 呼叫起點

DFS 深度優先搜尋

- 想必大家在聽過搜尋的課之後都對遞迴枚舉有一定的熟悉吧
- 該如何將遞迴枚舉轉換成 DFS 呢？
- 先來複習一下遞迴枚舉的步驟：

1. 呼叫起點
2. 枚舉可以走到的點/狀態

DFS 深度優先搜尋

- 想必大家在聽過搜尋的課之後都對遞迴枚舉有一定的熟悉吧
- 該如何將遞迴枚舉轉換成 DFS 呢？
- 先來複習一下遞迴枚舉的步驟：

1. 呼叫起點
2. 枚舉可以走到的點/狀態
3. 如果該狀態沒走過 $\rightarrow$ 繼續往下走

DFS 深度優先搜尋

- 在圖論上也是一樣，唯一比較不一樣的是第二個步驟
- 該如何轉換呢？

DFS 深度優先搜尋

- 在圖論上也是一樣，唯一比較不一樣的是第二個步驟
- 該如何轉換呢？
- 前面提到，我們使用相鄰串列時會紀錄每個點跟哪些點有邊連接
- 所以我們的第二步驟就是換成枚舉 G[u] 中的所有點
- 範例程式：dfs_1.cpp

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接
- 將每個不包含 1 的連通塊中最小的城市跟 1 連接

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接
- 將每個不包含 1 的連通塊中最小的城市跟 1 連接
- 枚舉 $i = 1 \sim n$，若 i 尚未被 dfs 過，代表這個點跟 1 還不連通

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接
- 將每個不包含 1 的連通塊中最小的城市跟 1 連接
- 枚舉 $i = 1 \sim n$，若 i 尚未被 dfs 過，代表這個點跟 1 還不連通
- 代表 i 是一個不與 1 連通的連通塊中編號最小的城市

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接
- 將每個不包含 1 的連通塊中最小的城市跟 1 連接
- 枚舉 $i = 1 \sim n$，若 i 尚未被 dfs 過，代表這個點跟 1 還不連通
- 代表 i 是一個不與 1 連通的連通塊中編號最小的城市
- $\Rightarrow$ dfs 他，並蓋一條 $(1, i)$ 的道路

DFS 例題

CSES 1666 Building Roads

有 n 座城市，m 條雙向道路，第 i 條道路連接 a_i, b_i
求最少需要再建造幾條道路才可以讓每一座城市都互相連通
$(n \leq 10^5, m \leq 2 \times 10^5)$

- 每一座城市都互相連通代表每一座城市都跟城市 1 相通
- 每個不包含 1 的連通塊都需要有一條道路連接
- 將每個不包含 1 的連通塊中最小的城市跟 1 連接
- 枚舉 $i = 1 \sim n$，若 i 尚未被 dfs 過，代表這個點跟 1 還不連通
- 代表 i 是一個不與 1 連通的連通塊中編號最小的城市
- $\Rightarrow$ dfs 他，並蓋一條 $(1, i)$ 的道路
- 參考程式：Building_Roads.cpp

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 首先，我們需要先知道一個點 u 經過點 i 走到點 v 的路徑長要怎麼求

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 首先，我們需要先知道一個點 u 經過點 i 走到點 v 的路徑長要怎麼求
- 直接想可能想不太到，那假設 i 是根結點呢？

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 首先，我們需要先知道一個點 u 經過點 i 走到點 v 的路徑長要怎麼求
- 直接想可能想不太到，那假設 i 是根結點呢？
- 不難發現，u 到 v 的距離就是 $\mathrm{dep(u)} + \mathrm{dep(v)} - 2$

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 首先，我們需要先知道一個點 u 經過點 i 走到點 v 的路徑長要怎麼求
- 直接想可能想不太到，那假設 i 是根結點呢？
- 不難發現，u 到 v 的距離就是 $\mathrm{dep(u)} + \mathrm{dep(v)} - 2$
- 那如果 i 的深度是 2 呢？

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 首先，我們需要先知道一個點 u 經過點 i 走到點 v 的路徑長要怎麼求
- 直接想可能想不太到，那假設 i 是根結點呢？
- 不難發現，u 到 v 的距離就是 $\mathrm{dep(u)} + \mathrm{dep(v)} - 2$
- 那如果 i 的深度是 2 呢？
- 那麼 u 到 v 的距離就會變成 $\mathrm{dep(u)} + \mathrm{dep(v)} - 2 - 2$，因為少掉了從 i 到 root 的來回路途
- 那我們就可以得到一個式子：$\mathrm{dis(u, i, v)} = \mathrm{dep(u)} + \mathrm{dep(v)} - 2 \times \mathrm{dep(i)}$

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 我們可以枚舉每個點 i，看經過 i 的最長路徑，取最長的，就可以得到答案了

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 我們可以枚舉每個點 i，看經過 i 的最長路徑，取最長的，就可以得到答案了
- 但是要怎麼求得經過每個點的最長路徑呢？

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 我們可以枚舉每個點 i，看經過 i 的最長路徑，取最長的，就可以得到答案了
- 但是要怎麼求得經過每個點的最長路徑呢？
- 已知 $\mathrm{dep(i)}$，所以我們只需要盡可能的最大化 $\mathrm{dep(u)}, \mathrm{dep(v)}$ 就可以了

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

- 我們可以枚舉每個點 i，看經過 i 的最長路徑，取最長的，就可以得到答案了
- 但是要怎麼求得經過每個點的最長路徑呢？
- 已知 $\mathrm{dep(i)}$，所以我們只需要盡可能的最大化 $\mathrm{dep(u)}, \mathrm{dep(v)}$ 就可以了
- 我們可以改造一下我們的 DFS 函式，使其能夠回傳該點子樹內的最大深度

DFS 例題

TIOJ 1152 銀河帝國旅行社

有一棵結點數最多 10^4 的樹，請問樹上任兩點的最遠距離？

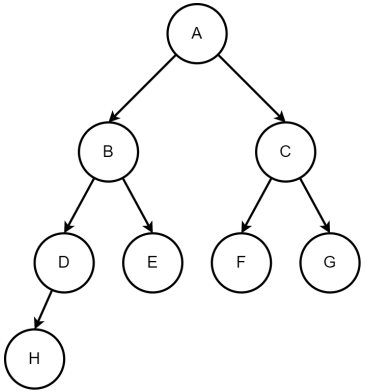
- 我們可以枚舉每個點 i，看經過 i 的最長路徑，取最長的，就可以得到答案了
- 但是要怎麼求得經過每個點的最長路徑呢？
- 已知 $\mathrm{dep(i)}$，所以我們只需要盡可能的最大化 $\mathrm{dep(u)}, \mathrm{dep(v)}$ 就可以了
- 我們可以改造一下我們的 DFS 函式，使其能夠回傳該點子樹內的最大深度
- 最後取 i 的子結點中回傳的前兩大 DFS 值，就可以得到最大的路徑長度了
- 參考程式：1152.cpp

