

# Tree Algorithms

나정휘

<https://justicehui.github.io/>

# 목차

- Euler Tour Technique
- LCA
- Sparse Table

# Euler Tour Technique

# Euler Tour Technique

- 오일러 투어 테크닉

- 오일러 회로(euler tour): 모든 간선을 정확히 한 번 사용해서 시작점으로 돌아오는 사이클

- 트리의 간선을 단방향 2개로 쪼갬 다음, 루트에서 시작하는 오일러 투어를 구해보자.

- 1 2 5 8 8 5 9 9 5 A A 5 2 6 6 2 1 3 3 1 4 7 7 4 1

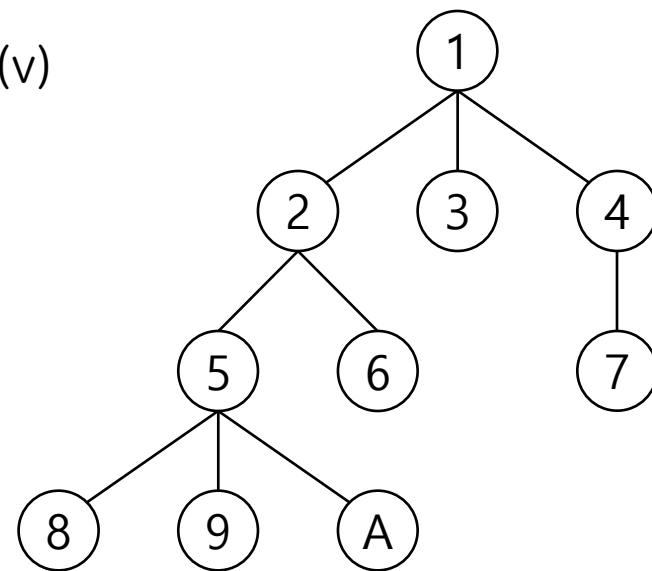
- 정점  $v$ 을 처음 방문하는 시간  $in(v)$ , 마지막으로 방문하는 시간  $out(v)$

- $v$ 의 자손은 항상  $in(v)$ 와  $out(v)$  사이에 존재함

- $in(v)$ 와  $out(v)$  사이에는  $v$ 의 자손만 존재함

- 트리의 서브 트리를 수열의 구간으로 생각할 수 있음

- 누적합, 세그먼트 트리 등의 선형 자료 구조와 결합하기 좋음



# Euler Tour Technique

- BOJ 16404 주식회사 승범이네
  - 1:  $v$ 를 루트로 하는 서브 트리에 속한 정점에  $w$ 를 더하는 쿼리
  - 2:  $v$ 의 값을 구하는 쿼리
- 오일러 투어 테크닉 사용하면
  - 1:  $\text{in}(v)$ 부터  $\text{out}(v)$ 까지  $w$ 를 더하는 쿼리
  - 2:  $\text{in}(v)$ 의 값을 구하는 쿼리
- range update point query
- 세그먼트 트리 or 펜윅 트리 사용해서 해결할 수 있음

# Euler Tour Technique

```

● ● ●

#include <bits/stdc++.h>
using namespace std;
constexpr int SZ = 1 << 17;

int N, Q, In[101010], Out[101010], T[SZ<<1];
vector<int> G[101010];

void Add(int l, int r, int v){
    for(l|=SZ, r|=SZ; l<=r; l>>=1, r>>=1){
        if(l & 1) T[l++] += v;
        if(~r & 1) T[r--] += v;
    }
}

int Get(int x){
    int ret = 0;
    for(x|=SZ; x; x>>=1) ret += T[x];
    return ret;
}

void DFS(int v){
    static int cnt = 0;
    In[v] = ++cnt;
    for(auto i : G[v]) DFS(i);
    Out[v] = cnt;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> Q;
    for(int i=1, p; i<=N; i++) cin >> p, G[max(0, p)].push_back(i);
    DFS(1);
    for(int q=1; q<=Q; q++){
        int op, a, b; cin >> op;
        if(op == 1) cin >> a >> b, Add(In[a], Out[a], b);
        else cin >> a, cout << Get(In[a]) << "\n";
    }
}
```

질문?

