

Basic Technic

김현정 Acka1357@gmail.com

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- N 개의 나무가 있다.
- 상근이는 총 M 미터의 나무가 필요하다.
- 절단기의 높이 H 를 지정한다.
- 각 나무에서 절단기의 위에 존재하는 부분이 잘린다.
- 총합 M 미터 이상의 나무를 가져가기 위한 절단기 높이의 최대값

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- N개의 나무가 있다.
 - 상근이는 총 M미터의 나무가 필요하다.
 - 절단기의 높이 H를 지정한다.
 - 각 나무에서 절단기의 위에 존재하는 부분이 잘린다.
 - 총합 M미터 이상의 나무를 가져가기 위한 절단기 높이의 최대값
-
- $1 \leq N \leq 1,000,000$
 - $1 \leq M \leq 2,000,000,000$

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- 20, 15, 10, 17 높이의 나무가 있을 때
- 절단기의 높이가 15라면
- 20 => 5
- 17 => 2
- 총 7만큼의 나무를 얻을 수 있다.

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- '절단기에 설정할 수 있는 높이의 최대값'
- 반드시 M만큼의 나무를 가져와야 하는 것은 아니다
- M이상의 나무를 얻을 수 있는 높이 중 최대값

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- 일단 다 해본다.
- $H = \text{Min}(h[i]) \sim \text{Max}(h[i])$
- $\text{sum} = H$ 보다 커서 잘려나가는 부분의 합

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- 일단 다 해본다.
- 시간복잡도: $O(H * N)$
 - $0 < H \leq 1,000,000,000$
 - $1 \leq N \leq 1,000,000$

Parametric Search

최적값을 Binary Search를 응용해 찾아내는 문제 해결 기법

Binary Search

원하는 값이 존재하는지를 판단하기

- 정렬된 N개의 원소를 가진 배열에서
- X라는 값이 있는지 알고 싶다면?

```
bool is_exist(int* arr, int N, int X){  
    // your code  
}
```

Binary Search

원하는 값이 존재하는지를 판단하기

- 배열의 0 ~ N-1번째 원소까지 모두 탐색해본다
- 시간복잡도: $O(N)$

(if $i < j$, $a_i \leq a_j$)

0	1	2	N-3	N-2	N-1
a_0	a_1	a_2					a_{N-3}	a_{N-2}	a_{N-1}



$a_i == X ?$


Binary Search

원하는 값이 존재하는지를 판단하기

- 답이 될 수 있는 범위 $0 \sim N-1$ 중 임의의 값 M 에 대해서
- M 과 찾는 값 X 의 대소관계에 따라
- 범위를 좁혀나간다.

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M \quad ? \quad X$


Binary Search

원하는 값이 존재하는지를 판단하기

- 답이 될 수 있는 범위 $L \sim R$ 중 임의의 값 M 에 대해서
- M 과 찾는 값 X 의 대소관계에 따라
- 범위를 좁혀나간다.

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M \quad ? \quad X$


Binary Search

원하는 값이 존재하는지를 판단하기

- $A[M]$ 이 X 보다 크다면

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M > X$


Binary Search

원하는 값이 존재하는지를 판단하기

- $L \sim M-1$ 범위에 X 가 존재

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M > X$


Binary Search

원하는 값이 존재하는지를 판단하기

- $A[M]$ 이 X 보다 작다면

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M < X$


Binary Search

원하는 값이 존재하는지를 판단하기

- $M+1 \sim R$ 범위에 X 가 존재

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M < X$


Binary Search

원하는 값이 존재하는지를 판단하기

- $A[M]$ 이 X 라면 \leftarrow 정답

(if $i < j$, $a_i \leq a_j$)

0	1	...	L	...	$M-1$	M	$M+1$...	R	...	$N-2$	$N-1$
a_0	a_1		a_L		a_{M-1}	a_M	a_{M+1}		a_R		a_{N-2}	a_{N-1}


 $a_M == X$

Binary Search

원하는 값이 존재하는지를 판단하기

- 이 과정을 $L \leq R$ 이 성립하지 않을 때까지 반복한다.

```
bool is_exist(int* arr, int N, int X){
    int l = 0, r = N-1, m;
    while(l <= r){
        m = (l + r) / 2;
        if(arr[m] > X) r = m - 1;
        else if(arr[m] < X) l = m + 1;
        else return true;
    }
    return false;
}
```

Binary Search

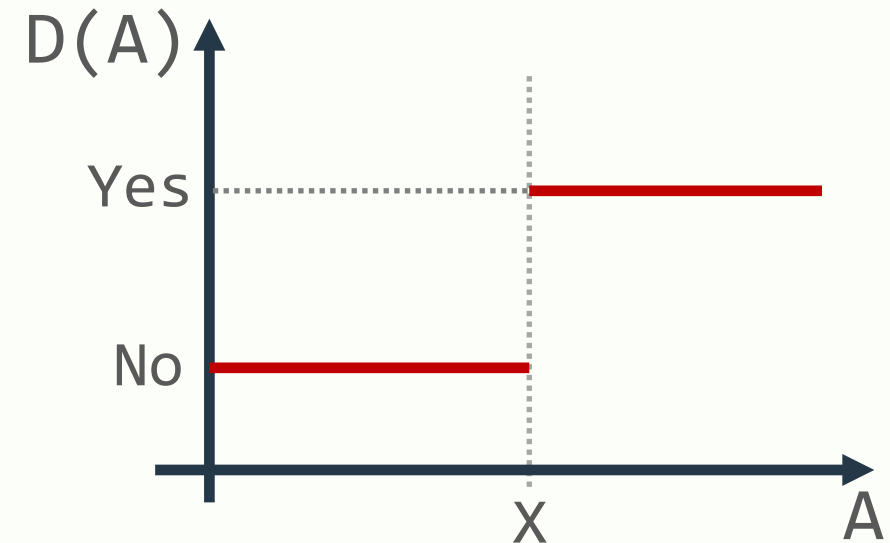
원하는 값이 존재하는지를 판단하기

- $M = (L + R) / 2$ 라고 하면
- 한 번의 조사마다 범위가 반으로 줄어든다.
- 시간복잡도: $O(\log N)$

Parametric Search

Binary Search를 이용해 최적값 구하기

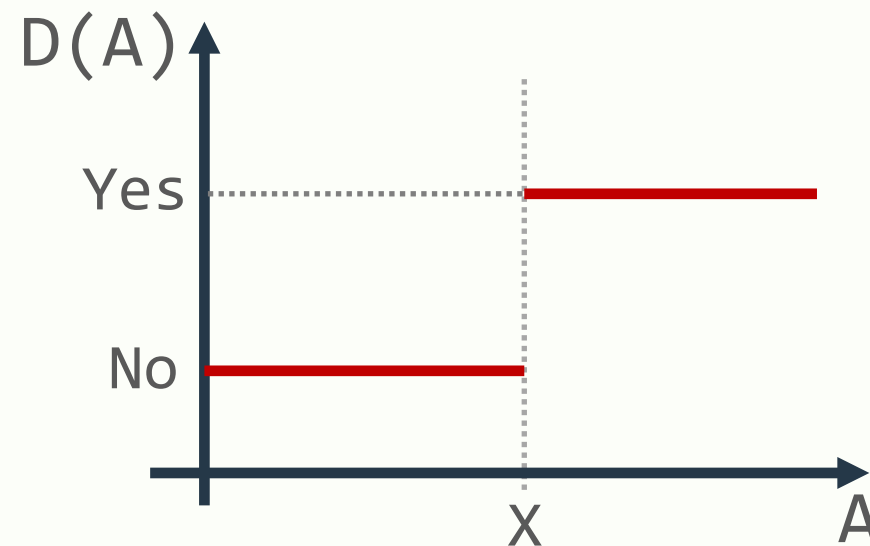
- 최적(최소값/최대값)이 되는 정답 X 를 찾는 문제에서
- X 를 한번에 구할 수는 없지만
- X 의 추정치 A 가 답이 될 수 있는지 판정할 수 있을 때($D(A) = \text{Yes/No}$)
- 이 판단을 반복해 X 를 찾는 방법



Parametric Search

Binary Search를 이용해 최적값 구하기

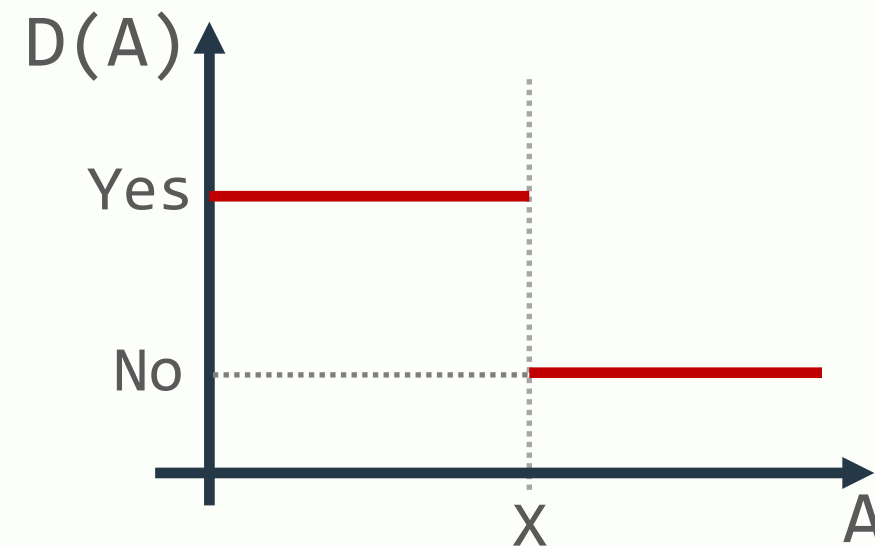
- X 의 추정치 A 를 모두 볼 수도 있지만(Linear Search)
- $D(A)$ 가 X 를 기준으로 나뉘어진다면 Binary Search를 이용할 수 있다.
- 최적해를 구하는 문제가
- Binary Search + YES/NO Problem
- 의 조합으로 변형된다.