樹壓平 Euler Tour Technique

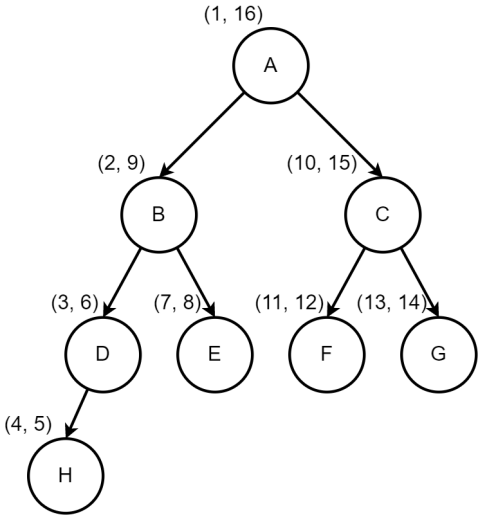
- 我們可以利用 DFS 過程的一些資訊，將一棵樹序列化，因此樹壓平也被稱為樹序列化

樹壓平 Euler Tour Technique

- 我們可以利用 DFS 過程的一些資訊，將一棵樹序列化，因此樹壓平也被稱為樹序列化
- 我們可以利用一個變數 timestamp 紀錄每一個點入點及出點的時間



樹壓平 Euler Tour Technique



樹壓平 Euler Tour Technique

- 如此一來我們就可以得到一個樹序列

ett	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
V	A	B	D	H	H	D	E	E	B	C	F	F	G	G	C	A

- 那這個樹序列可以做什麼事呢？

樹壓平 Euler Tour Technique

- 如此一來我們就可以得到一個樹序列

ett	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
V	A	B	D	H	H	D	E	E	B	C	F	F	G	G	C	A

- 那這個樹序列可以做什麼事呢？
- 觀察一下序列中兩個同樣字母之間的字母

樹壓平 Euler Tour Technique

- 如此一來我們就可以得到一個樹序列

ett	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
V	A	B	D	H	H	D	E	E	B	C	F	F	G	G	C	A

- 那這個樹序列可以做什麼事呢？
- 觀察一下序列中兩個同樣字母之間的字母
- 可以發現到，時間戳記位在某個點 u 入點及出點之間的點，就是 u 的子孫

樹壓平 Euler Tour Technique

- 如此一來我們就可以得到一個樹序列

ett	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
V	A	B	D	H	H	D	E	E	B	C	F	F	G	G	C	A

- 那這個樹序列可以做什麼事呢？
- 觀察一下序列中兩個同樣字母之間的字母
- 可以發現到，時間戳記位在某個點 u 入點及出點之間的點，就是 u 的子孫
- 所以我們可以利用樹序列來快速求得某個點 v 是否為點 u 的子孫

樹壓平 Euler Tour Technique

- 如此一來我們就可以得到一個樹序列

ett	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
V	A	B	D	H	H	D	E	E	B	C	F	F	G	G	C	A

- 那這個樹序列可以做什麼事呢？
- 觀察一下序列中兩個同樣字母之間的字母
- 可以發現到，時間戳記位在某個點 u 入點及出點之間的點，就是 u 的子孫
- 所以我們可以利用樹序列來快速求得某個點 v 是否為點 u 的子孫
- 這個技巧經常用來處理一些樹上的操作，例如 LCA (共同最長子序列) 或是一些樹上的區間操作等等
- 參考程式：ett.cpp

樹壓平例題

ABC 187E

有一 n 個節點、$n-1$ 條邊的樹，第 i 條邊連接 a_i, b_i，每個結點一開始都有一個值 c_i，c_i 一開始都是 0

共有 q 次操作，對於第 i 次操作輸入 t_i, e_i, x_i 有兩種情況：

1. $t_i = 1$：對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點，將他們的 c_i 全部加上 x_i
2. $t_i = 2$：對於所有 b_{e_i} 能走到且不經過 a_{e_i} 的點，將他們的 c_i 全部加上 x_i

求做完所有操作之後，每個點的 c_i 都是多少？$(n, q \leq 2 \times 10^5)$

- 先來解讀一下操作的意思：

樹壓平例題

ABC 187E

有一 n 個節點、$n-1$ 條邊的樹，第 i 條邊連接 a_i, b_i，每個結點一開始都有一個值 c_i，c_i 一開始都是 0

共有 q 次操作，對於第 i 次操作輸入 t_i, e_i, x_i 有兩種情況：

1. $t_i = 1$：對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點，將他們的 c_i 全部加上 x_i
2. $t_i = 2$：對於所有 b_{e_i} 能走到且不經過 a_{e_i} 的點，將他們的 c_i 全部加上 x_i

求做完所有操作之後，每個點的 c_i 都是多少？$(n, q \leq 2 \times 10^5)$

- 先來解讀一下操作的意思：
- 假設 $\mathrm{dep}(a_{e_i}) < \mathrm{dep}(b_{e_i})$，那麼對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點就是除了 b_{e_i} 的子孫的所有點，反之就是 b_{e_i} 的子孫

樹壓平例題

ABC 187E

有一 n 個節點、$n-1$ 條邊的樹，第 i 條邊連接 a_i, b_i，每個結點一開始都有一個值 c_i，c_i 一開始都是 0

共有 q 次操作，對於第 i 次操作輸入 t_i, e_i, x_i 有兩種情況：

1. $t_i = 1$：對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點，將他們的 c_i 全部加上 x_i
2. $t_i = 2$：對於所有 b_{e_i} 能走到且不經過 a_{e_i} 的點，將他們的 c_i 全部加上 x_i

求做完所有操作之後，每個點的 c_i 都是多少？$(n, q \leq 2 \times 10^5)$

- 先來解讀一下操作的意思：
- 假設 $\mathrm{dep}(a_{e_i}) < \mathrm{dep}(b_{e_i})$，那麼對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點就是除了 b_{e_i} 的子孫的所有點，反之就是 b_{e_i} 的子孫
- 所以我們需要做的事情就是將某個點的祖先或是子孫的 c_i 全部加上 x_i

