Dynamic Programming with Data Structures

2019-2020 Winter
20141574 임지환 (Sogang University)

1. Dynamic Programming on Trees
   - Introduction
   - Diameter of Tree
   - Examples (2533 사회망 서비스)

2. Dynamic Programming using a deque
   - Usage of deque in dynamic programming
   - Examples (15678 연세워터파크, 17365 별다줄)

- 기존의 dynamic programming 유형

$$dp[n] = dp[n-1] + dp[n-2]$$
$$dp[n] = dp[n-1] + dp[n-m]$$
$$dp[n][W] = \max\{dp[n-1][W-w_i] + v_i, dp[n-1][W]\}$$
$$dp[l][r] = \min_{l \le k < r}\{dp[l][k] + dp[k+1][r]\} + C[l][r]$$
$$\dots$$

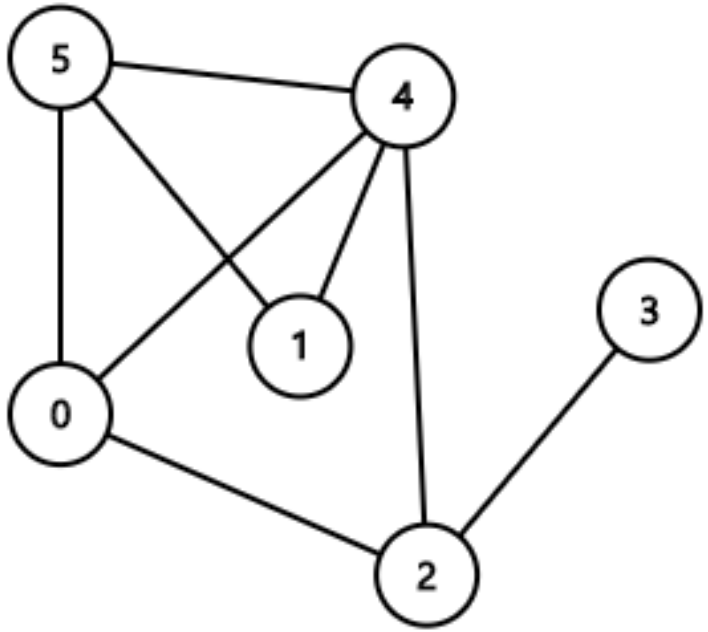
- 기존의 dynamic programming 유형

$$dp[n] = dp[n-1] + dp[n-2]$$
$$dp[n] = dp[n-1] + dp[n-m]$$
$$dp[n][W] = \max\{dp[n-1][W-w_i] + v_i, dp[n-1][W]\}$$
$$dp[l][r] = \min_{l \le k < r}\{dp[l][k] + dp[k+1][r]\} + C[l][r]$$
...

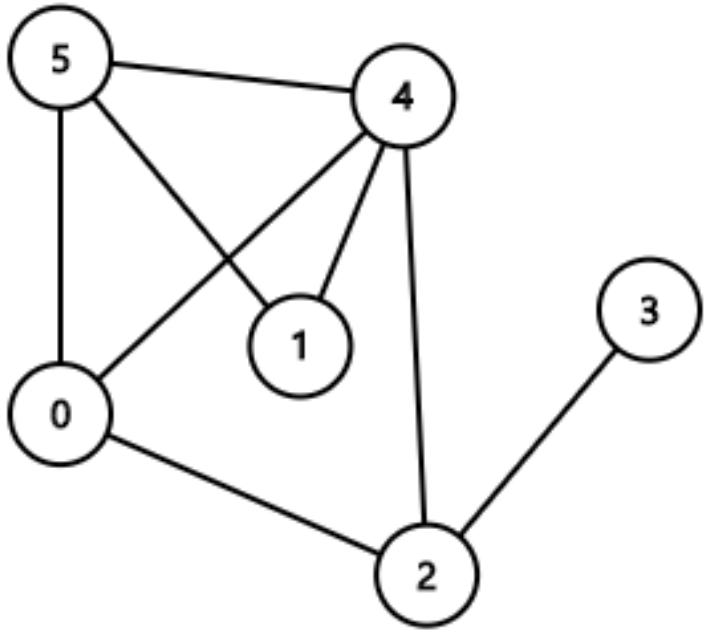
On array

Dynamic Programming On Trees

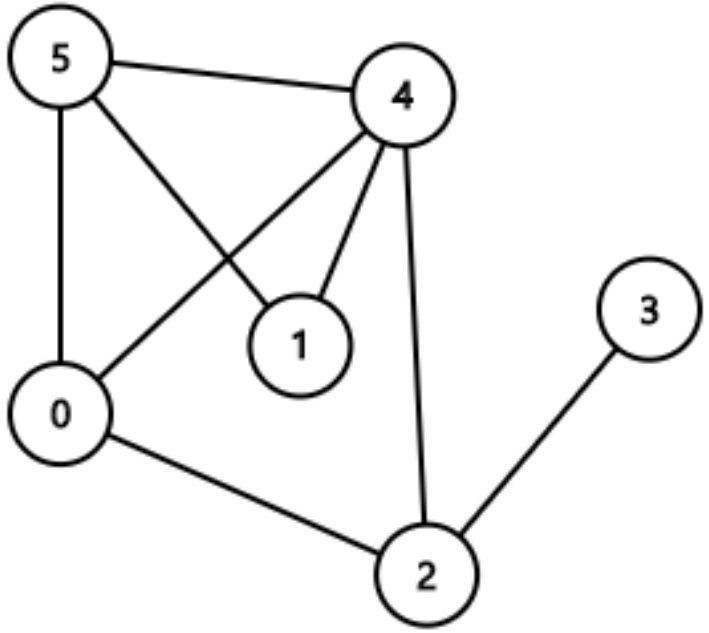
- 그래프에서 다이나믹 프로그래밍을 적용할 수 있을까?