Parametric Search

Binary Search를 이용해 최적값 구하기

- $X^2 == K$ 가 되는 X 를 구하여라
- $D(A): A * A \leq K$

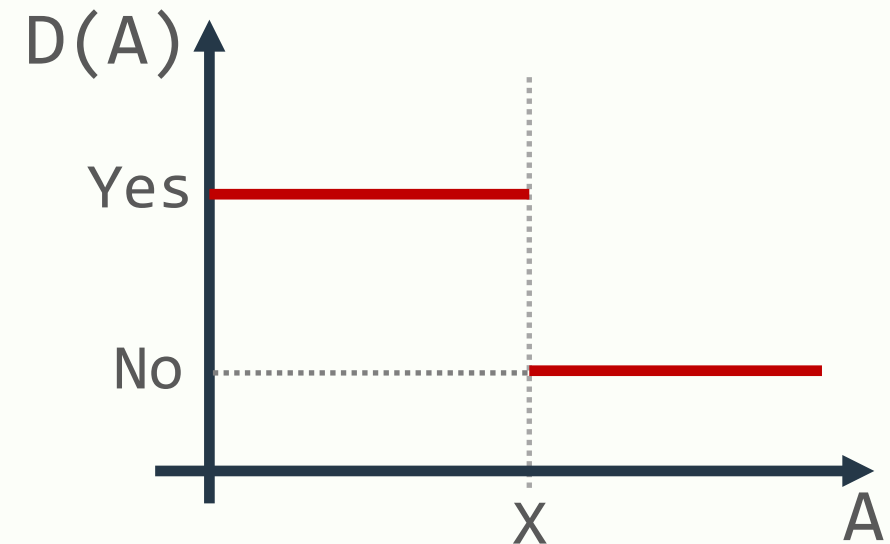


Parametric Search

Binary Search를 이용해 최적값 구하기

- $X^2 == K$ 가 되는 X 를 구하여라
- $D(A): A * A \leq K$ \longrightarrow

```
bool determine(double a, int K){  
    return a * a <= K;  
}
```



Parametric Search

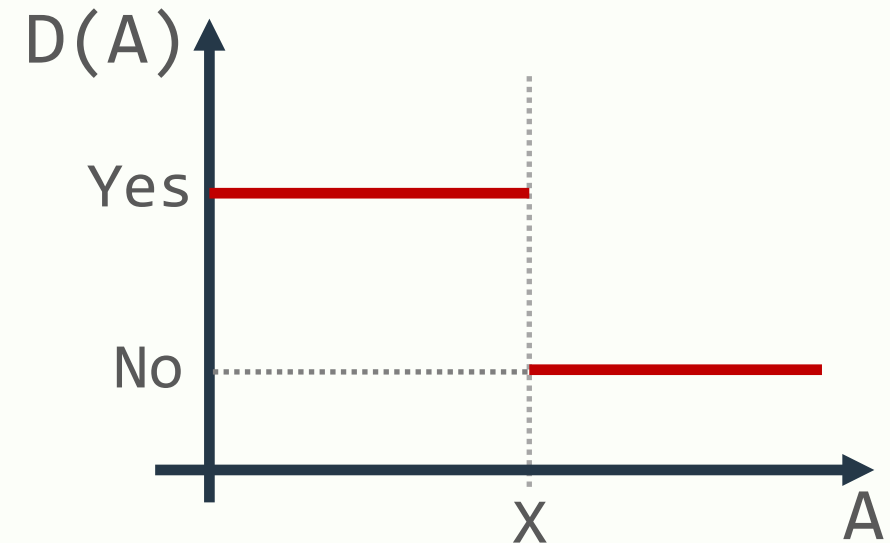
Binary Search를 이용해 최적값 구하기

- $X^2 == K$ 가 되는 X 를 구하여라

- $D(A): A * A \leq K \longrightarrow$

```
bool determine(double a, int K){
    return a * a <= K;
}
```

```
double get_sqrt(int K){
    double l = 0, r = K, m;
    while(r - l < 1e-9){
        m = (l + r) / 2;
        if(determine(m, K)) l = m;
        else r = m;
    }
    return l;
}
```



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- 추정치 H 를 정한다.
- N 개의 나무 중 H 보다 커서 잘려나가는 부분의 합 sum 을 구한다.

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- $sum0$ 이 M 보다 작다면, 더 잘라내야 한다
- \Rightarrow 절단기 높이는 H 보다 낮아져야 한다.

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- $sum0$ 이 M 보다 작다면, 더 잘라내야 한다
- \Rightarrow 절단기 높이는 H 보다 낮아져야 한다.
- $sum0$ 이 M 보다 크거나 같다면, H 는 답의 후보가 된다.
- $\Rightarrow H$ 일때 필요한 나무의 양 이상을 얻을 수 있으므로
- \Rightarrow 이후 보다 높은 H 에서 가능한지를 조사한다.

나무자르기

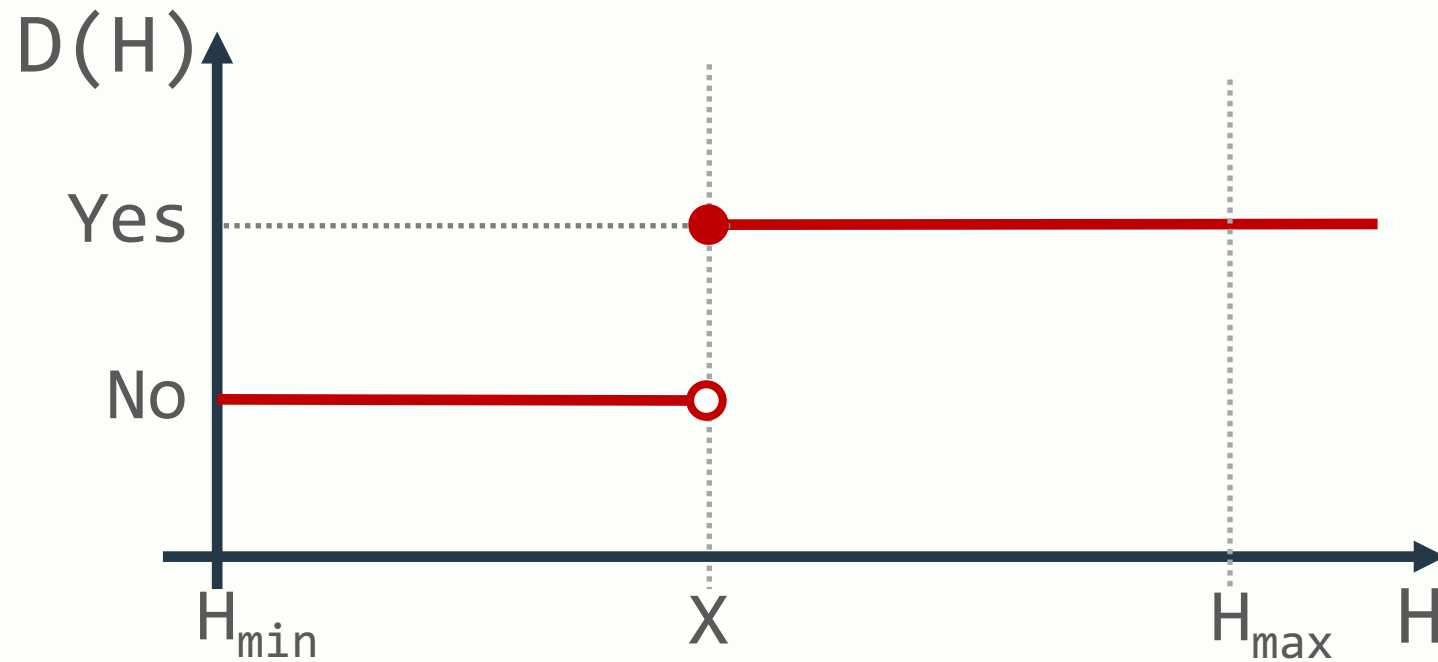
Problem: <https://www.acmicpc.net/problem/2805>

- sum이 M보다 작다면, 더 잘라내야 한다
- => 절단기 높이는 H보다 낮아져야 한다.
- sum이 M보다 크거나 같다면, H는 답의 후보가 된다.
- => H일때 필요한 나무의 양 이상을 얻을 수 있으므로
- => 이후 보다 높은 H에서 가능한지를 조사한다.
- $D(H): \text{sum} \geq M$

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

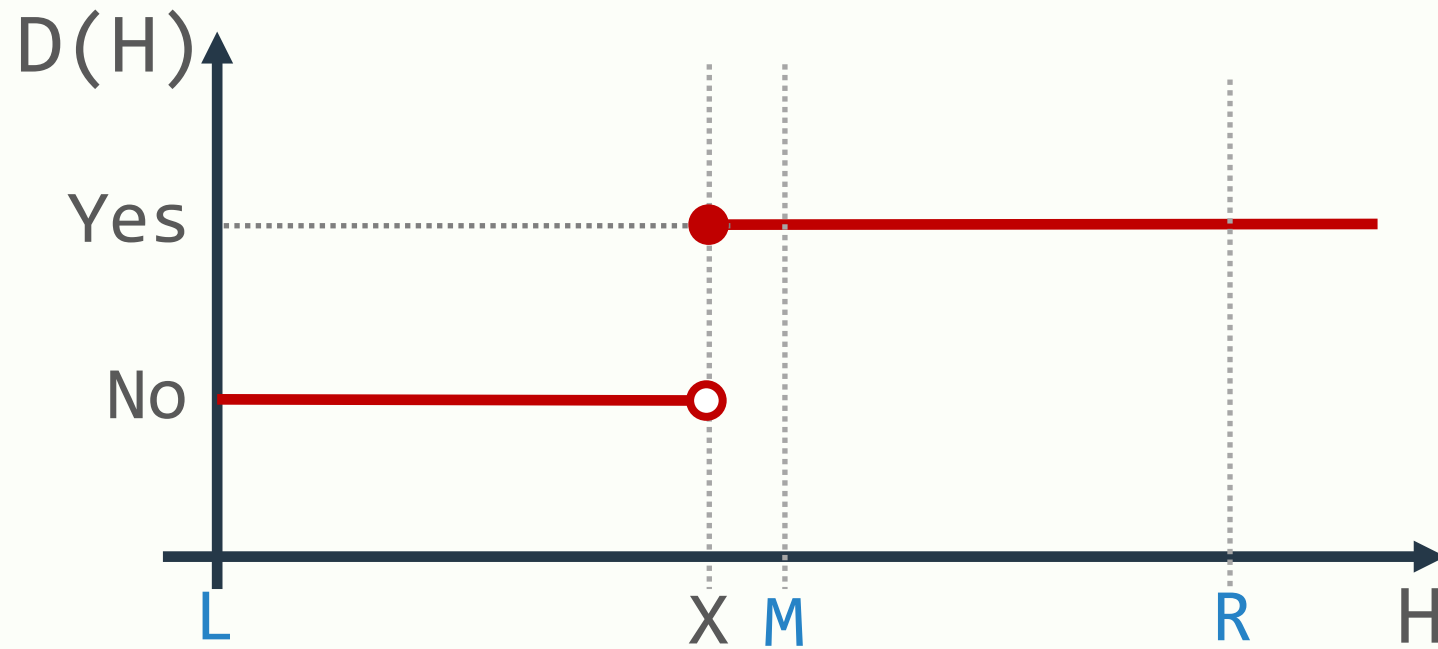
- $D(H)$: $\text{sum} \geq M$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