樹壓平例題

ABC 187E

有一 n 個節點、$n-1$ 條邊的樹，第 i 條邊連接 a_i, b_i，每個結點一開始都有一個值 c_i，c_i 一開始都是 0

共有 q 次操作，對於第 i 次操作輸入 t_i, e_i, x_i 有兩種情況：

1. $t_i = 1$：對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點，將他們的 c_i 全部加上 x_i
2. $t_i = 2$：對於所有 b_{e_i} 能走到且不經過 a_{e_i} 的點，將他們的 c_i 全部加上 x_i

求做完所有操作之後，每個點的 c_i 都是多少？$(n, q \leq 2 \times 10^5)$

- 先來解讀一下操作的意思：
- 假設 $\mathrm{dep}(a_{e_i}) < \mathrm{dep}(b_{e_i})$，那麼對於所有 a_{e_i} 能走到且不經過 b_{e_i} 的點就是除了 b_{e_i} 的子孫的所有點，反之就是 b_{e_i} 的子孫
- 所以我們需要做的事情就是將某個點的祖先或是子孫的 c_i 全部加上 x_i
- 判斷子孫的部分剛剛講過了，要判斷點 v 是否為點 u 的祖先我們只需要判斷 $\mathrm{intime}(v) > \mathrm{intime}(u)$ 即可

樹壓平例題

- 定義 $\mathrm{cnt[i]}$ 是第 i 個時間點走到的點的 $\mathrm{c_i}$ 值
- 假設要操作的點是 $\mathrm{a, b}$ 且 $\mathrm{dep(a)} < \mathrm{dep(b)}$

樹壓平例題

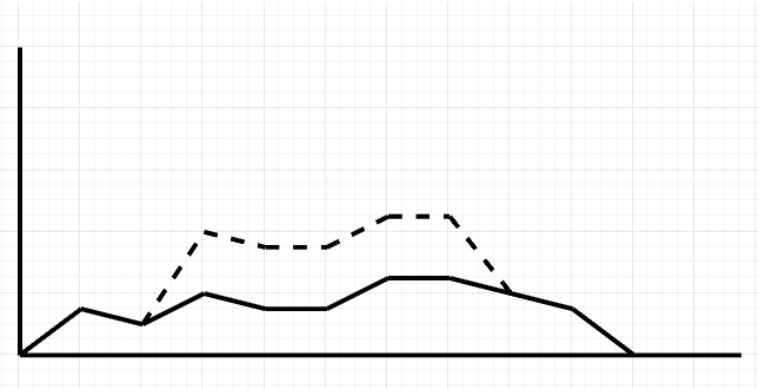
- 定義 $\mathrm{cnt[i]}$ 是第 i 個時間點走到的點的 $\mathrm{c_i}$ 值
- 假設要操作的點是 $\mathrm{a, b}$ 且 $\mathrm{dep(a) < dep(b)}$
- 那麼當 $\mathrm{t_i = 1}$ 時，就是將除了 $\mathrm{[intime[b], outtime[b]]}$ 以外的時間都加上 $\mathrm{x_i}$，也就是將 $\mathrm{[1, intime[b]), (outtime[b], n * 2]}$ 都加上 $\mathrm{x_i}$
- 反之，如果 $\mathrm{t_i = 2}$，就是將 $\mathrm{[intime[b], outtime[b]]}$ 都加上 $\mathrm{x_i}$
- 至於 $\mathrm{dep(a) > dep(b)}$ 就是相反

樹壓平例題

- 如果一個一個加的話時間複雜度會是恐怖的 $\mathcal{O}(\mathrm{nq})$

樹壓平例題

- 如果一個一個加的話時間複雜度會是恐怖的 $\mathcal{O}(\mathrm{nq})$
- 我們可以將 cnt 畫成 $\mathrm{x} - \mathrm{y}$ 圖



樹壓平例題

- 如果我們觀察斜率的話，可以發現到斜率只有在起點跟終點有變化而已

樹壓平例題

- 如果我們觀察斜率的話，可以發現到斜率只有在起點跟終點有變化而已
- 我們或許可以不維護 c_i，而是維護 $c_i - c_{i-1}$，也就是斜率
- 如此一來我們每次操作就只需要修改兩個點就可以了

樹壓平例題

- 如果我們觀察斜率的話，可以發現到斜率只有在起點跟終點有變化而已
- 我們或許可以不維護 c_i，而是維護 $c_i - c_{i-1}$，也就是斜率
- 如此一來我們每次操作就只需要修改兩個點就可以了
- 最後要還原出 c_i 的值，就只需要算出前綴和就可以了
- 這樣維護斜率而降低操作次數的數列就叫做**差分數列**

樹壓平例題

- 結合以上兩點，我們就可以整理出 $\mathrm{dep(a)} < \mathrm{dep(b)}$ 時的操作方式：

樹壓平例題

- 結合以上兩點，我們就可以整理出 $\mathrm{dep(a)} < \mathrm{dep(b)}$ 時的操作方式：
- $\mathrm{t_i} = 1$：cnt[1] += x, cnt[intime[b]] -= x, cnt[outtime[b] + 1] += x, cnt[n * 2 + 1] -= x
- $\mathrm{t_i} = 2$：cnt[intime[b]] += x, cnt[outtime[b] + 1] += x

樹壓平例題

- 結合以上兩點，我們就可以整理出 $\mathrm{dep(a)} < \mathrm{dep(b)}$ 時的操作方式：
- $\mathrm{t_i} = 1$：cnt[1] += x, cnt[intime[b]] -= x, cnt[outtime[b] + 1] += x, cnt[n * 2 + 1] -= x
- $\mathrm{t_i} = 2$：cnt[intime[b]] += x, cnt[outtime[b] + 1] += x
- 最後，如果要求出每個時間點確切的 $\mathrm{c_i}$ 值，就只需要對 cnt 做前綴和就好了！

樹壓平例題

- 結合以上兩點，我們就可以整理出 $\mathrm{dep(a)} < \mathrm{dep(b)}$ 時的操作方式：
- $\mathrm{t_i} = 1$：cnt[1] += x, cnt[intime[b]] -= x, cnt[outtime[b] + 1] += x, cnt[n * 2 + 1] -= x
- $\mathrm{t_i} = 2$：cnt[intime[b]] += x, cnt[outtime[b] + 1] += x
- 最後，如果要求出每個時間點確切的 $\mathrm{c_i}$ 值，就只需要對 cnt 做前綴和就好了！
- 因為我們在做樹壓平的時候有紀錄每個時間點走到的點是哪一個，所以可以直接利用樹壓平所記錄的資訊將 cnt 轉換成每個點的 $\mathrm{c_i}$
- 時間複雜度：$\mathcal{O}(\mathrm{n} + \mathrm{q})$
- 參考程式：ABC187E.cpp

