Advanced
Competitive Programming

國立成功大學ACM-ICPC程式競賽培訓隊
nckuacm@imslab.org

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

Week 11
Advance Graph

Articulation Point, Strongly Connected Components, Lowest Common Ancestor

在開始之前，先複習一下符號

- G 代表題目所給定的圖
- V 代表點集合，E 代表邊集合
  （另外 $|S|$ 代表集合大小）
- $Edge(a,b)$ 代表 a,b 之間的邊
- $Dist(a,b)$ 代表 a,b 之間的距離

Tarjan

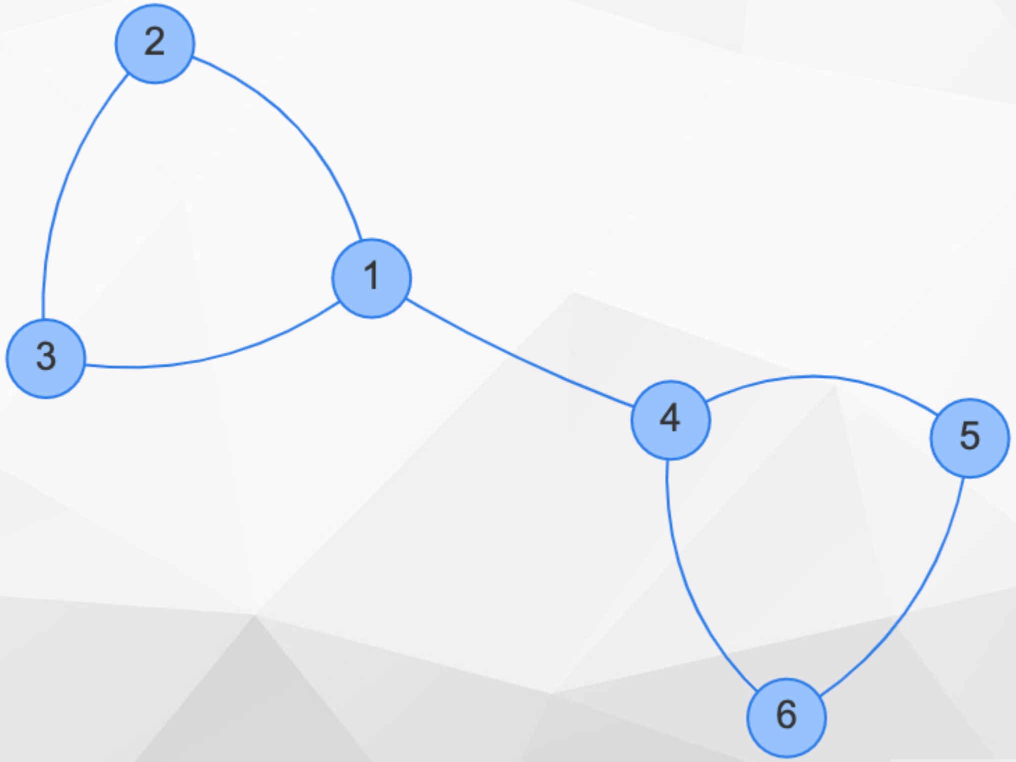
準備好迎接 Tarjan 全家桶了嗎

Before Tarjan

- 割點（支那用語，又稱關節點，AP）
- 橋（bridge）

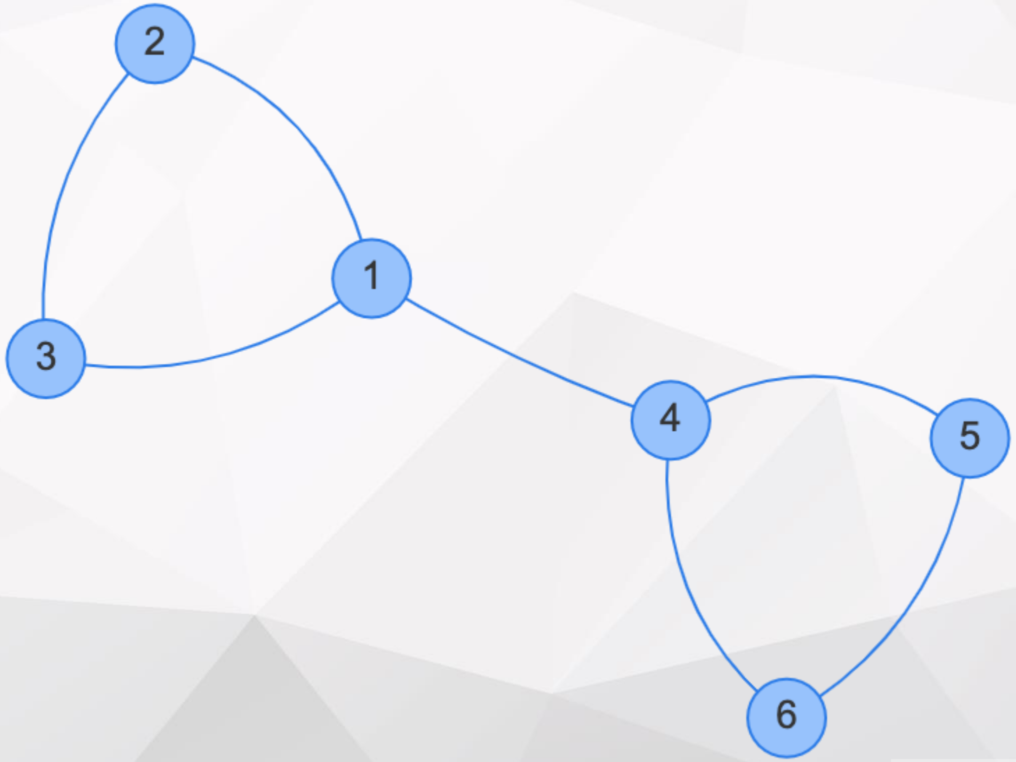
割點

- 如果把這個點拔掉，此聯通塊會一分為二，則稱呼這個點為割點
- 以下圖為例，割點即 1,4



橋

- 如果把這個邊拔掉，會讓聯通塊一分為二，那麼就稱那個邊為「橋」