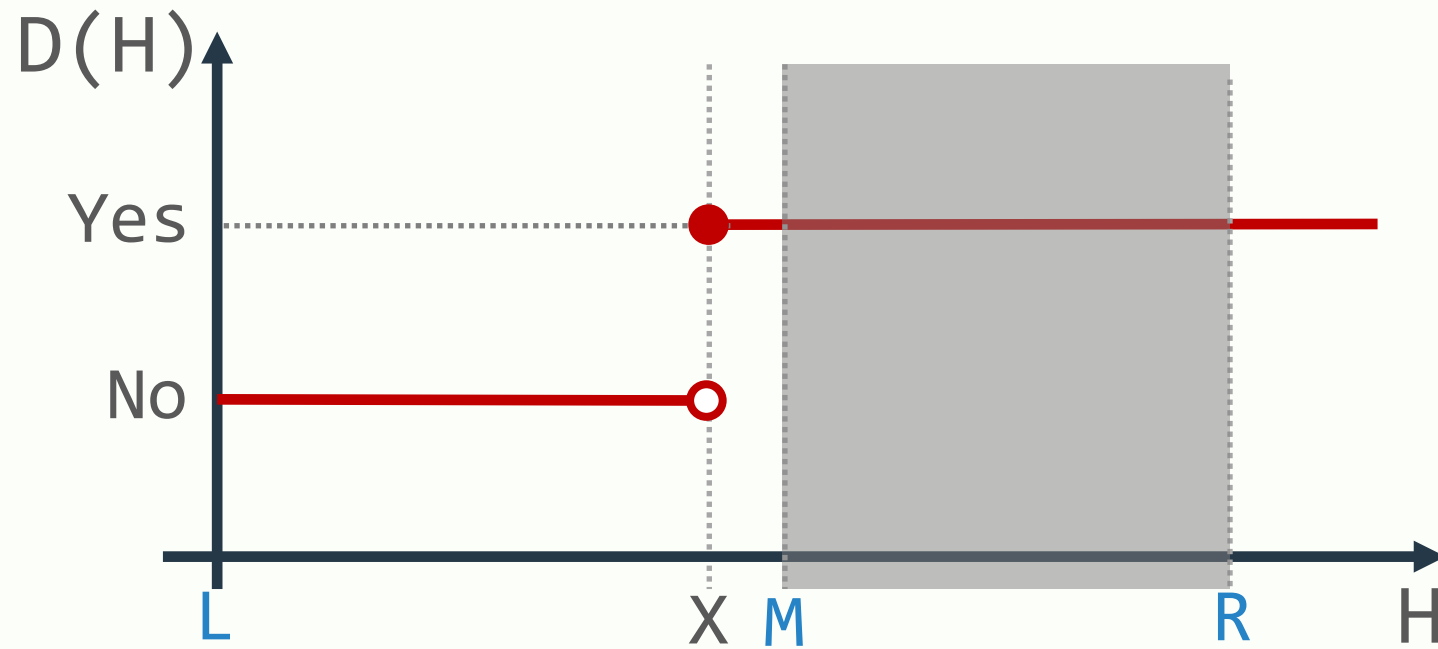
- $D(M) == \text{true}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

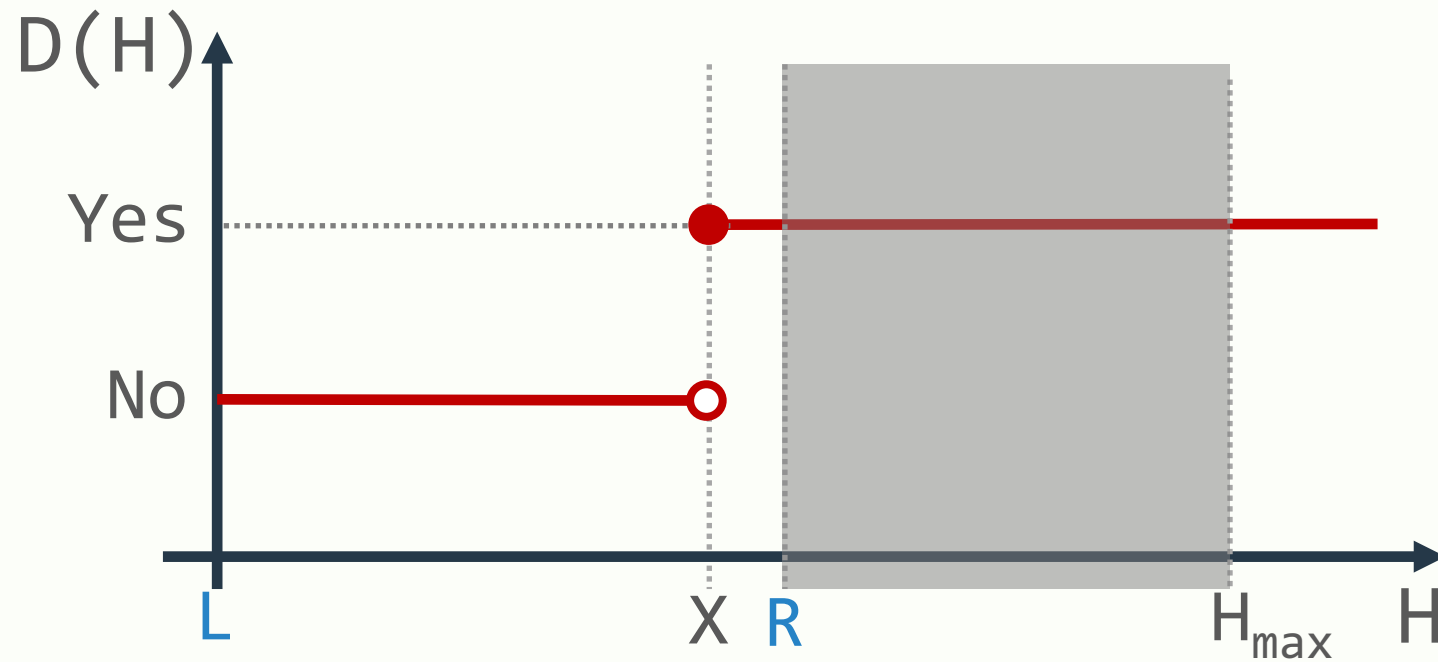
- $D(M) == \text{true}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

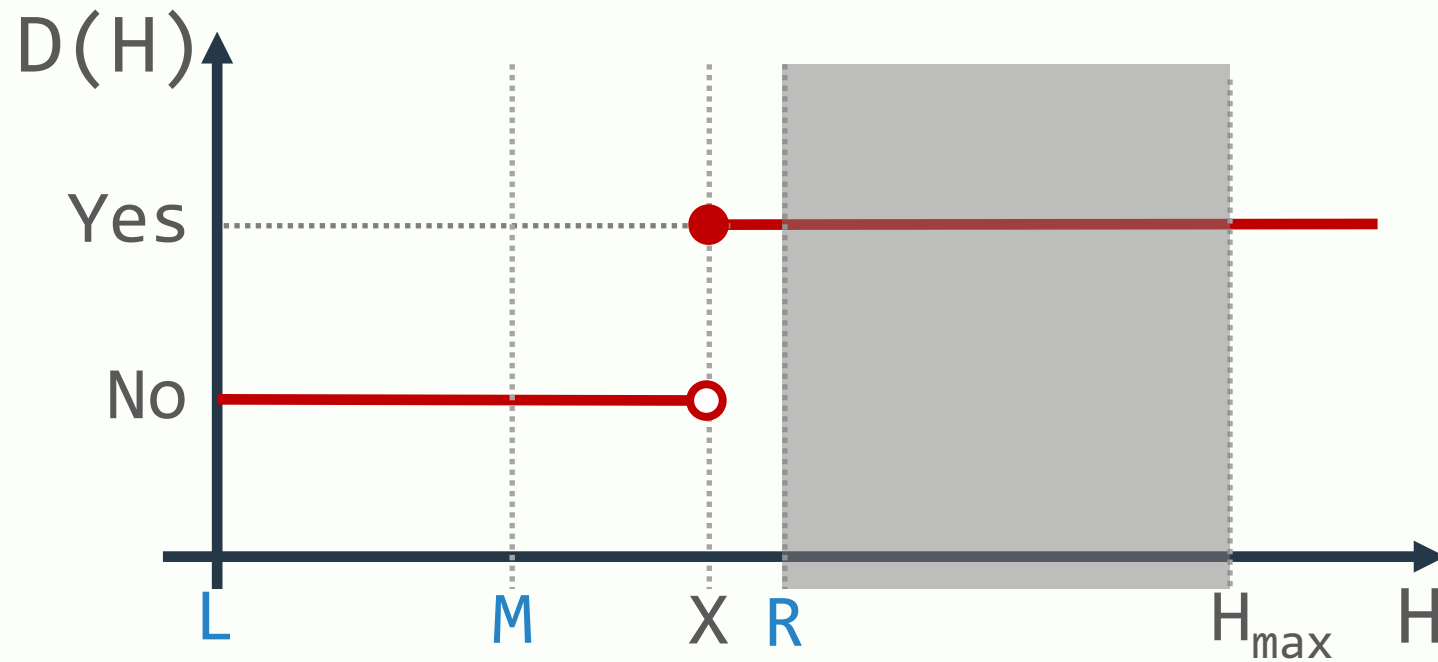
- $D(M) == \text{true}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