BFS 廣度優先搜尋

- 將 BFS 改成在圖論上也是同理
- 將枚舉點的步驟改成是枚舉相鄰的點即可
- 需要注意的是，DFS 因為是遞迴，所以呼叫了就會馬上執行，但 BFS 是用 queue，所以在 push 的時候必須同時更新該點已經走過 否則可能會產生一個點被多次 push 的情況發生，這會導致你的複雜度直接爆炸
- 參考程式：bfs_1.cpp

BFS 例題

CSES 1667 Message Route

Sam 的網路有 n 台電腦還有 m 條網路線，每條網路線連接電腦 a_i, b_i
求從電腦 1 傳輸資料到電腦 n 的最短路徑，並將其構造出來 $(n \leq 10^5, m \leq 2 \times 10^5)$

- 求出最短需要幾步之後，要怎麼構造出路徑呢

BFS 例題

CSES 1667 Message Route

Sam 的網路有 n 台電腦還有 m 條網路線，每條網路線連接電腦 a_i, b_i
求從電腦 1 傳輸資料到電腦 n 的最短路徑，並將其構造出來 $(n \leq 10^5, m \leq 2 \times 10^5)$

- 求出最短需要幾步之後，要怎麼構造出路徑呢
- 我們可以紀錄每一個點是從哪裡走過來的

BFS 例題

CSES 1667 Message Route

Sam 的網路有 n 台電腦還有 m 條網路線，每條網路線連接電腦 a_i, b_i
求從電腦 1 傳輸資料到電腦 n 的最短路徑，並將其構造出來 $(n \leq 10^5, m \leq 2 \times 10^5)$

- 求出最短需要幾步之後，要怎麼構造出路徑呢
- 我們可以紀錄每一個點是從哪裡走過來的
- 全部走過之後，從終點往回走，並且把經過的點都記錄下來
- 就可以得到一組最短路徑了
- 參考程式：Message_Route.cpp

二分圖 Bipartite graph

什麼是二分圖

一張圖的節點分為兩個集合，且集合內部沒有邊的圖
如果將點塗為黑色或白色 (0/1)，那麼一個合法的二分圖中的每條邊必連接一黑點及一白點

二分圖

塗顏色問題

現在給你一張圖，你可以把每個節點塗成黑色或白色，但是每條邊相鄰的兩個點需要塗不同顏色
求能不能成功塗色

- 我們或許可以使用位元枚舉來枚舉每個點是不是 0 (黑色) 還是 1 (白色)，但這樣要 $\mathcal{O}(2^n)$，顯然太慢了

二分圖

塗顏色問題

現在給你一張圖，你可以把每個節點塗成黑色或白色，但是每條邊相鄰的兩個點需要塗不同顏色
求能不能成功塗色

- 我們或許可以使用位元枚舉來枚舉每個點是不是 0 (黑色) 還是 1 (白色)，但這樣要 $\mathcal{O}(2^n)$，顯然太慢了
- 我們可以先觀察到一個性質：
- 如果有一個點 u 被塗成黑色，那麼所有跟 u 相鄰的點 v 就一定只能被塗成白色

二分圖

塗顏色問題

現在給你一張圖，你可以把每個節點塗成黑色或白色，但是每條邊相鄰的兩個點需要塗不同顏色
求能不能成功塗色

- 我們或許可以使用位元枚舉來枚舉每個點是不是 0 (黑色) 還是 1 (白色)，但這樣要 $\mathcal{O}(2^n)$，顯然太慢了
- 我們可以先觀察到一個性質：
- 如果有一個點 u 被塗成黑色，那麼所有跟 u 相鄰的點 v 就一定只能被塗成白色
- 那我們就可以從一個點開始塗，然後不斷的將與其相鄰的點塗成相反顏色就好
- 可以用 dfs / bfs 來實作

二分圖

塗顏色問題

現在給你一張圖，你可以把每個節點塗成黑色或白色，但是每條邊相鄰的兩個點需要塗不同顏色
求能不能成功塗色

- 我們或許可以使用位元枚舉來枚舉每個點是不是 0 (黑色) 還是 1 (白色)，但這樣要 $\mathcal{O}(2^n)$，顯然太慢了
- 我們可以先觀察到一個性質：
- 如果有一個點 u 被塗成黑色，那麼所有跟 u 相鄰的點 v 就一定只能被塗成白色
- 那我們就可以從一個點開始塗，然後不斷的將與其相鄰的點塗成相反顏色就好
- 可以用 dfs / bfs 來實作
- 如果塗到一半發現有一個相鄰的點已經被塗過了，且該點的顏色跟目前點的顏色一樣，那就代表出現非法的情況了

二分圖例題

CSES 1668 Building Teams

NHDK 有 n 個人，其中有 m 條朋友關係，Sam 希望將這 n 個人分成兩組，且在同一組的人都沒有朋友關係
求出一組分組方式

- 基本上只需要將 dfs 過程中紀錄的塗色結果輸出出來就好了
- 參考程式：Building_Teams.cpp

二分圖例題

TPR17 pD New Year

TPR 國的總統想要將 TPR 國內的家庭分成兩組，並且另有姻親關係的兩個家庭會被分成不同的組別
總統以為自己很聰明，所以他先自己分了一次，但其實他分出來的錯誤率很大，為了不要打擊他的自尊心，請問總統提出的方案需要更改幾個家庭的分組才能夠使其合法，若沒有辦法的話則輸出 -1

二分圖例題

TPR17 pD New Year

TPR 國的總統想要將 TPR 國內的家庭分成兩組，並且另有姻親關係的兩個家庭會被分成不同的組別
總統以為自己很聰明，所以他先自己分了一次，但其實他分出來的錯誤率很大，為了不要打擊他的自尊心，請問總統提出的方案需要更改幾個家庭的分組才能夠使其合法，若沒有辦法的話則輸出 -1

- 可以觀察到一個性質
- 一個滿足二分圖的連通塊中，只有兩種分組方式

二分圖例題

TPR17 pD New Year

TPR 國的總統想要將 TPR 國內的家庭分成兩組，並且另有姻親關係的兩個家庭會被分成不同的組別
總統以為自己很聰明，所以他先自己分了一次，但其實他分出來的錯誤率很大，為了不要打擊他的自尊心，請問總統提出的方案需要更改幾個家庭的分組才能夠使其合法，若沒有辦法的話則輸出 -1