# Euler Tour Technique

- BOJ 14287 회사 문화 3
  - 1:  $v$ 와  $v$ 의 모든 조상에  $w$ 를 더하는 쿼리
  - 2:  $v$ 의 값을 구하는 쿼리
- 오일러 투어 테크닉 사용하면
  - 1:  $\text{in}(v)$ 에  $w$ 를 더하는 쿼리
  - 2:  $\text{in}(v)$ 부터  $\text{out}(v)$ 까지의 합을 구하는 쿼리
- point update range query
- 세그먼트 트리 or 펜윅 트리 사용해서 해결할 수 있음



# Euler Tour Technique

- BOJ 25207 바벨탑의 저주
  - 문제 요약
    - 정점마다 정수  $A_i$ 가 적혀 있음
    - 각 정점을 루트로 하는 서브 트리마다, 서브 트리에 속한 모든 정점의 자식 순서를 뒤집었을 때
    - 정점에 적힌  $A_i$ 가 뒤집기 전과 동일한지 판별하는 문제
  - 서브 트리가 팰린드롬인지 판별하는 문제
  - Euler Tour Technique를 사용하면  $in(v) \sim out(v)$ 가 팰린드롬인지 판별하는 문제
    - 팰린드롬 판별은 Manacher algorithm을 사용해도 되고, 해싱을 이용해도 됨
    - 수열에서 단순히 가중치만 저장하면 틀리고, 정점의 깊이까지 함께 저장해야 함

질문?

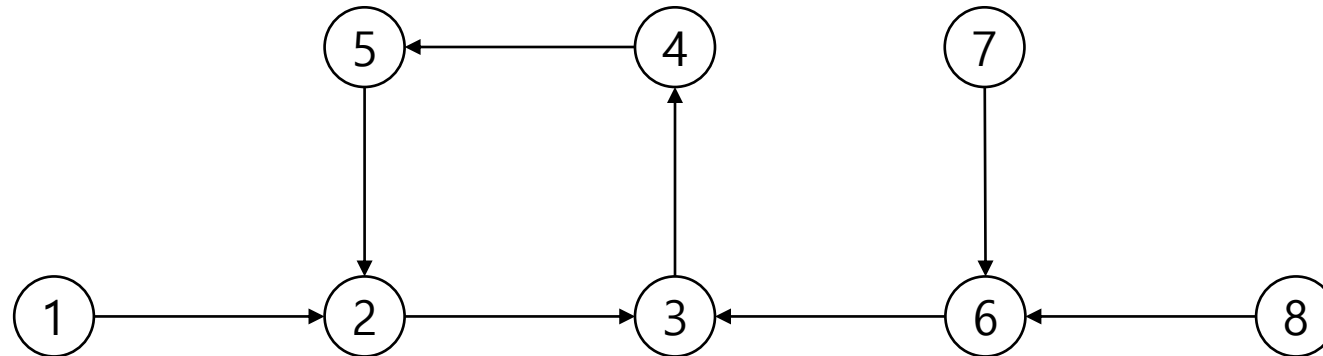
# Sparse Table

# Sparse Table

- BOJ 17435 합성함수와 쿼리
  - 문제 요약
    - 정의역과 치역이 모두  $m$  이하의 자연수인 함수  $f(x)$ 가 주어진다.
    - $f^1(x) = f(x)$ ,  $f^{n+1}(x) = f(f^n(x))$  라고 정의하자.
    - $n, x$ 가 주어지면  $f^n(x)$ 를 빠르게 계산해야 한다.