- 同樣的一張圖，橋即 $Edge(1,4)$



Robert Endre Tarjan

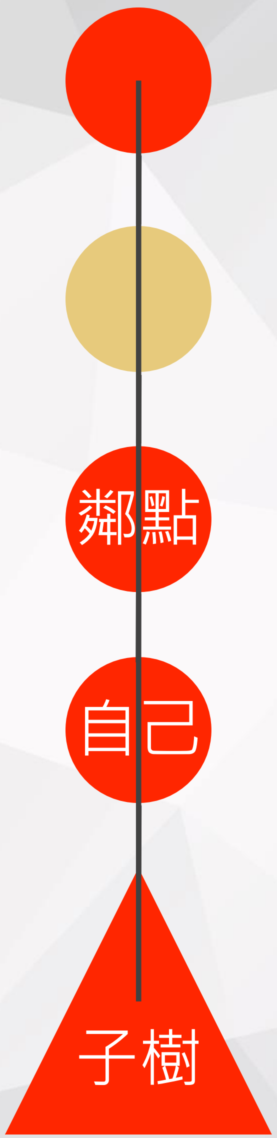
- 著名圖論大師，1986 年圖靈獎得主
- 開發過許多著名演算法，且不少都以他的名字命名，很常讓人搞混

統一的符號

- 令 T 為 G 的搜尋樹
- $D(u)$ 表示在建立 T 時，u 第一次經過時的深度
- $L(u)$ 表示對於 u，其在 T 中的子樹內所有節點，和這些節點在 G 上的鄰點，以及 u 本身的最低深度

$L(u)$

- 上一頁聽起來有點繞口
  不過換成這樣好像就比較好理解了：
  - 假設 T_u 表示在 u 在 T 上的子樹
  - 那麼 $L(u) = min(L(i)), \forall i \in T_u$



根據定義，我們可以有這些結果

- $rootisAP \Leftrightarrow root$ 有兩個子節點
- 除了 $root$ 以外的所有點 u，在 G 上要成為 AP 的充要條件為，在 T 中至少有一個子節點 w 滿足 $D(u) \leq L(w)$
- 包含 $root$ 在內的所有節點 u 和其子節點 w，$Edge(u, w)$在 G 中要成為 bridge 的充要條件為 $D(u) \leq L(w)$
（或是說有其中一個點是割點）