- 可以觀察到一個性質
- 一個滿足二分圖的連通塊中，只有兩種分組方式
- 那我們就針對每個連通塊，看哪種分組方式跟總統的最接近，那就選那個方案就好
- 參考程式：TPR17_PD.cpp

二分圖例題

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

二分圖例題

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 可以發現到，如果將一組合作關係的 $\mathrm{x_i}, \mathrm{y_i}$ 連成一條邊，那麼滿足兩人不在同一組內其實就是要滿足這張圖是一個二分圖

二分圖例題

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 可以發現到，如果將一組合作關係的 $\mathrm{x_i}, \mathrm{y_i}$ 連成一條邊，那麼滿足兩人不在同一組內其實就是要滿足這張圖是一個二分圖
- 那麼我們其實可以一個一個加邊，如果發現加上這條邊之後會使得這張圖不是一個二分圖，那就代表這個調查員的資料是錯的

二分圖例題

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 可以發現到，如果將一組合作關係的 $\mathrm{x_i}, \mathrm{y_i}$ 連成一條邊，那麼滿足兩人不在同一組內其實就是要滿足這張圖是一個二分圖
- 那麼我們其實可以一個一個加邊，如果發現加上這條邊之後會使得這張圖不是一個二分圖，那就代表這個調查員的資料是錯的
- 但以我們目前所學的東西，並沒有辦法在非 $\mathcal{O}(\mathrm{p} \mid \mathrm{E} \mid)$ 的時間做出來

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 但是我們可以觀察到一個性質

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 但是我們可以觀察到一個性質
- 當加入了一條邊會使其變得不合法之後，不管後面再加了甚麼邊，都不會再變成合法的了
- 這其實就是某種單調性

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 但是我們可以觀察到一個性質
- 當加入了一條邊會使其變得不合法之後，不管後面再加了甚麼邊，都不會再變成合法的了
- 這其實就是某種單調性
- 想到單調性就會想到二分搜，所以我們可以對他二分搜找出提供錯誤資料且還沒被發現的調查員中編號最小的一個

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 我們可以寫一個 check(mid) 代表使用調查員 $1 \sim \mathrm{mid}$ 的資料是否會是合法的

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 我們可以寫一個 check(mid) 代表使用調查員 $1 \sim \mathrm{mid}$ 的資料是否會是合法的
- 判斷合法一樣使用 DFS 就可以了

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 我們可以寫一個 check(mid) 代表使用調查員 $1 \sim \mathrm{mid}$ 的資料是否會是合法的
- 判斷合法一樣使用 DFS 就可以了
- 但我們要怎麼知道哪些邊是第 $1 \sim \mathrm{mid}$ 的調查員回傳的呢？

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 我們可以寫一個 check(mid) 代表使用調查員 $1 \sim \mathrm{mid}$ 的資料是否會是合法的
- 判斷合法一樣使用 DFS 就可以了
- 但我們要怎麼知道哪些邊是第 $1 \sim \mathrm{mid}$ 的調查員回傳的呢？
- 我們可以將存圖的 vector<int> 改成 vector<pair<int, int»，在 second 的地方存調查員的編號，在 DFS 時只走訪 second 是 $1 \sim \mathrm{mid}$ 的邊就可以了

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 找到一個提供錯誤資訊的調查員之後，我們可以將他標計為非法，之後在做 DFS 時也忽略該調查員提供的資料即可

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 找到一個提供錯誤資訊的調查員之後，我們可以將他標計為非法，之後在做 DFS 時也忽略該調查員提供的資料即可
- 因為已知錯誤的調查員最多只有 3 個，所以我們只需要二分搜 3 次就可以得到答案了

APCS 202111 P4 真假子圖 (ZJ g598)

NHDK 有 n 個工作人員，工作人員被分成 A, B 兩組，並且一個人 x 可能會跟另外一個人 y 合作，稱為一組合作關係，且 x, y 不會在同一組內
Colten 原先有每組合作關係的名單，但是因為一些因素遺失了一部分，只剩下 m 組合作關係，所以他派出了 p 個調查員來合作關係，每個調查員都會回傳 k 組合作關係
但是調查員回傳的資料可能不是正確的，也就是可能不會滿足兩個人不在同一組內，請你求出有哪些調查員的資料是錯誤的，已知回傳錯誤資料的調查員不超過三位

- 找到一個提供錯誤資訊的調查員之後，我們可以將他標計為非法，之後在做 DFS 時也忽略該調查員提供的資料即可
- 因為已知錯誤的調查員最多只有 3 個，所以我們只需要二分搜 3 次就可以得到答案了
- 時間複雜度 $\mathcal{O}(\mathrm{p} \log \mid \mathrm{E} \mid)$
- 據說這一場只有五個人滿級分
- 參考程式：g598.cpp

拓撲排序

什麼是拓撲排序

- 拓撲排序是用來解決圖上點的順序的方法
- 跟一般的排序無關

什麼是拓撲排序

- 拓撲排序是用來解決圖上點的順序的方法
- 跟一般的排序無關
- 能夠做拓撲排序的圖必定是一張有向無環圖 (DAG)
- 換句話說，要驗證一張圖是否是有向無環圖就可以用拓撲排序來驗證

拓撲排序的例子

想像你今天是一位大學生，你需要修一些課，但有些課需要先修完某些其他的課才能夠修，例如 A 上完才可以上 B，B 上完才可以上 C

- 這種情況顯然就是 $A \to B \to C$ 嘛

拓撲排序的例子

想像你今天是一位大學生，你需要修一些課，但有些課需要先修完某些其他的課才能夠修，例如 A 上完才可以上 B，B 上完才可以上 C

- 這種情況顯然就是 $A \to B \to C$ 嘛
- 那如果是 A 上完才可以上 B，B 上完才可以上 C，C 上完才可以上 A 呢？

拓撲排序的例子

想像你今天是一位大學生，你需要修一些課，但有些課需要先修完某些其他的課才能夠修，例如 A 上完才可以上 B，B 上完才可以上 C

- 這種情況顯然就是 $\mathrm{A} \to \mathrm{B} \to \mathrm{C}$ 嘛
- 那如果是 A 上完才可以上 B，B 上完才可以上 C，C 上完才可以上 A 呢？
- 我們就會發現它產生了矛盾，找不到一組拓撲排序