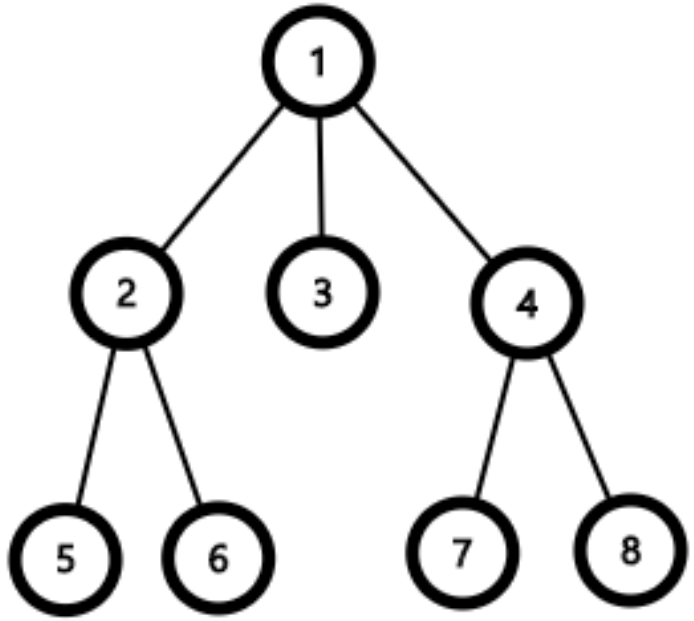


- 그래프에서 다이나믹 프로그래밍을 적용할 수 있을까?
  - 4 affected by 0,1,2, ...
  - 2 affected by 0,4, ...

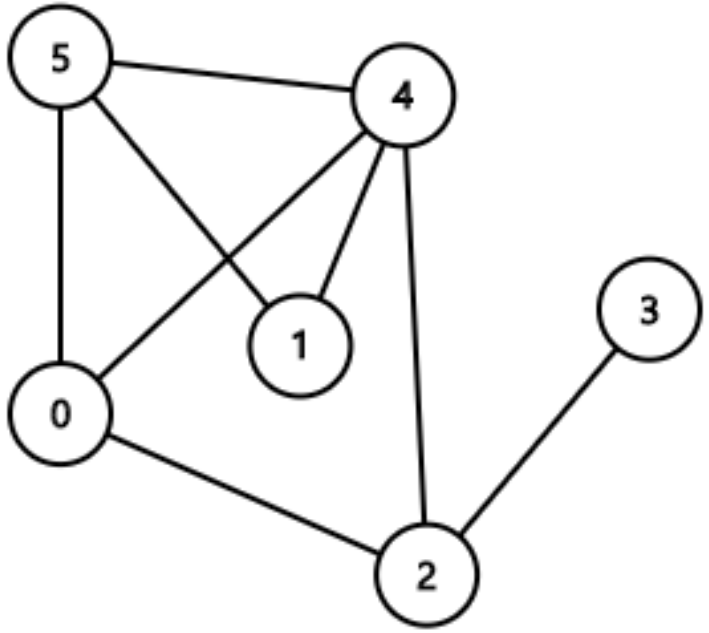


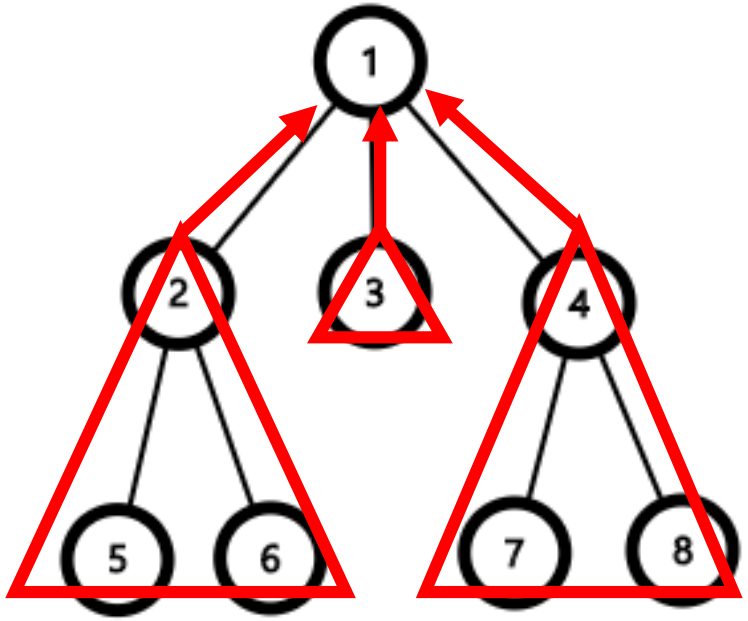
- 트리에서 다이나믹 프로그래밍을 적용할 수 있을까?



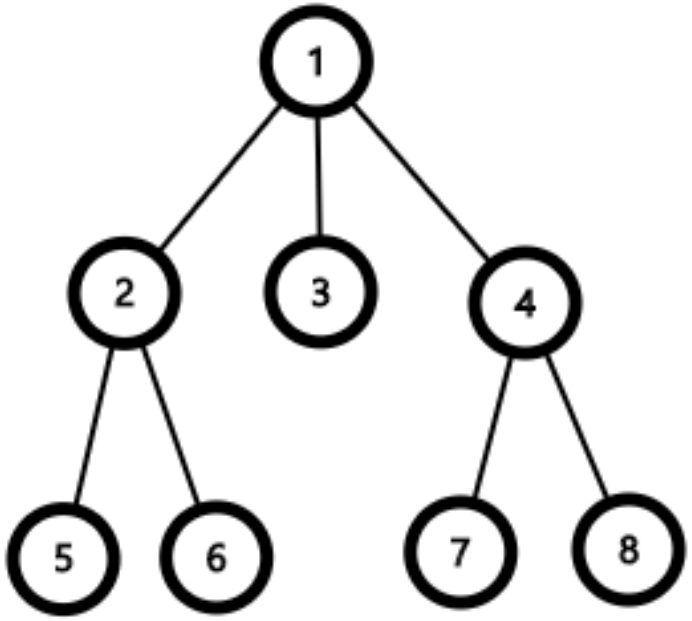


- 트리에서 다이나믹 프로그래밍을 적용할 수 있을까?