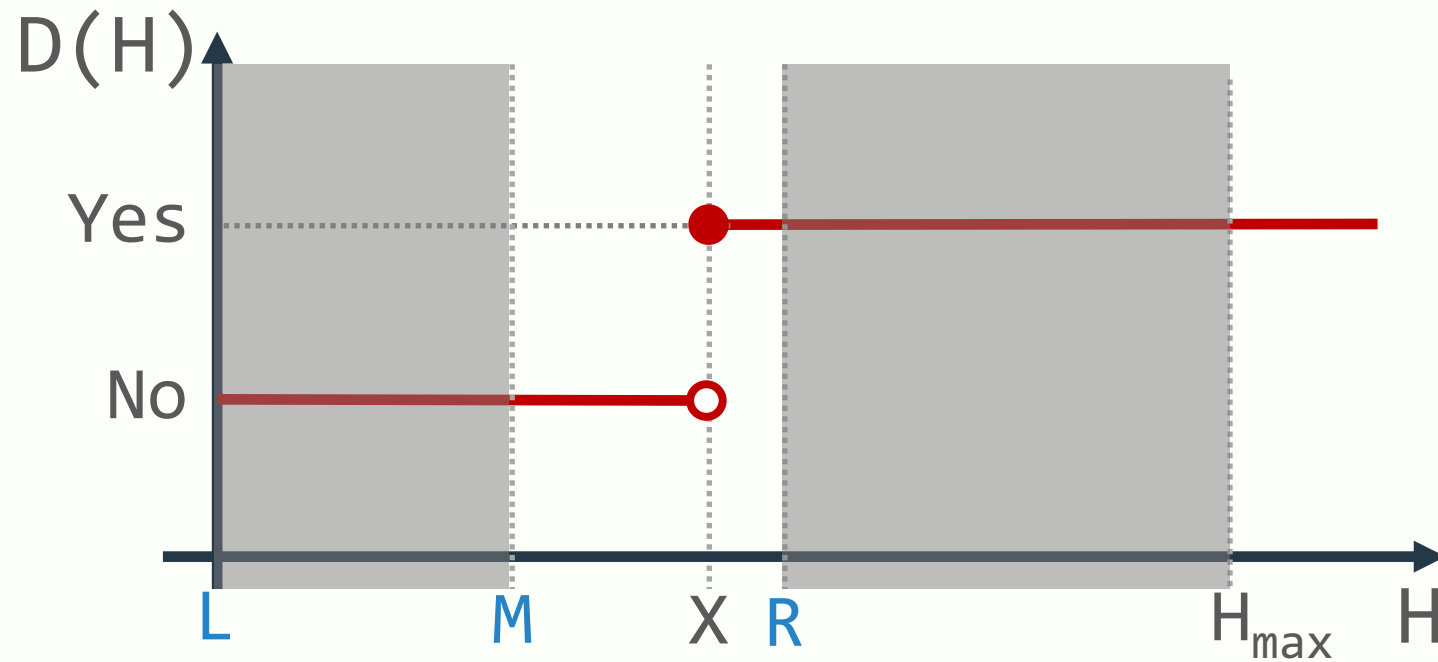
- $D(M) == \text{false}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

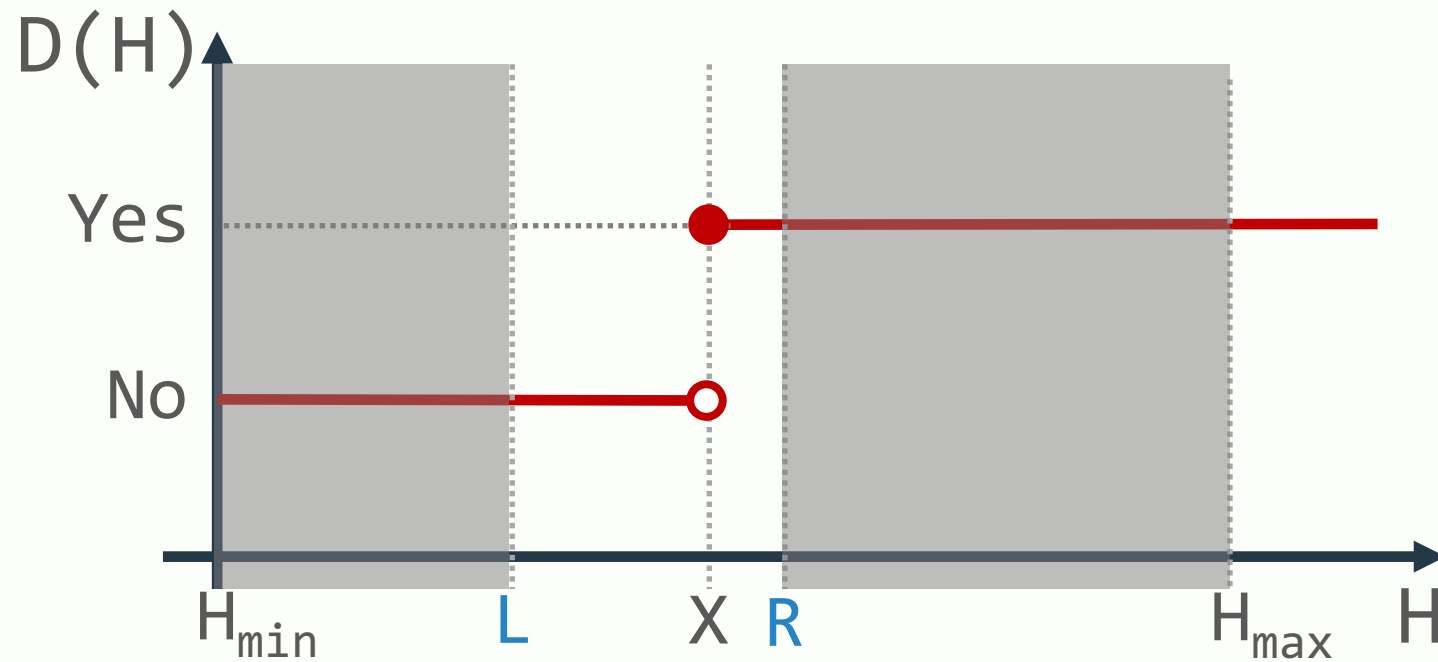
- $D(M) == \text{false}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

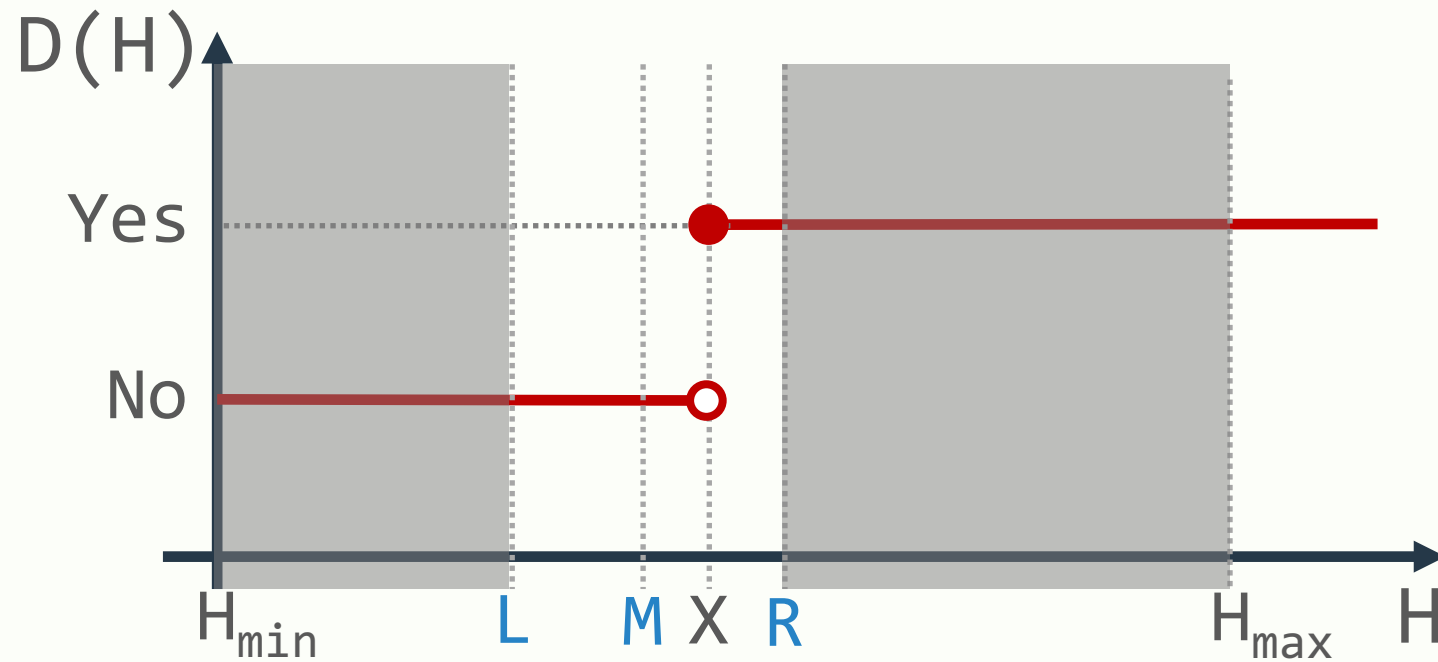
- $D(M) == \text{false}$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- $D(M)$: $\text{sum} \geq M$



나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- $D(M)$: $\text{sum} \geq M$

```
bool is_possible(int* arr, int cuth, int M){  
    long long sum = 0;  
    for(int i = 0; i < N; i++)  
        if(arr[i] > cuth) sum += arr[i] - cuth;  
    return sum >= M  
}
```

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- Parametric Search using $D(H)$

```
int ans = 0;
int l = 0, r = maxh, m;
while(l <= r){
    m = (l + r) / 2;
    if(is_possible(m)){
        ans = m;
        l = m + 1;
    }
    else r = m - 1;
}
```

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- 시간복잡도: $O(\log(H) * N)$
 - $O(\log(H)) * O(D) \Rightarrow O(\log(H)) * O(N)$
 - $0 < H \leq 1,000,000,000$
 - $1 \leq N \leq 1,000,000$

나무자르기

Problem: <https://www.acmicpc.net/problem/2805>

- C/C++:
<https://gist.github.com/Acka1357/b1a98f872d4c44ffb2bcb41e5c1a4497>
- JAVA:
<https://gist.github.com/Acka1357/c4a5f4cd4c865cd908acad4d7a47f4ed>

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $0 \sim N$ 이 각각 $N+1$ 개의 집합으로 존재
- M 개의 0과 1로 구분되는 명령어
- Op0: Merge(Set of X, Set of Y)
- Op1: Is_Same_Set(Element X, Element Y)
- Op1이 수행될 때 마다 YES/NO를 출력

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $1 \leq N \leq 1,000,000$
- $1 \leq M \leq 100,000$