拓撲排序的例子

想像你今天是一位大學生，你需要修一些課，但有些課需要先修完某些其他的課才能夠修，例如 A 上完才可以上 B，B 上完才可以上 C

- 這種情況顯然就是 $\mathrm{A} \to \mathrm{B} \to \mathrm{C}$ 嘛
- 那如果是 A 上完才可以上 B，B 上完才可以上 C，C 上完才可以上 A 呢？
- 我們就會發現它產生了矛盾，找不到一組拓撲排序
- 拓撲排序就是在解決這類型的問題

如何求拓撲排序

- 拓撲排序主要有兩種求法
  1. DFS
  2. Kahn 演算法

如何求拓撲排序

- 拓撲排序主要有兩種求法
  1. DFS
  2. Kahn 演算法
- 在找拓撲排序之前，我們需要先將關係圖建成一張圖

如何求拓撲排序

- 拓撲排序主要有兩種求法
  1. DFS
  2. Kahn 演算法
- 在找拓撲排序之前，我們需要先將關係圖建成一張圖
- 如果需要先修 u 才能修 v 的話，我們會建一張由 u 連向 v 的邊

DFS 求拓撲排序

- 剛剛有說到，是一張有向無環圖就沒有辦法拓撲排序
- 所以我們的首要目的就是要找出這張圖有沒有環

DFS 求拓撲排序

- 剛剛有說到，是一張有向無環圖就沒有辦法拓撲排序
- 所以我們的首要目的就是要找出這張圖有沒有環
- 假設我們在 dfs 的時候走到了一個點 u，與其相鄰的點 v 中有任意一個存在於目前所走的路徑上，那就代表這張圖有環
- 那該如何得知一個點在目前所走的路徑上呢？

DFS 求拓撲排序

- 剛剛有說到，是一張有向無環圖就沒有辦法拓撲排序
- 所以我們的首要目的就是要找出這張圖有沒有環
- 假設我們在 dfs 的時候走到了一個點 u，與其相鄰的點 v 中有任意一個存在於目前所走的路徑上，那就代表這張圖有環
- 那該如何得知一個點在目前所走的路徑上呢？
- 我們通常會開一個 vis[] 代表哪些點走過了，並且在走到一個點 i 的時候就把 vis[i] 標為 1

DFS 求拓撲排序

- 剛剛有說到，是一張有向無環圖就沒有辦法拓撲排序
- 所以我們的首要目的就是要找出這張圖有沒有環
- 假設我們在 dfs 的時候走到了一個點 u，與其相鄰的點 v 中有任意一個存在於目前所走的路徑上，那就代表這張圖有環
- 那該如何得知一個點在目前所走的路徑上呢？
- 我們通常會開一個 vis[] 代表哪些點走過了，並且在走到一個點 i 的時候就把 vis[i] 標為 1
- 我們可以擴充一下，如果點 i 在目前走的路徑上，就標為 2，而退出該點的時候再標回 1 代表該點走過但並沒有位在目前的路徑上

DFS 求拓撲排序

- 剛剛有說到，是一張有向無環圖就沒有辦法拓撲排序
- 所以我們的首要目的就是要找出這張圖有沒有環
- 假設我們在 dfs 的時候走到了一個點 u，與其相鄰的點 v 中有任意一個存在於目前所走的路徑上，那就代表這張圖有環
- 那該如何得知一個點在目前所走的路徑上呢？
- 我們通常會開一個 vis[] 代表哪些點走過了，並且在走到一個點 i 的時候就把 vis[i] 標為 1
- 我們可以擴充一下，如果點 i 在目前走的路徑上，就標為 2，而退出該點的時候再標回 1 代表該點走過但並沒有位在目前的路徑上
- 如此一來，我們就能夠寫出一個能夠判斷一張圖有沒有環的 DFS 了

求是否有環的練習

CSES 1669 Round Trip

TPR 國有 n 座城市，其中有 m 條道路
請問是否有一個路徑起終點相同，且經過的城市數量 > 1，如果有則將其構造出來
$(\mathrm{n} \leq 10^5, \mathrm{m} \leq 2 \times 10^5)$

- 這一題除了要找到是否有環，還要構造出一組環的解

求是否有環的練習

CSES 1669 Round Trip

TPR 國有 n 座城市，其中有 m 條道路
請問是否有一個路徑起終點相同，且經過的城市數量 > 1，如果有則將其構造出來
$(\mathrm{n} \leq 10^5, \mathrm{m} \leq 2 \times 10^5)$

- 這一題除了要找到是否有環，還要構造出一組環的解
- 我們可以利用一個 stack 來儲存目前路徑上的點

求是否有環的練習

CSES 1669 Round Trip

TPR 國有 n 座城市，其中有 m 條道路
請問是否有一個路徑起終點相同，且經過的城市數量 > 1，如果有則將其構造出來
$(\mathrm{n} \leq 10^5, \mathrm{m} \leq 2 \times 10^5)$

- 這一題除了要找到是否有環，還要構造出一組環的解
- 我們可以利用一個 stack 來儲存目前路徑上的點
- 如果遇到環的起終點 u (vis[u] == 2)，那我們就將這個 stack 裡面的點都取出來，直到取到 u 為止，那我們就可以成功地將這個環裡面的點都取出來了
- 參考程式：Round_Trip.cpp

DFS 求拓撲排序

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 知道怎麼判斷非法情況之後，就要來構造出一組解了

DFS 求拓撲排序

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 知道怎麼判斷非法情況之後，就要來構造出一組解了
- 可以發現到，在 DFS 的過程中，第一個走出去的點，也就是第一個沒有其他點可以走的點，就會是最後可以走的點

DFS 求拓撲排序

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 知道怎麼判斷非法情況之後，就要來構造出一組解了
- 可以發現到，在 DFS 的過程中，第一個走出去的點，也就是第一個沒有其他點可以走的點，就會是最後可以走的點
- 以此類推，第二個走出去的點，就是倒數第二個可以走的點

DFS 求拓撲排序

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 知道怎麼判斷非法情況之後，就要來構造出一組解了
- 可以發現到，在 DFS 的過程中，第一個走出去的點，也就是第一個沒有其他點可以走的點，就會是最後可以走的點
- 以此類推，第二個走出去的點，就是倒數第二個可以走的點
- 所以我們只要紀錄在 DFS 的過程中，出點的順序，最後再將他反轉，就可以得到一個拓撲排序了
- 參考程式：Course_Schedule_dfs.cpp

Kahn 演算法

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 觀察一下選課的例子，一門課可以選需要滿足什麼條件？

Kahn 演算法

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 觀察一下選課的例子，一門課可以選需要滿足什麼條件？
- 不難發現到，當修一門課前需要先修的課都修完，我們就可以修該門課