- 트리에서 다이나믹 프로그래밍을 적용할 수 있을까?



$$dp[N] = f_{i \in child\ of\ N}(dp[i])$$

- Principle of Tree DP

1. Leaf node의 값을 basis step으로 하여
2. 부모 node의 값을 차례로 구하여
3. 최종적으로 root node의 값 구하기

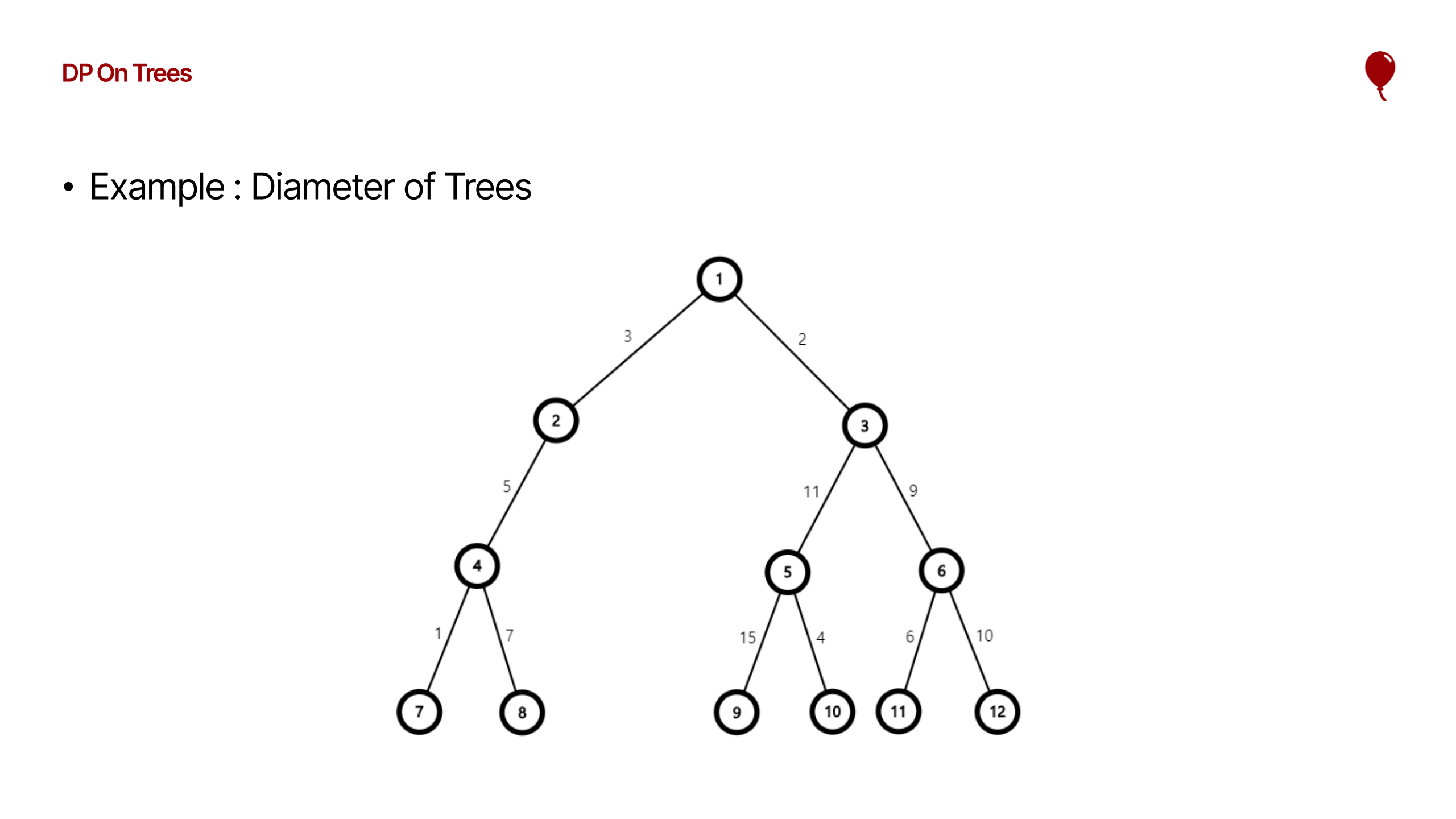
- Basic format of Tree DP

```
11    int dp[mxn];
12    void dfs(int cur, int prev = -1) {
13        for (int &next : adj[cur]) {
14            if (next == prev) continue;
15            dfs(next, cur);
16
17            dp[cur] = func(dp[next]);
18        }
19    }
```

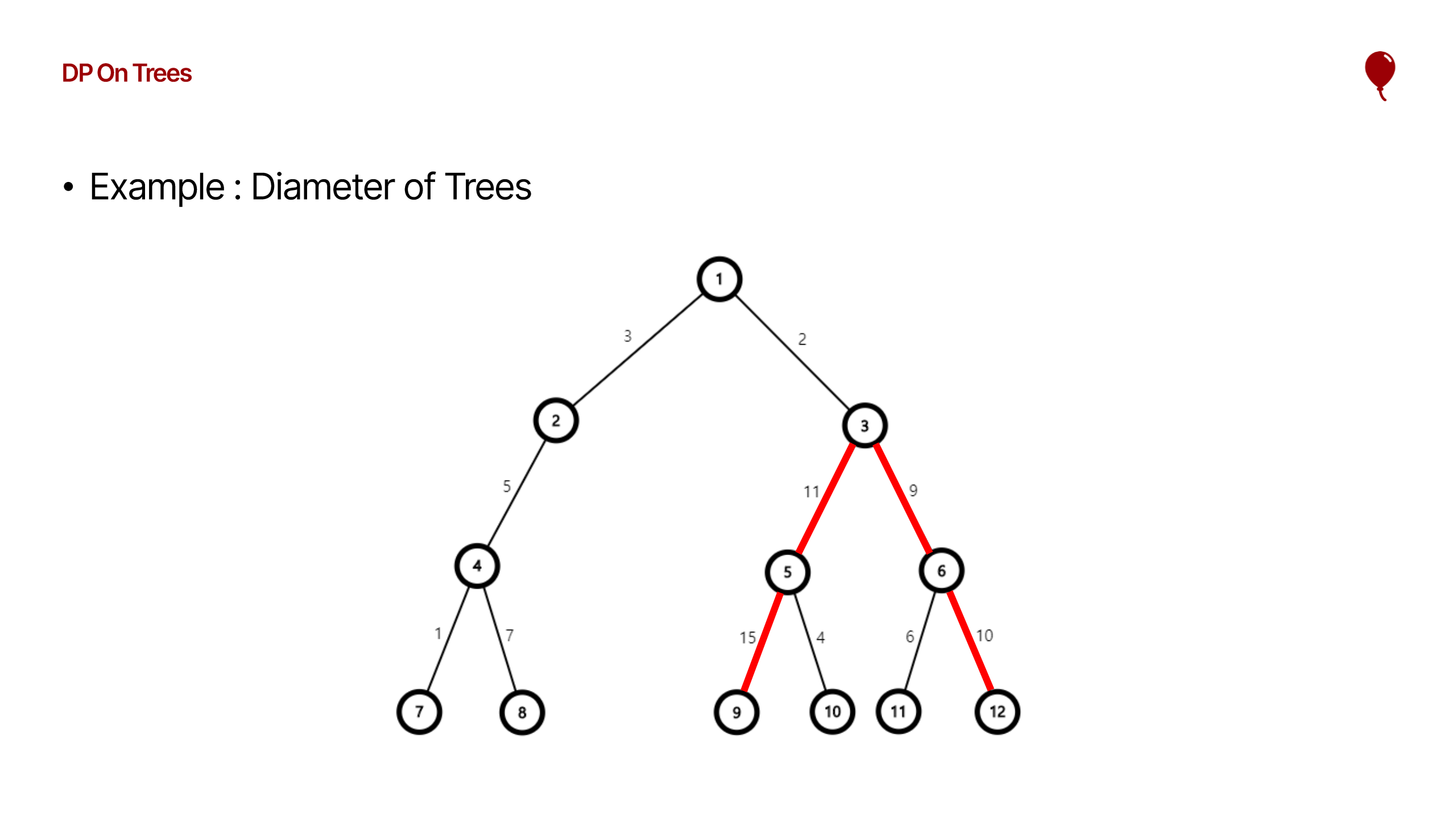
(a)

```
11    int dp[mxn];
12    int solve(int cur, int prev = -1) {
13        int &ret = dp[cur];
14        if (ret != -1) return ret;
15        ret = 0;
16
17        for (int &next : adj[cur]) {
18            if (next == prev) continue;
19
20            ret = func(solve(next, cur));
21        }
22        return ret;
23    }
```