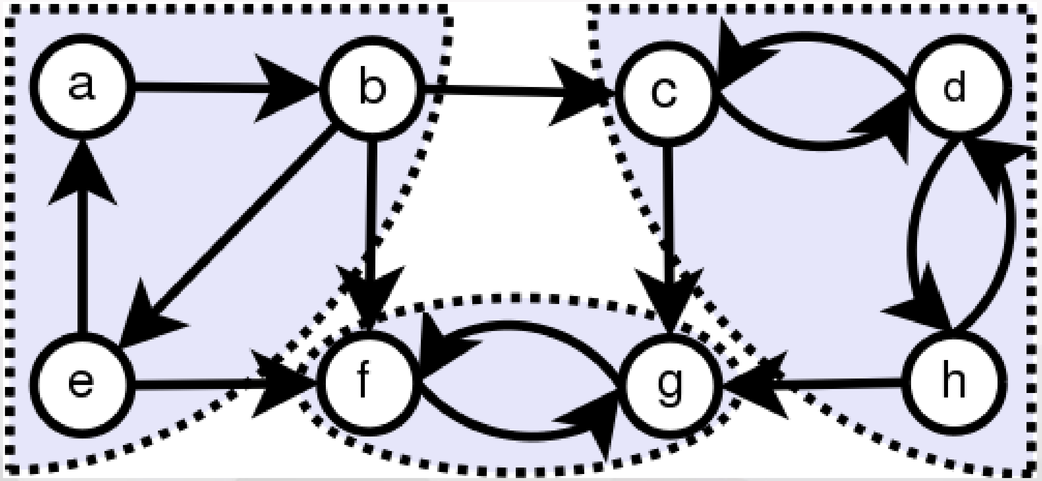
code

```cpp
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define maxN 100005
int D[maxN], L[maxN], tms;
vector < int > edges[maxN], AP;
vector < pair < int, int > > bridge;

void dfs ( int n, int p ){
    D[n] = L[n] = tms++;
    int cnt = 0;
    bool isAP = false;

    for ( auto i: edges[n] ){
        if ( i == p )
            continue;
        if ( !D[i] ){
            dfs ( i, n );
            cnt++;
            if ( D[n] <= L[i] )
                isAP = true;
            if ( D[n] <= L[i] )
                bridge.push_back ( make_pair ( n, i ) );
            L[n] = min ( L[n], L[i] );
        }
    }

    if ( n == p && cnt < 2 )
        isAP = false;
    if ( isAP )
        AP.push_back ( n );
}
```

強聯通分量（SCC）

何為強聯通分量

- 在有向圖中有些點可以互相到達，則會稱那些點在同一個 SCC 中
- 如下圖，a, b, e 在同一個 SCC 中
- SCC 是由多個環組成



縮點

- 在同一個 SCC 中的點都可以互相到達
- 在某些情況下可以把這些點當作是同一個點
- 令經過縮點操作的圖 G'，則 G' 保證為有向無環圖
  - 因為已經經過縮點了，所有環都會被壓成 SCC，故不會存在任何環
- 若 u 符合 $L(u) = D(u)$，則 u 的子樹所有未縮點的點可構成 SCC

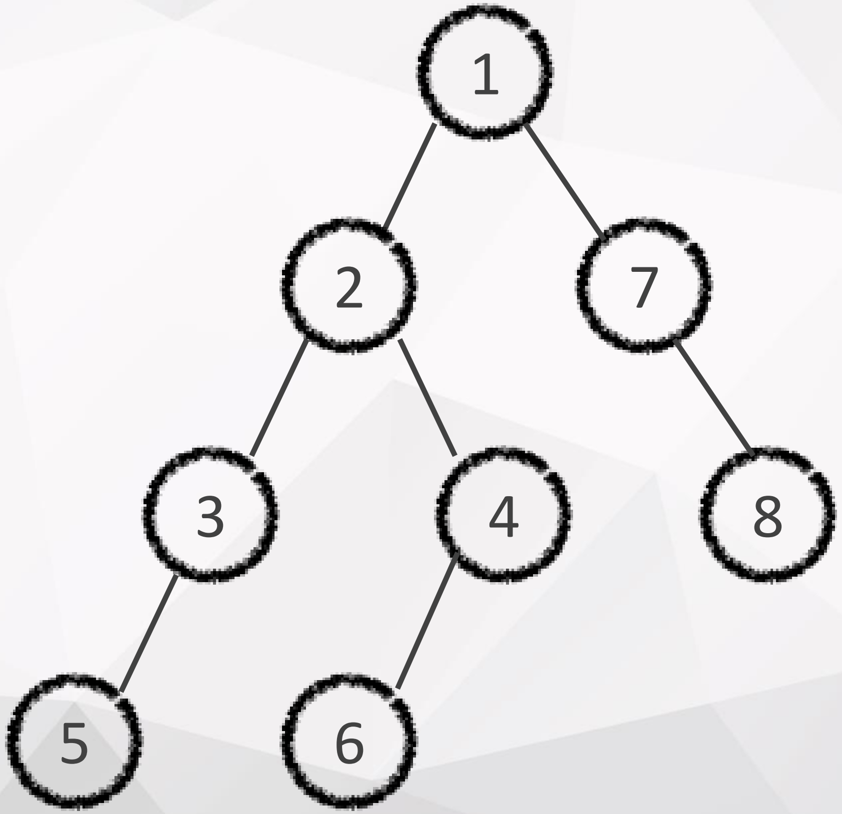
code

```cpp
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define maxN 100005

vector < int > edges[maxN];
int D[maxN], L[maxN], scc[maxN], tms = 1;
stack < int, vector < int > > st; // 用 vector 取代 deque
bool inSt[maxN];

void dfs ( int n ){
    D[n] = L[n] = tms++;
    st.push ( n );
    inSt[n] = true;
    for ( auto i: edges[n] ){
        if ( !D[i] ) // 還沒遍歷過就先 dfs
            dfs ( i );
        if ( inSt[i] ) // 走過就更新 L ( n )
            L[n] = min ( L[n], L[i] );
    }

    if ( D[n] == L[n] ){ // 如果條件符合 -> 做 SCC 縮點
        int swp = -1;
        while ( swp != p ){ // 做到 N 也處理完
            swp = st.top();
            st.pop();
            scc[swp] = n; // 紀錄與哪個點在同一個 SCC 中
                          // 或是另外紀錄 SCC 編號也可以

            inSt[swp] = false;
        }
    }
}

void SCC ( int n ){
    memset ( D, 0, sizeof D );
    for ( int i = 0 ; i < n ; i++ )
        if ( !D[i] ) // 還未遍歷過就對他 dfs
            dfs ( i );
}
```