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$ 의 7개의 집합

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$ 의 7개의 집합
- $0\ 1\ 3 \Rightarrow \{0, 3\}, \{1\}, \{2\}, \{4\}, \{5\}, \{6\}, \{7\}$

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$ 의 7개의 집합
- $0\ 1\ 3 \Rightarrow \{0, 3\}, \{1\}, \{2\}, \{4\}, \{5\}, \{6\}, \{7\}$
- $0\ 6\ 7 \Rightarrow \{0, 3\}, \{1\}, \{2\}, \{4\}, \{5\}, \{6, 7\}$
- $0\ 3\ 6 \Rightarrow \{0, 3, 6, 7\}, \{1\}, \{2\}, \{4\}, \{5\}$

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$ 의 7개의 집합
- $0\ 1\ 3 \Rightarrow \{0, 3\}, \{1\}, \{2\}, \{4\}, \{5\}, \{6\}, \{7\}$
- $0\ 6\ 7 \Rightarrow \{0, 3\}, \{1\}, \{2\}, \{4\}, \{5\}, \{6, 7\}$
- $0\ 3\ 6 \Rightarrow \{0, 3, 6, 7\}, \{1\}, \{2\}, \{4\}, \{5\}$
- $1\ 0\ 7 \Rightarrow \text{YES}$
- $1\ 1\ 7 \Rightarrow \text{NO}$

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- 일반적인 방법
- 각 원소가 몇 번 집합에 있는지 기록해두는 배열 $S[N]$
- 각 집합의 원소를 모아둔 $E[N]$ bucket

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- 일반적인 방법
- 각 원소가 몇 번 집합에 있는지 기록해두는 배열 $S[N]$
- 각 집합의 원소를 모아둔 $E[N]$ bucket
- Merge(X, Y): $S[E[S[Y]][i]] = S[X]$ ($0 \leq i \leq E[S[Y]].size$)
- Is_Same_Set(X, Y): $S[X] == S[Y]$

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- Merge: $O(N)$
- Is_Same_Set: $O(1)$
- 시간복잡도: $O(N * M)$
 - $1 \leq N \leq 1,000,000$
 - $1 \leq M \leq 100,000$

Disjoint-set

집합을 관리하는 효율적인 알고리즘

Disjoint-set

집합의 표현

- 그룹을 트리 구조로 관리
- `parent[i]`: i 번 노드의 부모
- `find(int x)`: x 번 노드의 최고 조상(루트)을 찾는 함수
- `link(int x, int y)`: x 노드가 속한 그룹과 y 노드가 속한 그룹을 잇는 함수

Disjoint-set

집합의 표현

- N이 8일 때 초기상태는 아래와 같다.



par

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(1, 2)$



par

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(1, 2)$



par

1	2	3	4	5	6	7	8
1	1	3	4	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(3, 4)$



par

1	2	3	4	5	6	7	8
1	1	3	4	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(3, 4)$



par

1	2	3	4	5	6	7	8
1	1	3	3	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(4, 5)$



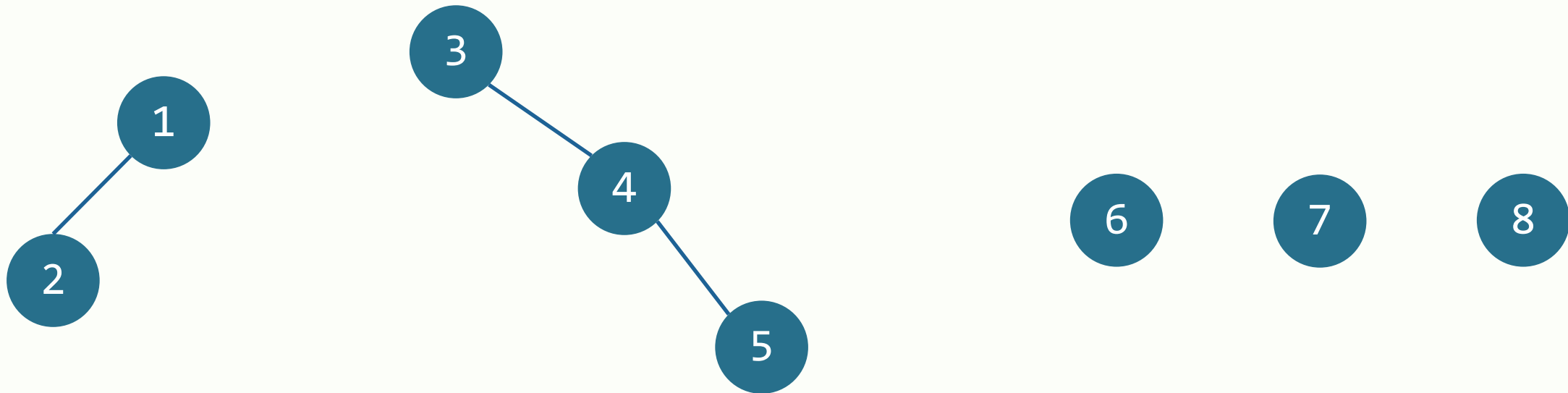
par

1	2	3	4	5	6	7	8
1	1	3	3	5	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(4, 5)$



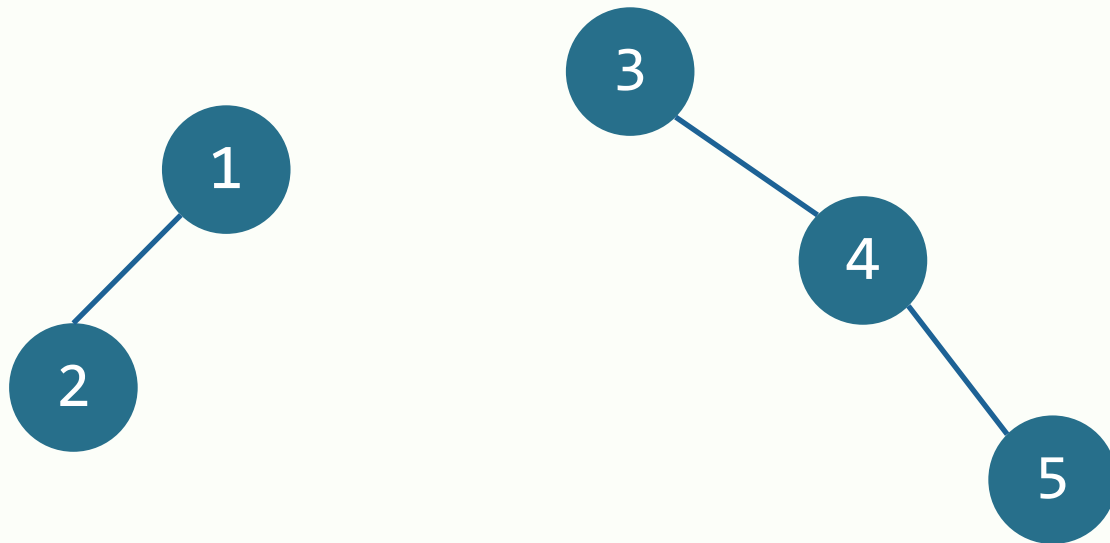
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- link(4, 5)



```
void link(int x, int y){  
    par[y] = x;  
}
```

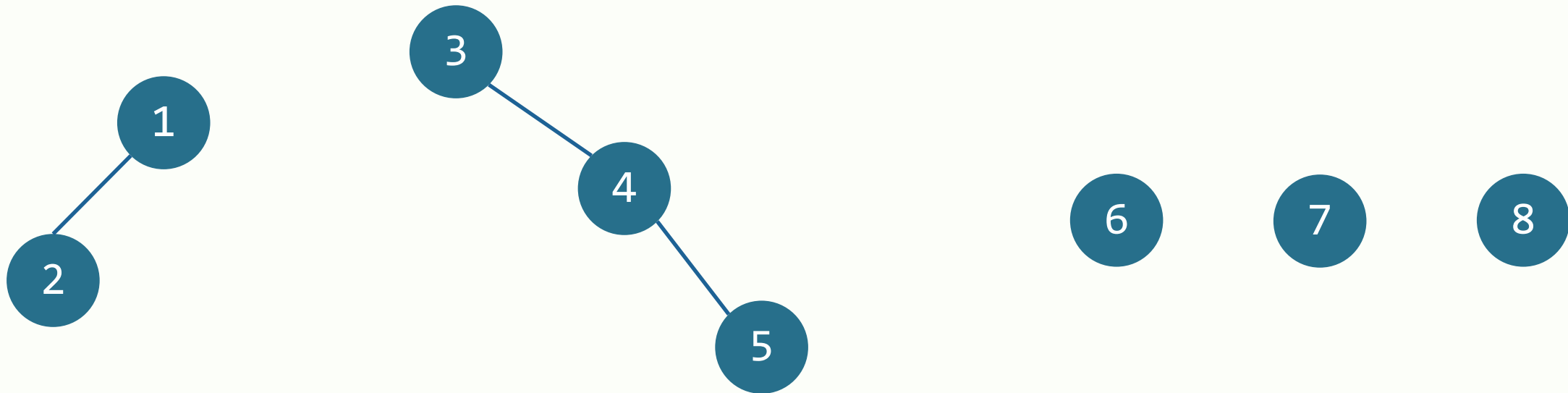
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



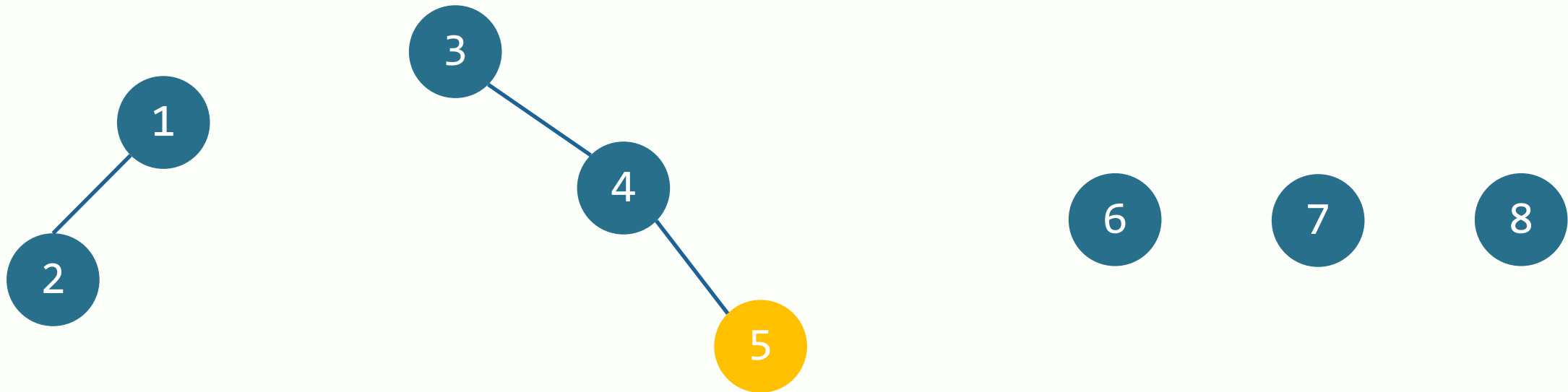
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