(b)



- Example : Diameter of Trees



- Example : Diameter of Trees

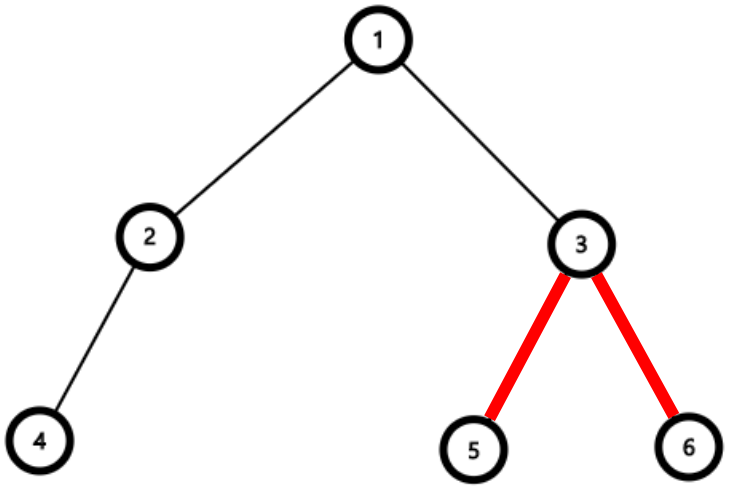
DP On Trees

- Two ways to get diameter of tree
  - Two times of DFS (Link)
  - Dynamic programming

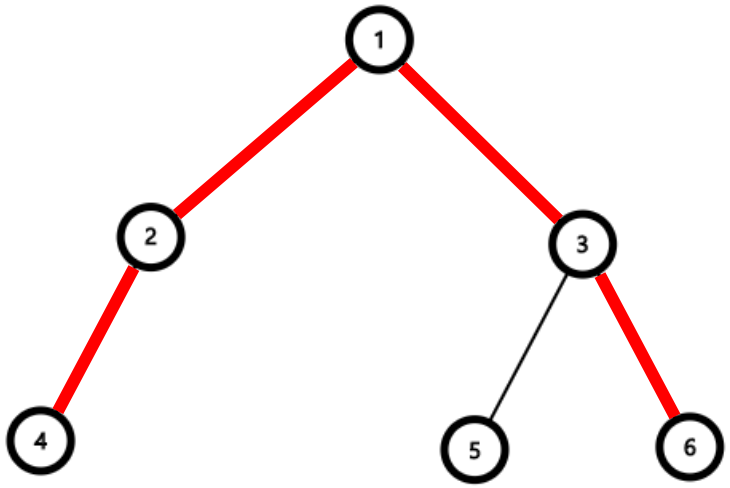
DP On Trees

- Two ways to get diameter of tree
  - Two times of DFS (Link)
  - Dynamic programming