# Sparse Table

- BOJ 17435 합성함수와 쿼리
  - functional graph
    - $f(x) = y$ 이면  $x \rightarrow y$  간선이 존재하는 방향 그래프
      - 모든 정점의 out degree는 1
      - 각 컴포넌트에는 최대 1개의 사이클 존재
      - 사이클에 속하지 않은 간선은 모두 사이클로 향함
    - $x$ 에  $f$ 를 한 번 적용하는 것은 한 칸 이동하는 것이라고 생각할 수 있음
      - $f^n(x)$ 는  $x$ 에서  $n$ 칸 이동한 위치



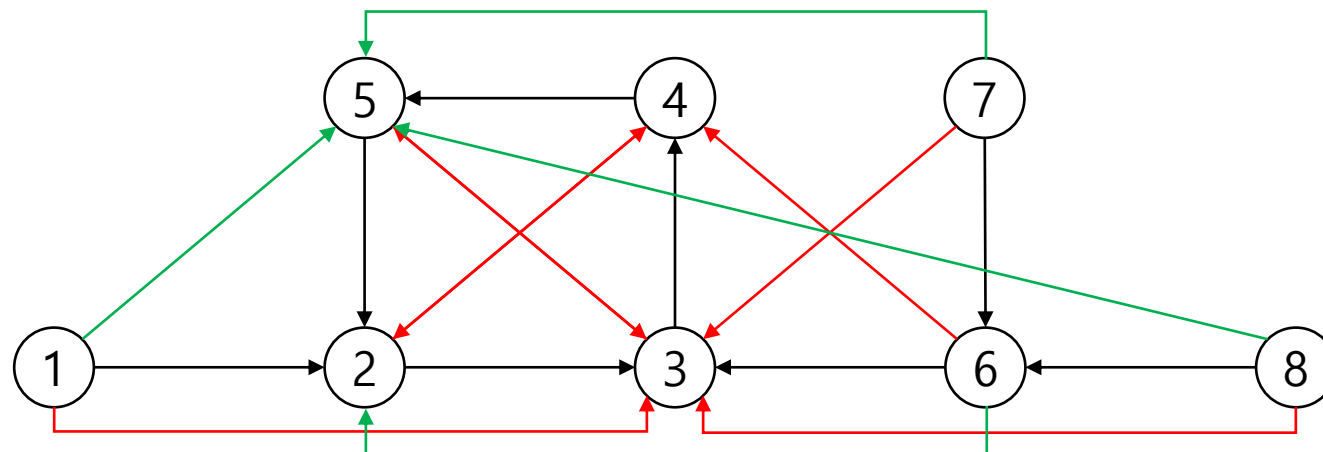
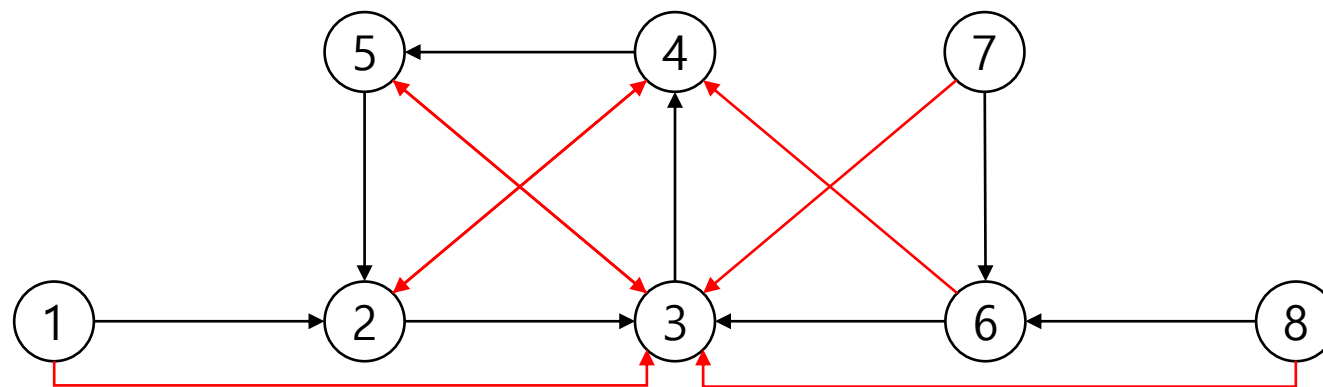
# Sparse Table