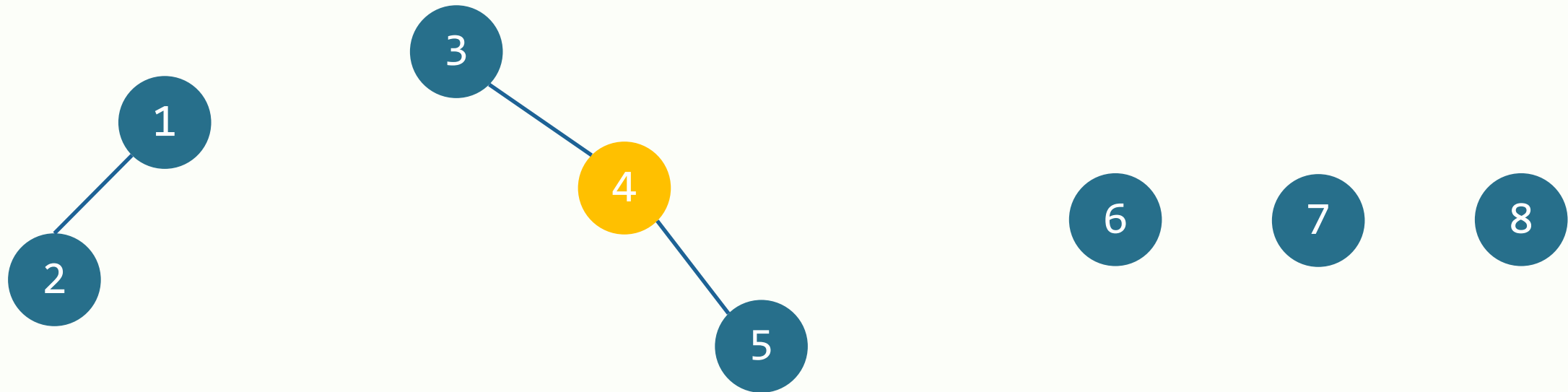
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



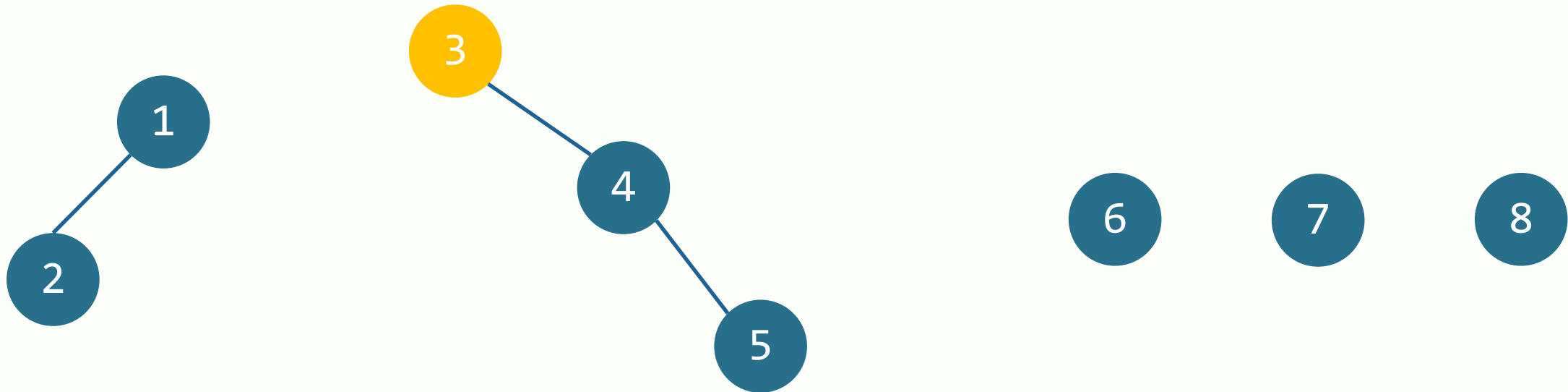
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



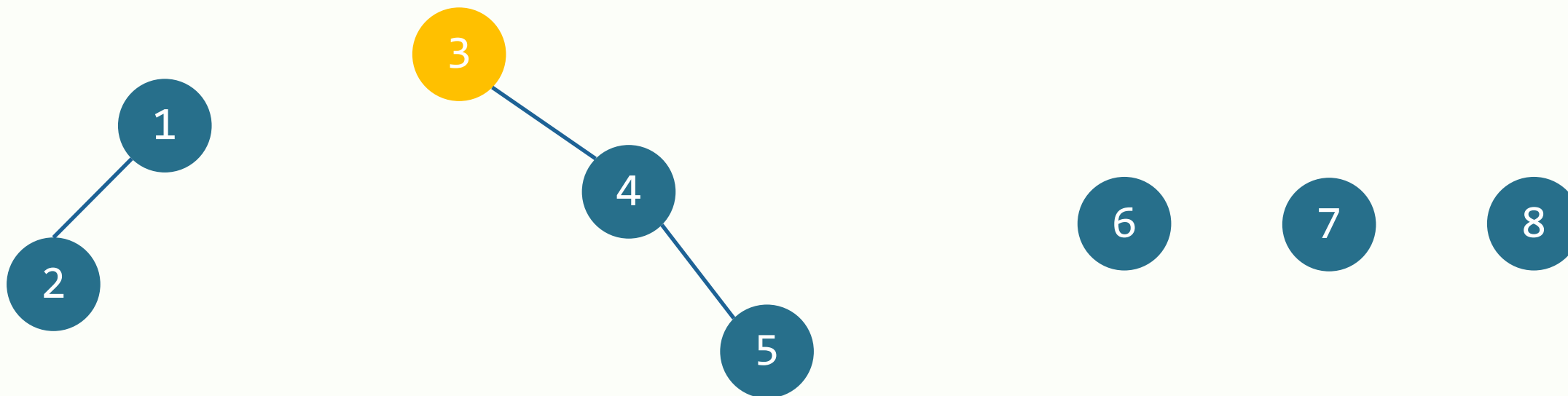
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 3$



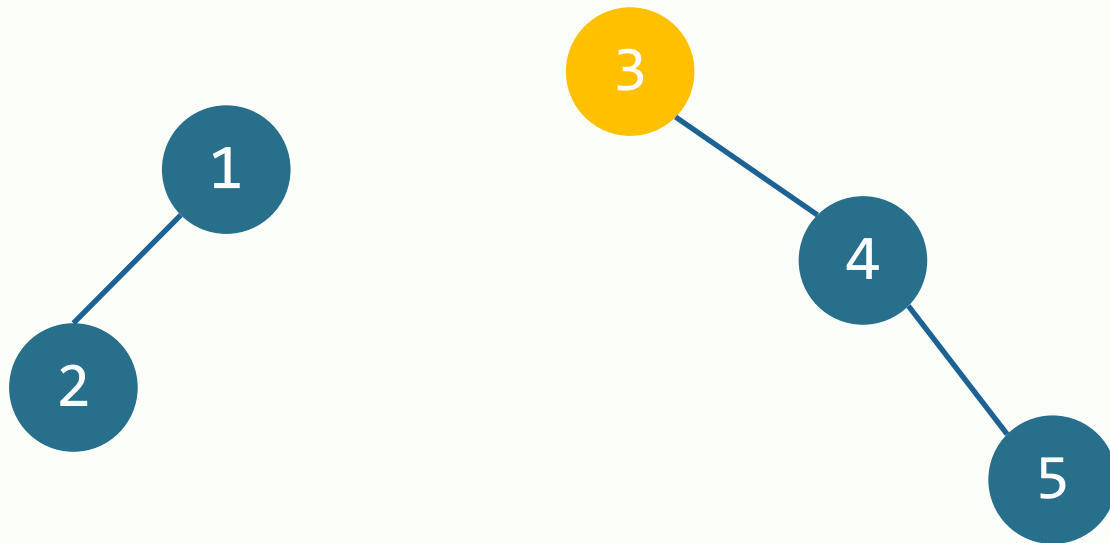
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 3$



```
int find(int x){  
    if(par[x] == x) return x;  
    else return find(par[x]);  
}
```

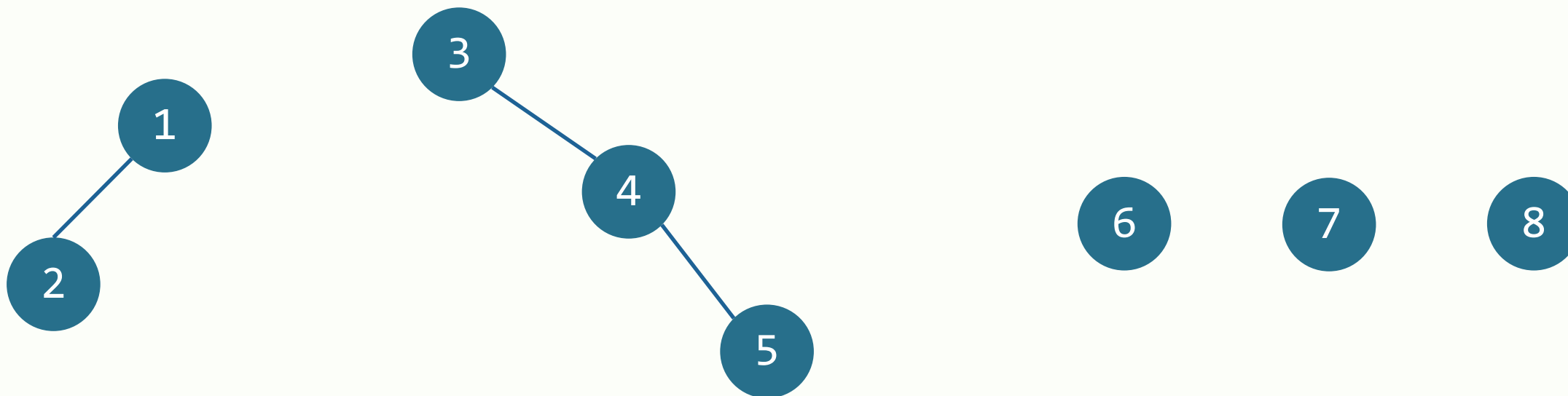
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4)$