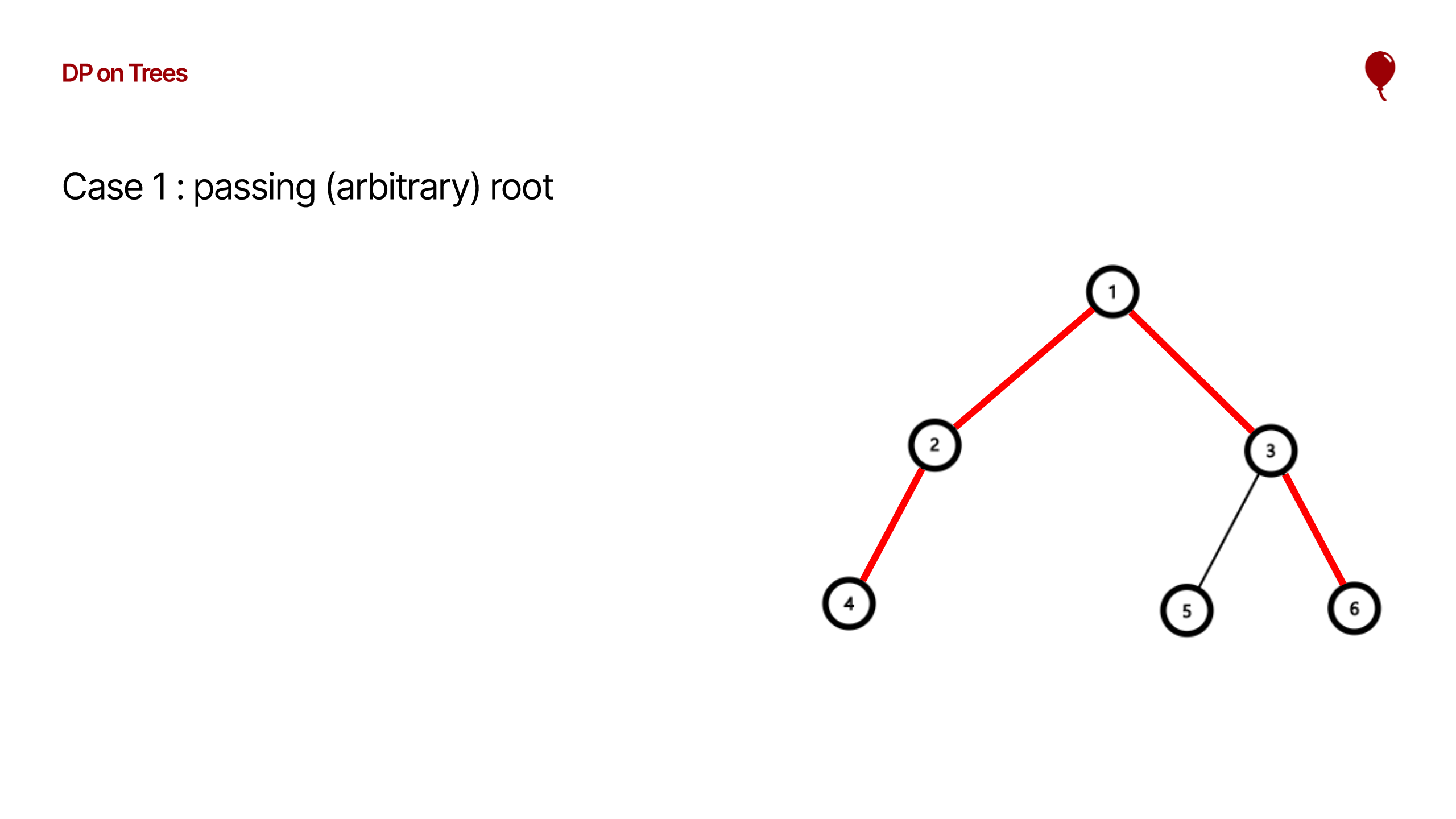
- Example : Diameter of Trees



(a) Diameter without passing (arbitrary) root



(b) Diameter passing (arbitrary) root



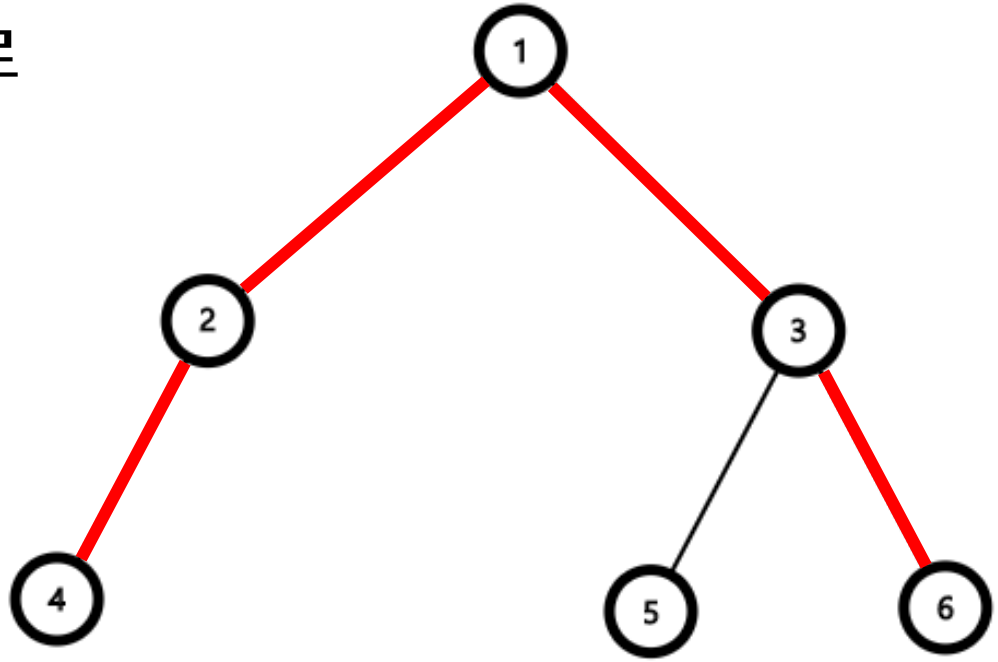
Case 1 : passing (arbitrary) root

Case 1 : passing (arbitrary) root

트리의 지름 = 1에서 leaf로 가는 경로 중 **가장 긴 경로**

+

두번째로 긴 경로

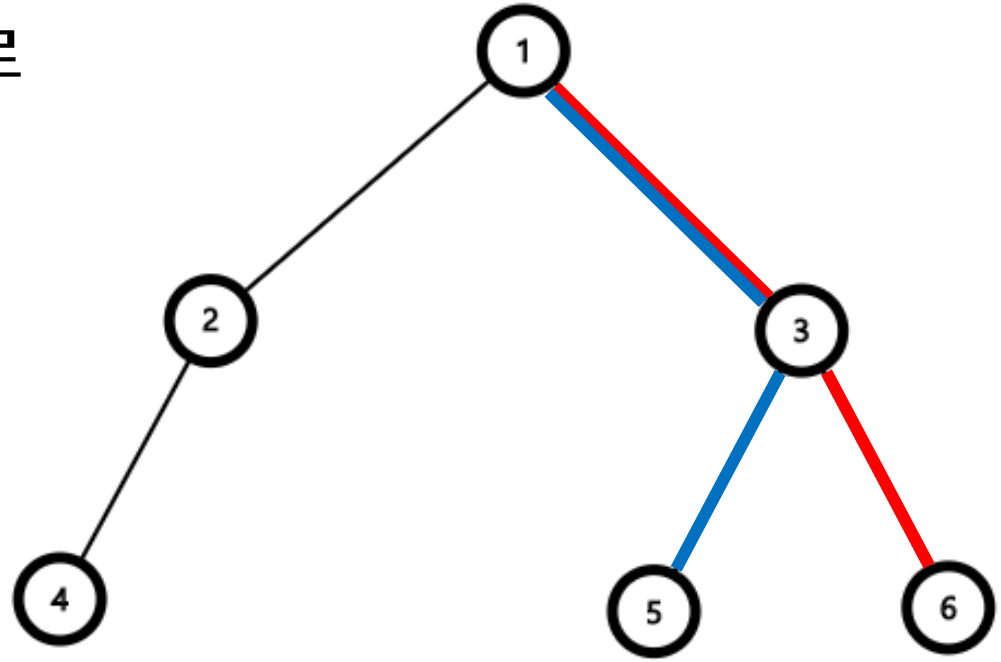


Case 1 : passing (arbitrary) root

트리의 지름 = 1에서 leaf로 가는 경로 중 **가장 긴 경로**

+

두번째로 긴 경로

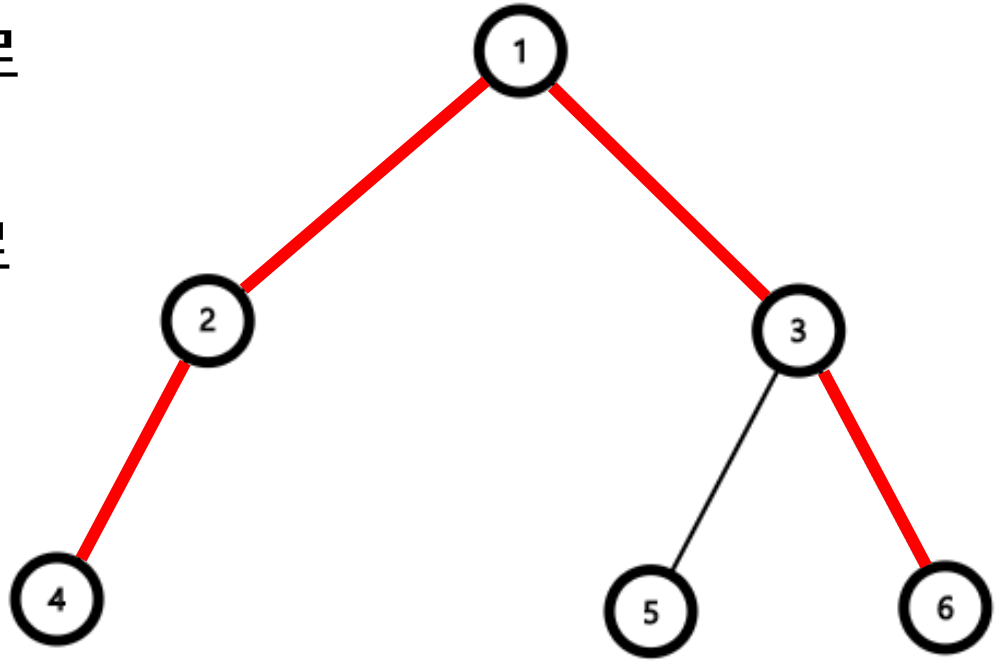


Case 1 : passing (arbitrary) root

트리의 지름 = 1에서 leaf로 가는 경로 중 **가장 긴 경로**

+

다른 child에 대하여 **두번째로 긴 경로**

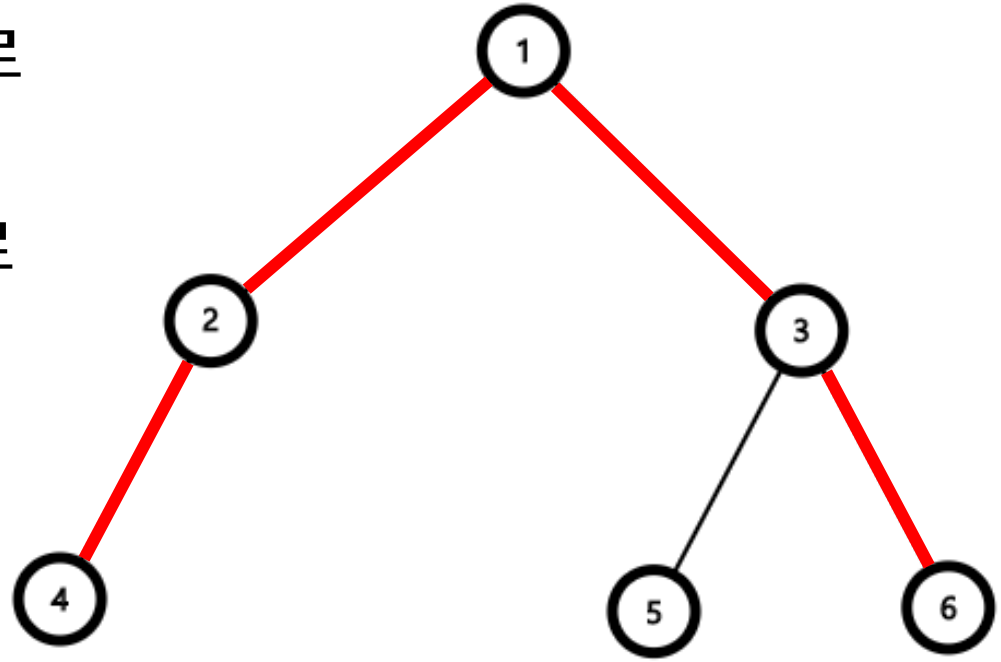


Case 1 : passing (arbitrary) root

트리의 지름 = 1에서 leaf로 가는 경로 중 **가장 긴 경로**

+

다른 child에 대하여 **두번째로 긴 경로**

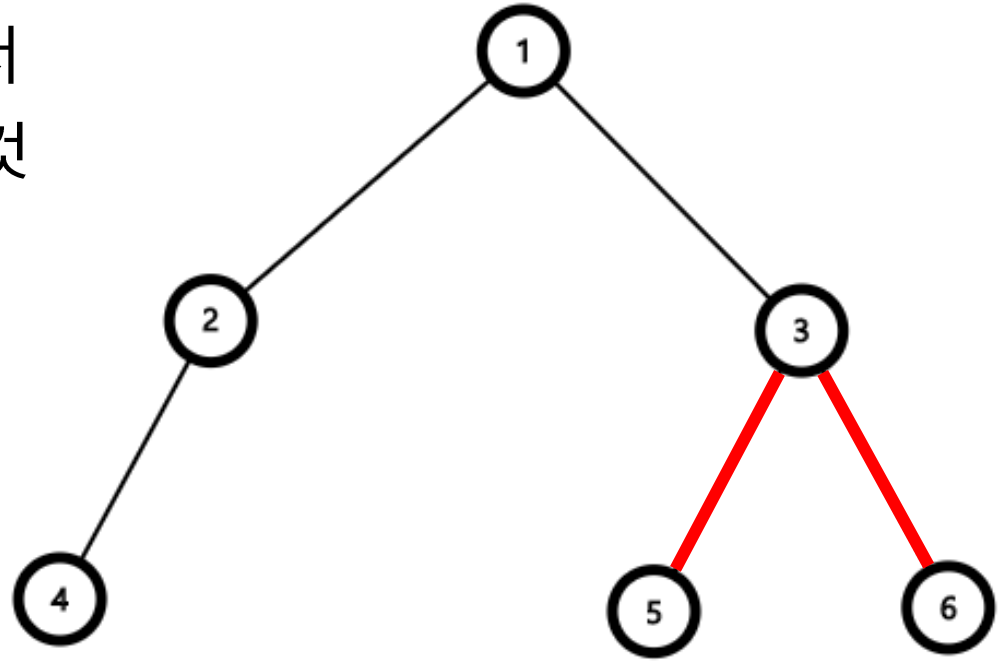


$mx[i]$: i를 root로 하는 subtree에서 leaf로 가는 경로 중 가장 긴 경로

$mx2[i]$: i를 root로 하는 subtree에서 leaf로 가는 경로 중 두번째로 긴 경로

Case 2 : without passing (arbitrary) root