Lowest Common Ancestor

最低共同祖先

- 顧名思義，是指在同一棵有根樹上，任意兩點的祖先中，最低且相同的節點
- 如右圖，3 & 4 的 LCA 為 2，5 & 8 是 1



先從暴力法開始

- 先將較低的節點往上（根節點）爬到與較高節點同高（也就是說，兩個節點的深度一樣）
- 將兩個節點同時向上拉，直到兩個節點重疊
- 單次查詢複雜度 $O(N)$

code

```
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define maxN 100005

vector < int > edges[maxN];
int parent[maxN], D[maxN];

void dfs ( int n, int p, int dep ){
    D[n] = dep++;
    parent[n] = p;
    for ( auto i: edges[n] )
        if ( i != p )
            dfs ( i );
}

inline void build ( void ){
    dfs ( 1 );
}

inline int LCA ( int u, int v ){
    if ( D[u] > D[v] )
        swap ( u, v );
    while ( D[u] != D[v] )
        v = parent[v];
    while ( u != v )
        u = parent[u], v = parent[v];

    return u;
}
```

想一下

- 經過觀察，每次做查詢都要往上跳 K 個節點
- 那麼預處理一下，每個節點的往上一層的節點、往上兩層的節點...

然後呢

- 複雜度超差，預處理 $O(NM)$，還浪費一大堆記憶體

啊哈，二進位！

- 沒錯，就是二進位分解
- 所以只要紀錄往上 2^k 個祖先就好啦

LCA 倍增法

LCA 倍增法

- $def: dp[u][k] = u$ 往上第 k 個祖先
  $\rightarrow dp[u][k] = dp[dp[u][k-1]][k-1]$
- 有點像是 Sparse Table
  - 操作步驟跟基礎 LCA 很像，也是先把兩邊調整到同樣的高度，接著同時往上跳

code

```
// by. MiohitoKiri5474
#include<bits/stdc++.h>

using namespace std;

#define maxN 100005
#define maxLog 17

vector < int > edges[maxN];
int parent[maxN], D[maxN], dp[maxN][maxLog], n;

void dfs ( int n, int p, int dep ){
    D[n] = dep++;
    parent[n] = p;
    for ( auto i: edges[n] )
        if ( i != p )
            dfs ( i, n, dep );
}

inline void build ( void ){
    memset ( dp, -1, sizeof dp );
    dfs ( 0, -1, 0 );
    for ( int k = 1 ; k < maxLog ; k++ )
        for ( int i = 0 ; i < n ; i++ )
            if ( dp[i][k - 1] != -1 )
                dp[i][k] = dp[dp[i][k - 1]][k - 1];
}

inline int findLCA ( int u, int v ){
    if ( D[u] < D[v] )
        swap ( u, v );

    for ( int i = maxLog - 1 ; i >= 0 ; i-- )
        if ( dp[u][i] != -1 && D[dp[u][i]] >= D[v] )
            u = dp[u][i];

    if ( u == v )
        return u;

    for ( int i = maxLog - 1 ; i >= 0 ; i-- )
        if ( dp[u][i] != dp[v][i] )
            u = dp[u][i], v = dp[v][i];

    return dp[u][0];
}
```

Questions?