- BOJ 17435 합성함수와 쿼리
  - 뜬금 없지만, 거듭제곱을  $O(\log n)$ 에 계산하는 방법을 생각해 보자.
    - $a^{13}$ 는  $a^1, a^4, a^8$ 을 이용해 계산
    - 13을 이진법으로 나타냈을 때 켜진 비트에 해당하는 거듭제곱 수를 활용
      - $13_{(10)} = 1101_{(2)} = 2^3 * 2^2 * 2^0$
  - $f^{2^k}(x)$ 를 모두 전처리해두면 비슷한 방법으로  $O(\log n)$ 에 계산할 수 있다.
    - $x$ 에서 13칸 전진 =  $x$ 에서 8칸 전진, 4칸 전진, 1칸 전진
    - $f^{2^k}(x)$ 를 전처리하는 방법을 알아보자

# Sparse Table

- BOJ 17435 합성함수와 쿼리
  - $P[0][j] = f(j)$ 
    - $j$ 에서  $2^0 = 1$ 번 이동한 결과
  - $P[i][j] = f^{2^i}(j)$ 
    - $j$ 에서  $2^i$ 번 이동한 결과
  - $P[i][j] = P[i-1][P[i-1][j]]$ 
    - $j$ 에서  $2^i$ 번 만큼 이동한 결과
    - $j$ 에서  $2^{i-1}$ 번 이동하고, 다시  $2^{i-1}$ 번 이동한 결과

# Sparse Table





# Sparse Table

```
● ● ●

#include <bits/stdc++.h>
using namespace std;

int N, Q, P[22][202020];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1; i<=N; i++) cin >> P[0][i];
    for(int i=1; i<22; i++) for(int j=1; j<=N; j++) P[i][j] = P[i-1][P[i-1][j]];
    cin >> Q;
    for(int q=1; q<=Q; q++){
        int n, x; cin >> n >> x;
        for(int i=0; n; i++, n>>=1) if(n & 1) x = P[i][x];
        cout << x << "\n";
    }
}
```

# Sparse Table

- BOJ 17435 합성함수와 쿼리
  - 전처리:  $O(N \log K)$
  - 쿼리:  $O(\log K)$
  - $N$  = 수의 범위,  $K$  =  $f$ 를 적용하는 횟수

질문?

# Sparse Table

- BOJ 10868 최솟값
  - 문제 요약
    - 업데이트가 없는 배열에서 Range Minimum Query
  - $P[i][j] = [j, j+2^i)$ 의 최솟값을 전처리하면  $O(\log N)$ 에 RMQ를 할 수 있음
  - 더 빠르게 할 수 있을까?

# Sprase Table

- BOJ 10868 최솟값
  - min 연산과 덧셈 연산의 차이: min은 같은 값에 여러 번 적용해도 결과가 바뀌지 않음
    - $\min(3, 3) = \min(3, 3, 3) = 3$
    - $3 + 3 \neq 3 + 3 + 3$
- $O(\log N)$ 개의 구간을 선택하는 이유
  - 겹치는 부분이 없도록 Sparse Table에 접근하기 위해서
  - 겹치는 것을 허용한다면?
    - s에서 시작해서 e를 넘지 않는 가장 긴 구간
    - e에서 끝나면서 s를 넘지 않는 가장 긴 구간
    - let  $k = \text{floor}(\log_2(e-s+1))$
    - $[s, s+2^k-1], [e-2^k+1, e]$
    - $O(1)$ 에 가능

# Sprase Table

```
● ● ●

#include <bits/stdc++.h>
using namespace std;

int N, Q, P[22][101010];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> Q;
    for(int i=1; i<=N; i++) cin >> P[0][i];
    for(int i=1; i<22; i++){
        for(int j=1; j<=N; j++){
            if(j+(1<<i)-1 <= N) P[i][j] = min(P[i-1][j], P[i-1][j+(1<<(i-1))]);
        }
    }
    for(int q=1; q<=Q; q++){
        int s, e; cin >> s >> e;
        int k = __lg(e-s+1);
        cout << min(P[k][s], P[k][e-(1<<k)+1]) << "\n";
    }
}
```

질문?