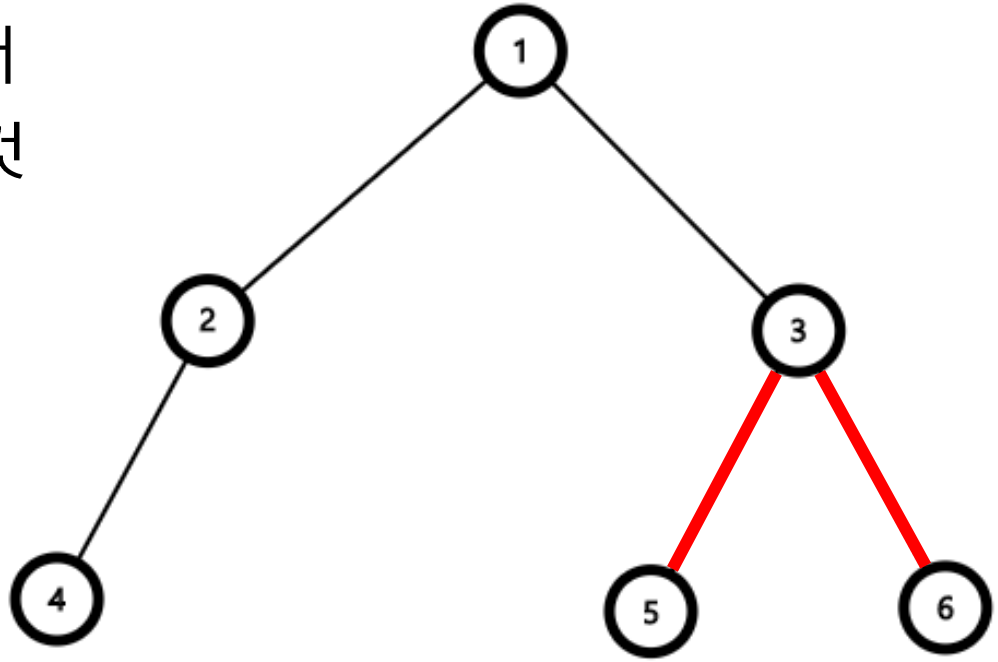
트리의 지름 = 1의 자식들을 root로 하는 subtree에서
나올 수 있는 트리의 지름 중 가장 큰 것



Case 2 : without passing (arbitrary) root

트리의 지름 = 1의 자식들을 root로 하는 subtree에서
나올 수 있는 트리의 지름 중 가장 큰 것

$dp[i]$: i를 root로 하는 subtree의 지름



- Merging Case 1 & Case 2

$$dp[i] = max \begin{cases} mx[i] + mx2[i] \\ dp[j] \ \ (j \in child\ of\ i) \end{cases}$$

$solve(cur)$: cur로부터 leaf까지 가장 긴 경로

```
10      vector<ii>adj[mxn];
11      int dp[mxn];
12      int solve(int cur, int prev = -1) {
13          int mx1 = 0, mx2 = 0, ret = 0;
14          for (auto &it : adj[cur]) {
15              int next = it.first, d = it.second;
16              if (next == prev) continue;
17
18              int cand = solve(next, cur) + d;
19              ret = max(ret, cand);
20              if (mx1 < cand) mx2 = mx1, mx1 = cand;
21              else if (mx2 < cand) mx2 = cand;
22
23              dp[cur] = max(dp[cur], dp[next]);
24              dp[cur] = max(dp[cur], mx1 + mx2);
25          }