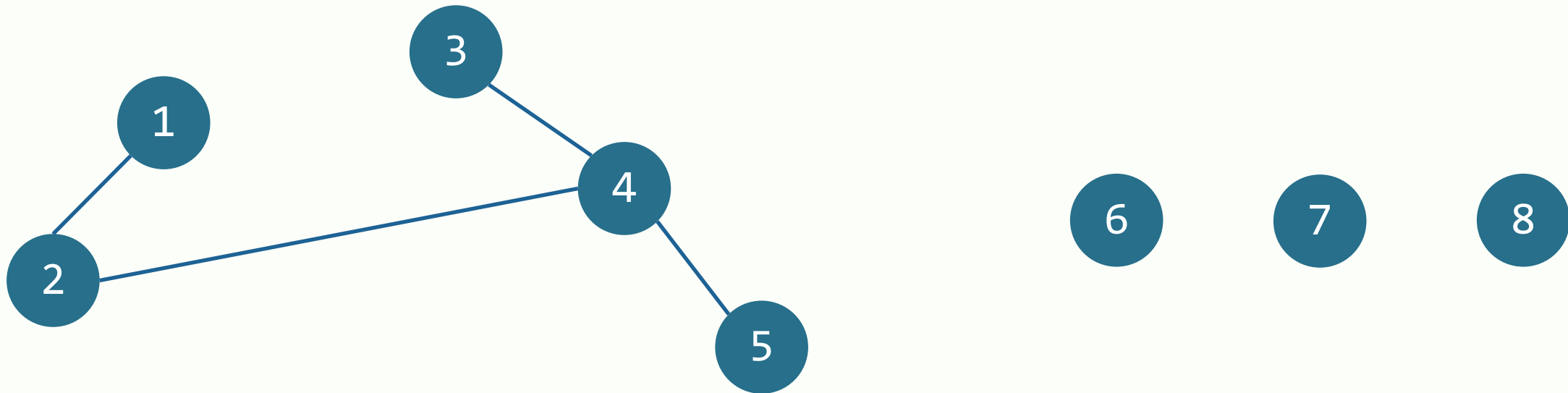
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4)$



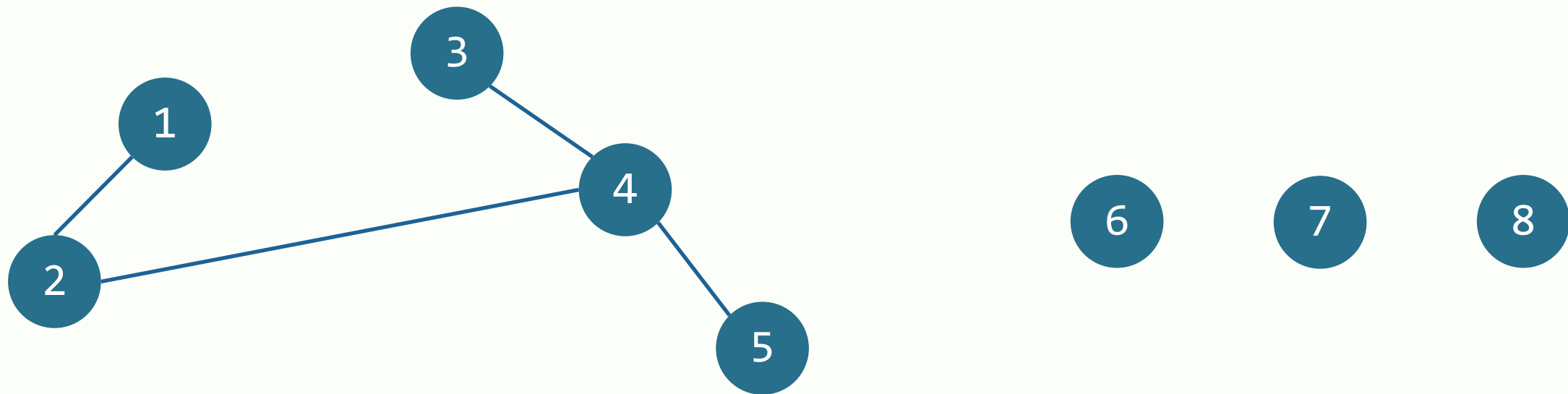
par

1	2	3	4	5	6	7	8
1	1	3	2	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(4) \Rightarrow 1, \text{find}(3) \Rightarrow 3$



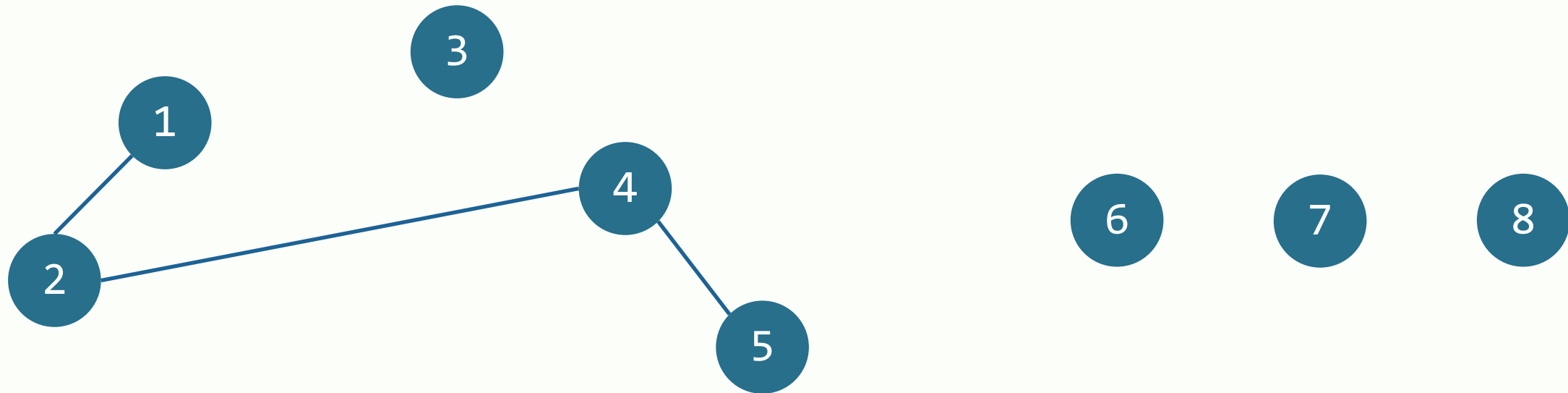
par

1	2	3	4	5	6	7	8
1	1	3	2	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(4) \Rightarrow 1, \text{find}(3) \Rightarrow 3$



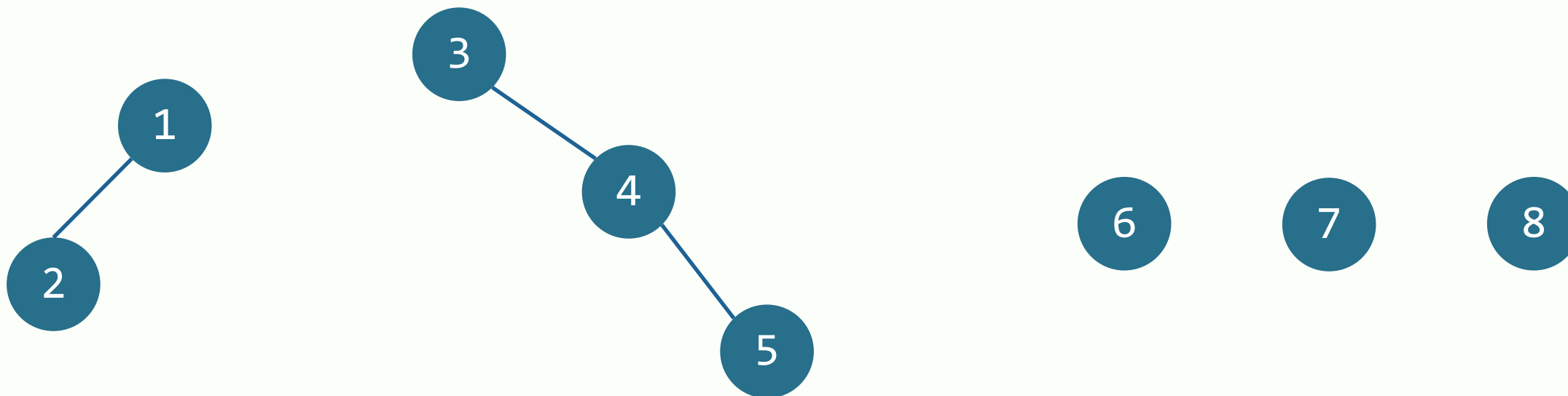
par

1	2	3	4	5	6	7	8
1	1	3	2	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4))$



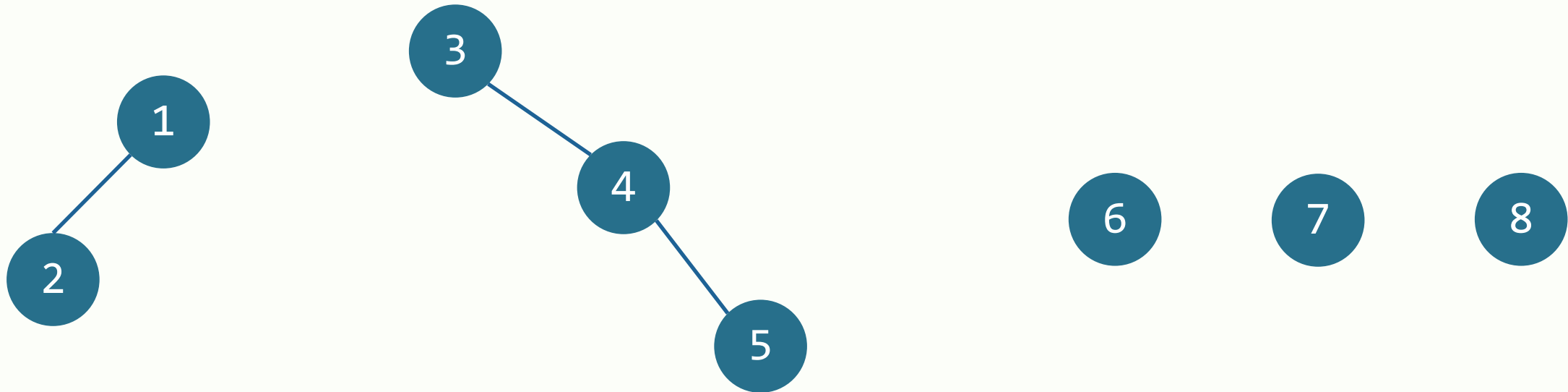
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(2, 3)$