Kahn 演算法

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 觀察一下選課的例子，一門課可以選需要滿足什麼條件？
- 不難發現到，當修一門課前需要先修的課都修完，我們就可以修該門課
- 該怎麼把這個性質轉成圖論呢？
- 把這個性質轉成圖論之後，我們就可以得到一個規則：當一個點 u 的所有入邊所連接的點 v 都走過之後，u 就可以走

Kahn 演算法

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 觀察一下選課的例子，一門課可以選需要滿足什麼條件？
- 不難發現到，當修一門課前需要先修的課都修完，我們就可以修該門課
- 該怎麼把這個性質轉成圖論呢？
- 把這個性質轉成圖論之後，我們就可以得到一個規則：當一個點 u 的所有入邊所連接的點 v 都走過之後，u 就可以走
- 我們可以得到一個策略：先從入度為 0 的點開始，每走到一個點 u，就把 u 的出邊能走到的點的 v 入度都 -1，當 $\deg^+(\mathrm{v}) = 0$ 時，那就代表 v 可以走了，那我們就將他放進 queue 當中

Kahn 演算法

CSES 1679 Course Schedule

給一張有向圖，輸出一組拓撲排序

- 觀察一下選課的例子，一門課可以選需要滿足什麼條件？
- 不難發現到，當修一門課前需要先修的課都修完，我們就可以修該門課
- 該怎麼把這個性質轉成圖論呢？
- 把這個性質轉成圖論之後，我們就可以得到一個規則：當一個點 u 的所有入邊所連接的點 v 都走過之後，u 就可以走
- 我們可以得到一個策略：先從入度為 0 的點開始，每走到一個點 u，就把 u 的出邊能走到的點的 v 入度都 -1，當 $\deg^+(\mathrm{v}) = 0$ 時，那就代表 v 可以走了，那我們就將他放進 queue 當中
- 這其實就是有限制條件的 BFS
- 參考程式：Course_Schedule_bfs.cpp

複雜度分析

- 時間複雜度：$\mathcal{O}(\mathrm{V})$
- 空間複雜度：$\mathcal{O}(\mathrm{V}+\mathrm{E})$

拓撲排序例題

CF 510C Fox And Names

Foxyy 想要出版一篇論文，他聽說作者列表總是會按照字典序排列，但他發現事實不是這樣，
給你作者的列表，請問你有沒有辦法將字母的順序重新排列，使得作者列表滿足按照字典序排列？

拓撲排序例題

CF 510C Fox And Names

Foxyy 想要出版一篇論文，他聽說作者列表總是會按照字典序排列，但他發現事實不是這樣，
給你作者的列表，請問你有沒有辦法將字母的順序重新排列，使得作者列表滿足按照字典序排列？

- 以第一筆範測的第 2、3 個作者為例：shamir 與 adleman 在比較字典序時，會比 s、a 哪個比較小

拓撲排序例題

CF 510C Fox And Names

Foxyy 想要出版一篇論文，他聽說作者列表總是會按照字典序排列，但他發現事實不是這樣，
給你作者的列表，請問你有沒有辦法將字母的順序重新排列，使得作者列表滿足按照字典序排列？

- 以第一筆範測的第 2、3 個作者為例：shamir 與 adleman 在比較字典序時，會比 s、a 哪個比較小
- 由於在作者列表裡 shamir < adleman，所以我們就可以得知在新的字母排列中，s < a，那麼我們就可以連一條 $\mathrm{s} \to \mathrm{a}$ 的邊

拓撲排序例題

CF 510C Fox And Names

Foxyy 想要出版一篇論文，他聽說作者列表總是會按照字典序排列，但他發現事實不是這樣，
給你作者的列表，請問你有沒有辦法將字母的順序重新排列，使得作者列表滿足按照字典序排列？

- 以第一筆範測的第 2、3 個作者為例：shamir 與 adleman 在比較字典序時，會比 s、a 哪個比較小
- 由於在作者列表裡 shamir < adleman，所以我們就可以得知在新的字母排列中，s < a，那麼我們就可以連一條 $\mathrm{s} \to \mathrm{a}$ 的邊
- 最後，依照我們所連的邊去做拓撲排序，就可以得到我們的答案了
- 參考程式：510C.cpp

拓撲排序例題

CF 1385E Directing Edges

有一張混和圖，請問所有的無向邊定向之後能否使這張圖能夠被拓撲排序？
如果可以，請輸出 YES 以及所有邊的內容，否則請輸出 NO。$(\mathrm{n} \leq 2 \times 10^5)$

拓撲排序例題

CF 1385E Directing Edges

有一張混和圖，請問所有的無向邊定向之後能否使這張圖能夠被拓撲排序？
如果可以，請輸出 YES 以及所有邊的內容，否則請輸出 NO。$(n \leq 2 \times 10^5)$

- 可以發現到，一條無向邊的存在並不影響一張圖能否拓撲排序

拓撲排序例題

CF 1385E Directing Edges

有一張混和圖，請問所有的無向邊定向之後能否使這張圖能夠被拓撲排序？
如果可以，請輸出 YES 以及所有邊的內容，否則請輸出 NO。$(n \leq 2 \times 10^5)$

- 可以發現到，一條無向邊的存在並不影響一張圖能否拓撲排序
- 因為一條邊所連接的兩點，一定有一個點是比較早走，一個是比較晚走的

拓撲排序例題

CF 1385E Directing Edges

有一張混和圖，請問所有的無向邊定向之後能否使這張圖能夠被拓撲排序？
如果可以，請輸出 YES 以及所有邊的內容，否則請輸出 NO。$(\mathrm{n} \leq 2 \times 10^5)$

- 可以發現到，一條無向邊的存在並不影響一張圖能否拓撲排序
- 因為一條邊所連接的兩點，一定有一個點是比較早走，一個是比較晚走的
- 所以我們只需要考慮包含有向邊的子圖

拓撲排序例題

CF 1385E Directing Edges

有一張混和圖，請問所有的無向邊定向之後能否使這張圖能夠被拓撲排序？
如果可以，請輸出 YES 以及所有邊的內容，否則請輸出 NO。$(n \leq 2 \times 10^5)$

- 可以發現到，一條無向邊的存在並不影響一張圖能否拓撲排序
- 因為一條邊所連接的兩點，一定有一個點是比較早走，一個是比較晚走的
- 所以我們只需要考慮包含有向邊的子圖
- 得出了拓撲排序的順序之後，再將無向邊定向為早走的點 $\rightarrow$ 晚走的點即可
- 參考程式：1385E.cpp