26
27          return ret;
28      }
```

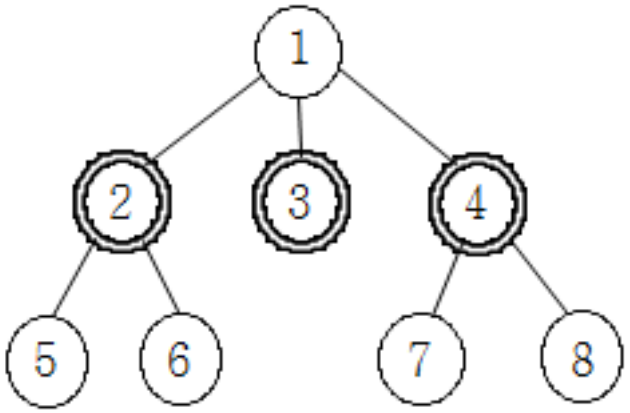
- Another implementation

```
10      vector<ii>adj[mxn];
11      int dp[mxn], mx[mxn];
12      void dfs(int cur, int prev = -1) {
13          int mx1 = 0, mx2 = 0;
14          for (auto &it : adj[cur]) {
15              int next = it.first, w = it.second;
16              if (next == prev) continue;
17
18              dfs(next, cur);
19              if (mx1 < mx[next] + w) {
20                  mx2 = mx1;
21                  mx1 = mx[next] + w;
22              }
23              else if (mx2 < mx[next] + w)
24                  mx2 = mx[next] + w;
25
26              dp[cur] = max(dp[cur], mx1 + mx2);
27              dp[cur] = max(dp[cur], dp[next]);
28              mx[cur] = max(mx[cur], mx1);
29          }
30      }
```

#2533 사회망 서비스(SNS)

- N명($2 \leq N \leq 10^6$)의 친구 관계가 트리 형태로 주어짐
- Early adaptor에 의해 새로운 아이디어가 전파
- Early adaptor가 아닌 사람들은 자신의 모든 친구들이 Early adaptor이어야 함.