LCA



# LCA

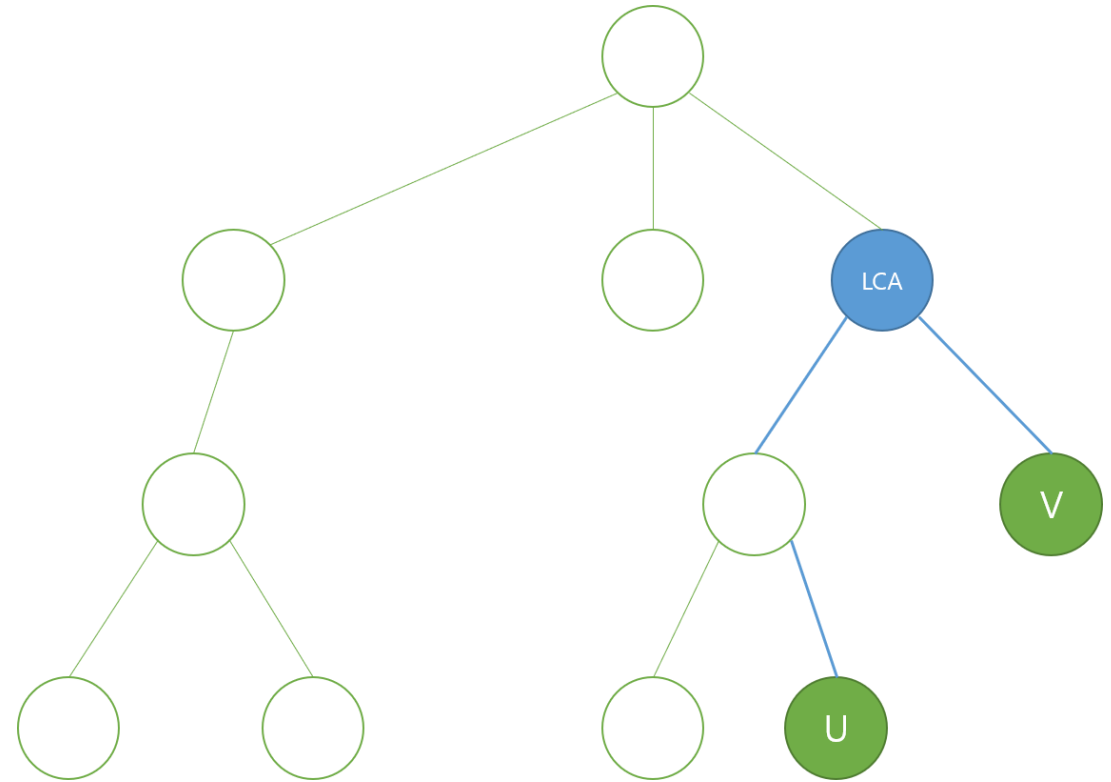
- LCA

- Lowest Common Ancestor

- Lowest : 가장 낮은
    - Common : 공통
    - Ancestor : 조상

- $LCA(u, v)$

- 트리의 정점  $u, v$ 의 공통되는 조상 중 가장 아래에 있는 정점



# LCA

- LCA
  - LCA를 구하는 방법
    - u와 v의 깊이가 다른 경우, 깊이를 동일하게 맞춘다.
    - u와 v가 동시에 한 칸 씩 올라가면서 처음으로  $u = v$ 가 되는 지점이 LCA
    - 시간 복잡도 :  $O(h) + O(h) = O(h) \Rightarrow O(N)$
  - 더 빠르게 할 수 있을까?