求最小字典序的拓撲排序

ABC 223D Restricted Permutation

求出一組最小字典序的拓撲排序

- 因為字典序是優先比較前面的元素，所以我們可以得到一個策略：優先走編號小的點

求最小字典序的拓撲排序

ABC 223D Restricted Permutation

求出一組最小字典序的拓撲排序

- 因為字典序是優先比較前面的元素，所以我們可以得到一個策略：優先走編號小的點
- 因為 queue 裡面的元素就是目前可以走的點，所以我們要執行這個策略就只需要在 queue 裡面挑最小的點走

求最小字典序的拓撲排序

ABC 223D Restricted Permutation

求出一組最小字典序的拓撲排序

- 因為字典序是優先比較前面的元素，所以我們可以得到一個策略：優先走編號小的點
- 因為 queue 裡面的元素就是目前可以走的點，所以我們要執行這個策略就只需要在 queue 裡面挑最小的點走
- queue、取最小值，應該不難想到可以使用 priority_queue
- 只需要將 queue 改成 priority_queue 就可以達成求最小字典序了
- 參考程式：topo_mindict.cpp

DAG 上 DP

DAG 上 DP

- 還記得早上的 DP 課嗎
- 如果今天把 DP 搬到了 DAG 上呢？

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 先假設他是一條鍊，且所有邊的方向都是一樣的

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 先假設他是一條鍊，且所有邊的方向都是一樣的
- 那我們可以很簡單的求出一個轉移式：dp[i] = max(dp[i], dp[i - 1] + 1)

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 先假設他是一條鍊，且所有邊的方向都是一樣的
- 那我們可以很簡單的求出一個轉移式：dp[i] = max(dp[i], dp[i - 1] + 1)
- 可以發現到，我們所做的事情就是在從以前走過的點做轉移
- 也就是說，我們需要先走過前面的點，才能夠走後面的點

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 那搬到圖上呢？

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 那搬到圖上呢？
- 可以發現到，” 需要先走過前面的點，才能夠走後面的點” 這個條件，在圖上其實就是拓撲排序的條件

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 那搬到圖上呢？
- 可以發現到，” 需要先走過前面的點，才能夠走後面的點” 這個條件，在圖上其實就是拓撲排序的條件
- 所以我們在做轉移的時候，只需要按照拓撲排序的順序走訪並轉移就可以了

DAG 上 DP

Atcoder DP Contest pG Longest Path

給一個 DAG 的有向圖，求一條最長的路徑

- 那搬到圖上呢？
- 可以發現到，” 需要先走過前面的點，才能夠走後面的點” 這個條件，在圖上其實就是拓撲排序的條件
- 所以我們在做轉移的時候，只需要按照拓撲排序的順序走訪並轉移就可以了
- 假設我們目前在點 u，與其相臨的點為 v 那我們就可以得到一個轉移式：dp[v] = max(dp[v], dp[u] + 1)
- 參考程式：ATCDP_pG.cpp

DAG 上 DP

ABC 188E Peddler

有 n 座城市，m 條**單向**道路，第 i 條道路連接 a_i, b_i，
有一個商人想要在這 n 座城市間交易，已知第 i 座城市的交易價為 p_i (買價與賣價相同)，
請問在只買賣各一次的情況下，商人最多可以獲利多少元？$(n, m \leq 2 \times 10^5)$

- 我們一樣先把他看成是一條鍊來想

DAG 上 DP

ABC 188E Peddler

有 n 座城市，m 條**單向**道路，第 i 條道路連接 a_i, b_i，
有一個商人想要在這 n 座城市間交易，已知第 i 座城市的交易價為 p_i (買價與賣價相同)，
請問在只買賣各一次的情況下，商人最多可以獲利多少元？$(n, m \leq 2 \times 10^5)$

- 我們一樣先把他看成是一條鍊來想
- 在第 i 個點賣掉商品的最大獲利就是 p[i] - min(p[j]) (j < i)

DAG 上 DP

ABC 188E Peddler

有 n 座城市，m 條**單向**道路，第 i 條道路連接 a_i, b_i，
有一個商人想要在這 n 座城市間交易，已知第 i 座城市的交易價為 p_i (買價與賣價相同)，
請問在只買賣各一次的情況下，商人最多可以獲利多少元？$(n, m \leq 2 \times 10^5)$

- 我們一樣先把他看成是一條鍊來想
- 在第 i 個點賣掉商品的最大獲利就是 p[i] - min(p[j]) (j < i)
- 最後的答案就會是 ans = max(p[i] - min(p[j])) (j < i)

DAG 上 DP

ABC 188E Peddler

有 n 座城市，m 條**單向**道路，第 i 條道路連接 a_i, b_i，
有一個商人想要在這 n 座城市間交易，已知第 i 座城市的交易價為 p_i (買價與賣價相同)，
請問在只買賣各一次的情況下，商人最多可以獲利多少元？$(n, m \leq 2 \times 10^5)$

- 我們一樣先把他看成是一條鍊來想
- 在第 i 個點賣掉商品的最大獲利就是 p[i] - min(p[j]) (j < i)
- 最後的答案就會是 ans = max(p[i] - min(p[j])) (j < i)
- 設 dp[i] 為 i 以前的最小買入價格，就可以得到最後的轉移式 ans = max(p[i] - dp[i])

DAG 上 DP

ABC 188E Peddler

有 n 座城市，m 條**單向**道路，第 i 條道路連接 a_i, b_i，
有一個商人想要在這 n 座城市間交易，已知第 i 座城市的交易價為 p_i (買價與賣價相同)，
請問在只買賣各一次的情況下，商人最多可以獲利多少元？$(n, m \leq 2 \times 10^5)$

- 我們一樣先把他看成是一條鍊來想
- 在第 i 個點賣掉商品的最大獲利就是 p[i] - min(p[j]) (j < i)
- 最後的答案就會是 ans = max(p[i] - min(p[j])) (j < i)
- 設 dp[i] 為 i 以前的最小買入價格，就可以得到最後的轉移式 ans = max(p[i] - dp[i])
- 最後，套上拓撲排序就可以了
- 參考程式：ABC188E.cpp

總結

總結

- 可以發現到，很多圖論演算法都是基於 DFS、BFS 去做延伸與修改的
- 所以要精熟圖論，就要先精熟 DFS、BFS
- 如果覺得很抽象的話，可以適度的把走訪順序都印出來
- 希望大家能夠對圖論有一定的熟悉