- 모든 사람에게 아이디어 전파가 되기 위한 최소 Early Adaptor의 수



Dynamic Programming
Using Deque

Using a deque

- Dynamic programming에서 deque의 사용

Deque이 재귀적인 구조인가?

Using a deque

- Dynamic programming에서 deque의 사용

~~Deque이 재귀적인 구조인가?~~

Sliding Window에서 활용

Using a deque

- Dynamic programming에서 deque의 사용

~~Deque이 재귀적인 구조인가?~~

Sliding Window에서 활용

고정된 구간 내 최대&최소 관리 가능

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

- 문제 해결 과정

1. 구간을 이동하는 과정에서 front를 적절히 pop, back을 적절히 append
   - pop_front : 구간을 벗어날 경우
2. Deque의 내부는 오름차순으로 저장
   - pop_back : 새로 넣기 위한 원소가 오름차순으로 저장되도록

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1		

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1	5	

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1	5	2

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1	2	

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

1	2	

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

2	3	

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

2	3	6

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

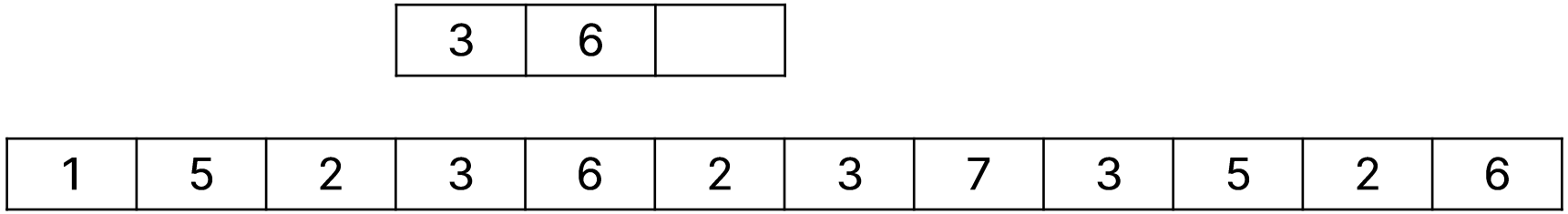
- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

2	3	6

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기



#11003 최솟값 찾기

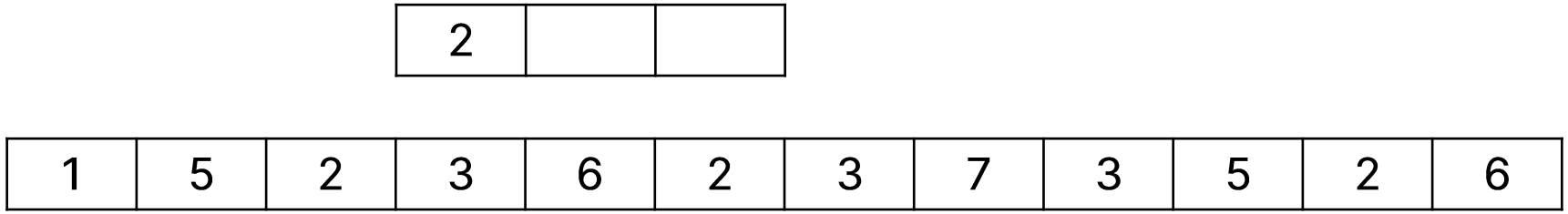
- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기

3	6	2

1	5	2	3	6	2	3	7	3	5	2	6

#11003 최솟값 찾기

- $N(1 \leq N \leq 5 \times 10^6)$개의 수열에서 구간길이가 $L(L \leq N)$인 모든 구간에서의 최솟값 찾기



#11003 최솟값 찾기

```
17      for (int i = 0; i < n; i++) {
18          while (!dq.empty() && l <= i - dq.front()) dq.pop_front();
19          while (!dq.empty() && a[dq.back()] >= a[i])
20              dq.pop_back();
21          dq.push_back(i);
22          cout << a[dq.front()] << ' ';
23      }
```

18 : 길이 L인 구간 관리
19~21 : deque 내 최솟값 관리를 위한 오름차순 순 저장(by index)

#15678 연세워터파크

- $N(2 \leq N \leq 10^5)$개의 징검다리와 $D(1 \leq D \leq N-1)$
- 모든 징검다리는 순서대로 $1 \sim N$의 번호가 있고, 각각 점수$(-10^9 \leq K_i \leq 10^9)$가 부여됨
- 번호의 차이가 D이하인 징검다리로만 이동 가능
- 아무 시작점으로부터 시작하여 아무 위치에서 종료

#Problem set

#1967 트리의 지름

#2533 사회망 서비스(SNS)

#17831 대기업 승범이네

#10273 고대 동굴 탐사

#1814 지붕 색칠하기

#1289 트리의 가중치

#17365 별다줄

#15678 연세워터파크

#5977 Mowing the Lawn

- 4색 정리(Four color theorem)

평면을 유한 개의 부분으로 나누어 각 부분에 색을 칠할 때, 서로 맞닿은 부분을 다른 색으로 칠한다면 4가지 색으로 충분하다.

마찬가지로, 그래프의 색을 칠하는 경우 또한 인접한 노드의 색이 다르게 칠하는 경우 4가지 색으로 coloring이 가능하다.

#17365 별다줄

- 주어진 긴 문자열을 $N(1 \leq N \leq 10^6)$개의 길이 300이하인 단어들의 접두사의 조합으로 만들 때 의 경우의 수

- 길이가 300이하이므로 다음과 같은 점화식을 구상 가능

$$dp[i] = \sum_{j=i-300}^{i-1} dp[j]$$