# LCA

- LCA
  - LCA를 효율적으로 구하는 방법
    - 앞에서 소개한 두 가지 작업을 각각  $O(\log N)$ 에 처리함
    - 깊이를 맞추는 작업
      - $v$ 가 더 밑에 있고 깊이의 차이를  $k$ 라고, 하면  $v$ 의  $k$ 번째 조상을 찾으면 됨
      - $P[i][v]$ 를  $v$ 의  $2^i$ 번째 조상이라고 정의하면  $O(\log N)$ 에 가능
  - 동시에 한 칸 씩 올리는 작업
    - Parametric Search를 한다.
    - $u \neq v$ 인 가장 높은 지점을 찾으면, 그 지점의 부모가 LCA가 된다.
      - $k = \lg, \lg-1, \lg-2, \dots, 1, 0$ 을 차례대로 보면서
      - $P[k][u] \neq P[k][v]$ 이면  $u, v$ 를  $P[i][u], P[i][v]$ 로 올림
    - $O(\log N)$ 에 가능

# LCA

```

#include <bits/stdc++.h>
using namespace std;

int N, Q, D[101010], P[22][101010];
vector<int> G[101010];

void DFS(int v, int b=-1){
    for(auto i : G[v]) if(i != b) D[i] = D[v] + 1, P[0][i] = v, DFS(i, v);
}

int LCA(int u, int v){
    if(D[u] < D[v]) swap(u, v);
    int diff = D[u] - D[v];
    for(int i=0; diff; i++, diff>>=1) if(diff & 1) u = P[i][u];
    if(u == v) return u;
    for(int i=21; i>=0; i--) if(P[i][u] != P[i][v]) u = P[i][u], v = P[i][v];
    return P[0][u];
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N;
    for(int i=1,s,e; i<N; i++) cin >> s >> e, G[s].push_back(e), G[e].push_back(s);
    DFS(1);
    for(int i=1; i<22; i++) for(int j=1; j<=N; j++) P[i][j] = P[i-1][P[i-1][j]];
    cin >> Q;
    for(int q=1; q<=Q; q++){
        int u, v; cin >> u >> v;
        cout << LCA(u, v) << "\n";
    }
}
```

질문?

# LCA

- BOJ 3176 도로 네트워크
  - 경로의 최솟값과 최댓값을 구하는 문제
  - 경로를 LCA 기준으로 쪼개면, 어떤 정점에서 조상까지 올라가는 경로 2개로 생각할 수 있음
- $Min[i][v]$  = v부터 출발해서  $2^i$ 개의 간선을 타고 이동하면서 만나는 최솟값
- $Min[i][v] = \min( Min[i-1][v], Min[i-1][P[i-1][v]] )$
- Max도 똑같이 할 수 있음

# LCA

```
void Build(){
    for(int i=1; i<22; i++) for(int j=1; j<=N; j++) {
        P[i][j] = P[i-1][P[i-1][j]];
        A[i][j] = min(A[i-1][j], A[i-1][P[i-1][j]]);
        B[i][j] = max(B[i-1][j], B[i-1][P[i-1][j]]);
    }
}

pair<int,int> Solve(int u, int v){
    if(D[u] < D[v]) swap(u, v);
    int diff = D[u] - D[v], mn = 0x3f3f3f3f, mx = 0xc0c0c0c0;
    for(int i=0; diff; i++, diff>>=1) if(diff & 1) {
        mn = min(mn, A[i][u]);
        mx = max(mx, B[i][u]);
        u = P[i][u];
    }
    if(u == v) return {mn, mx};
    for(int i=21; i>=0; i--) if(P[i][u] != P[i][v]) {
        mn = min({mn, A[i][u], A[i][v]});
        mx = max({mx, B[i][u], B[i][v]});
        u = P[i][u], v = P[i][v];
    }
    mn = min({mn, A[0][u], A[0][v]});
    mx = max({mx, B[0][u], B[0][v]});
    return {mn, mx};
}
```

질문?



# 과제

- 필수

- 16404 주식회사 승범이네
- 14287 회사 문화 3
- 25207 바벨탑의 저주
- 17435 합성함수와 쿼리
- 10868 최솟값
- 11438 LCA 2
- 3176 도로 네트워크

- 심화

- 24889 통행량 조사
- 14589 Line Friends (Large)
- 21607 Polynomial and Easy Que...
- 10169 안전한 비상연락망
- 24942 Jail