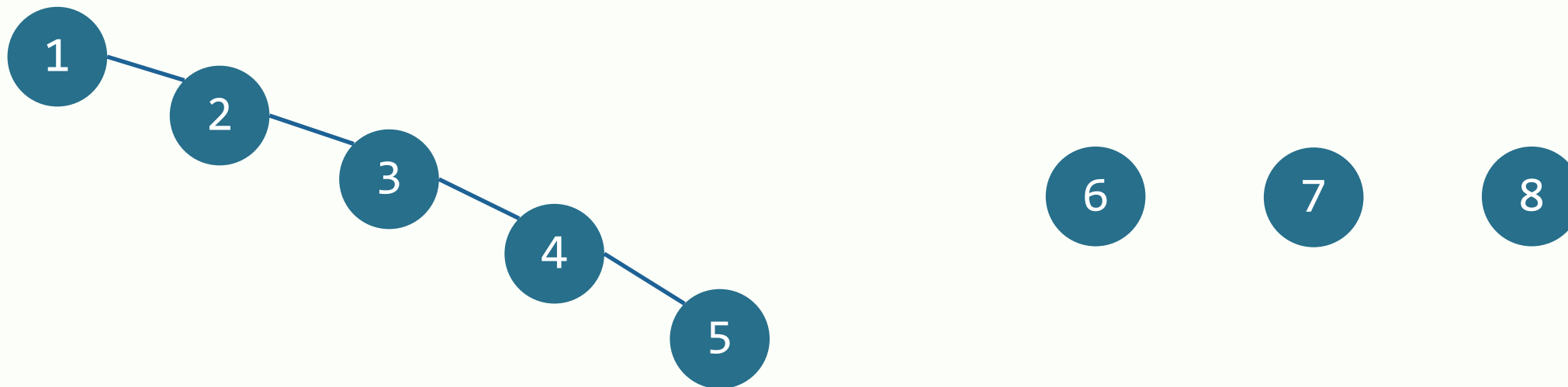
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(2, 3)$



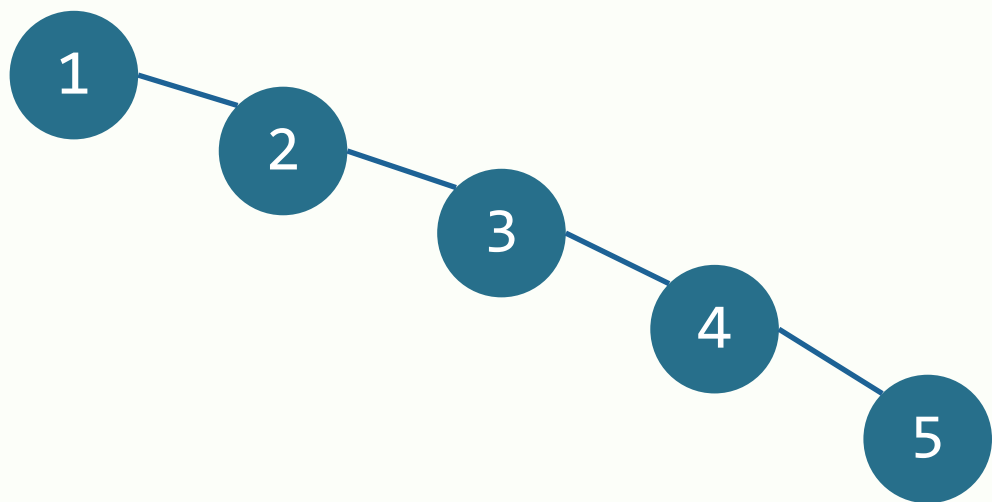
par

1	2	3	4	5	6	7	8
1	1	2	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(2, 3)$



```
void link(int x, int y){
    par[find(y)] = x;
}
```

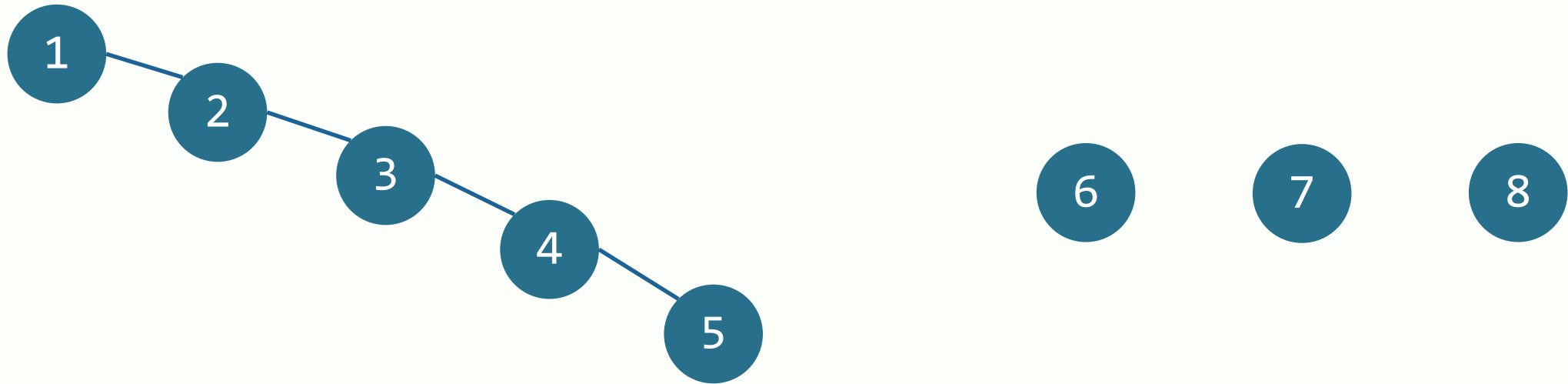
par

1	2	3	4	5	6	7	8
1	1	2	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



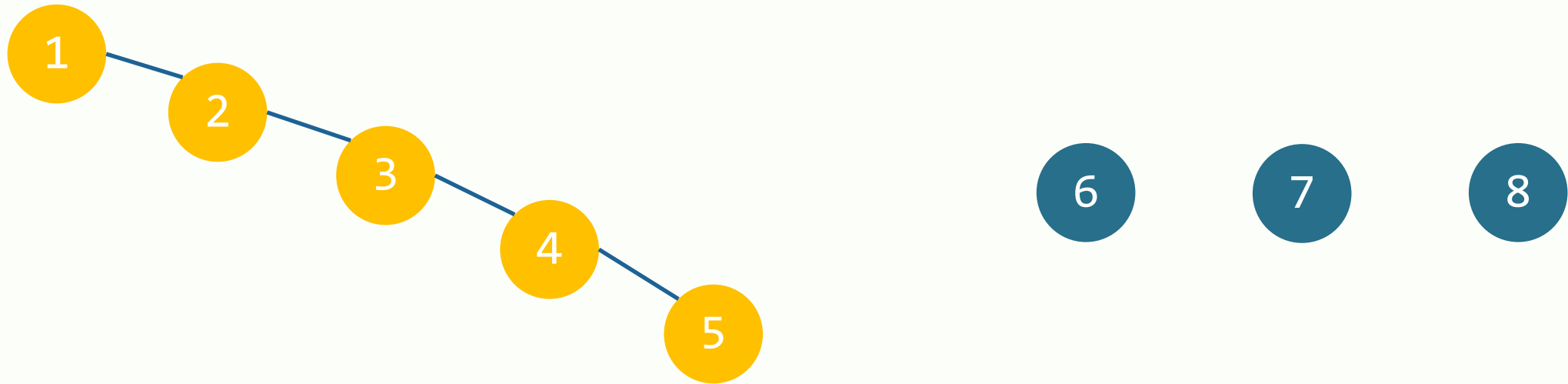
par

1	2	3	4	5	6	7	8
1	1	2	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$



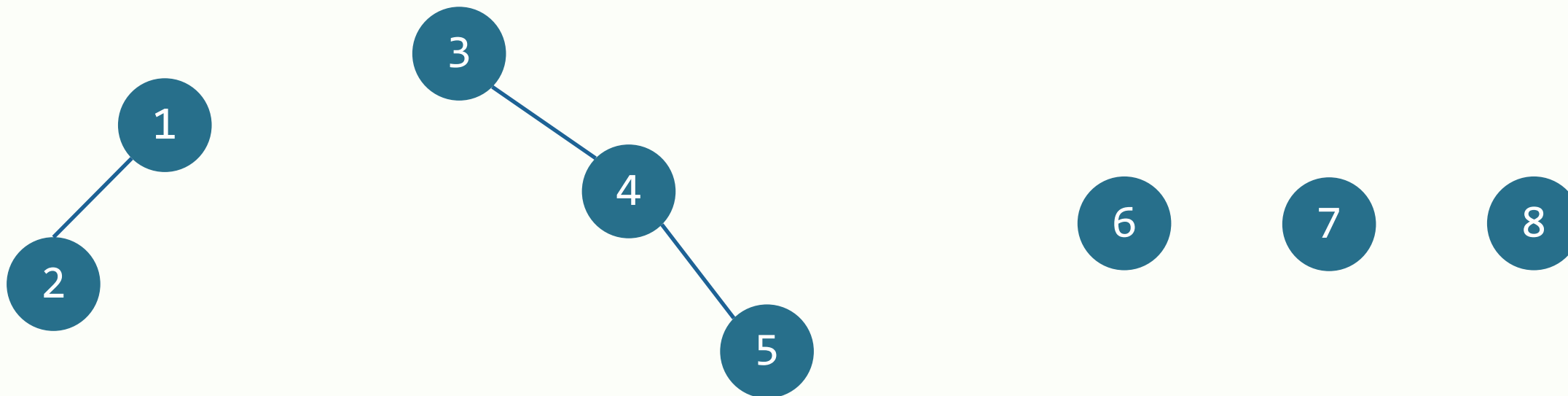
par

1	2	3	4	5	6	7	8
1	1	2	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4))$



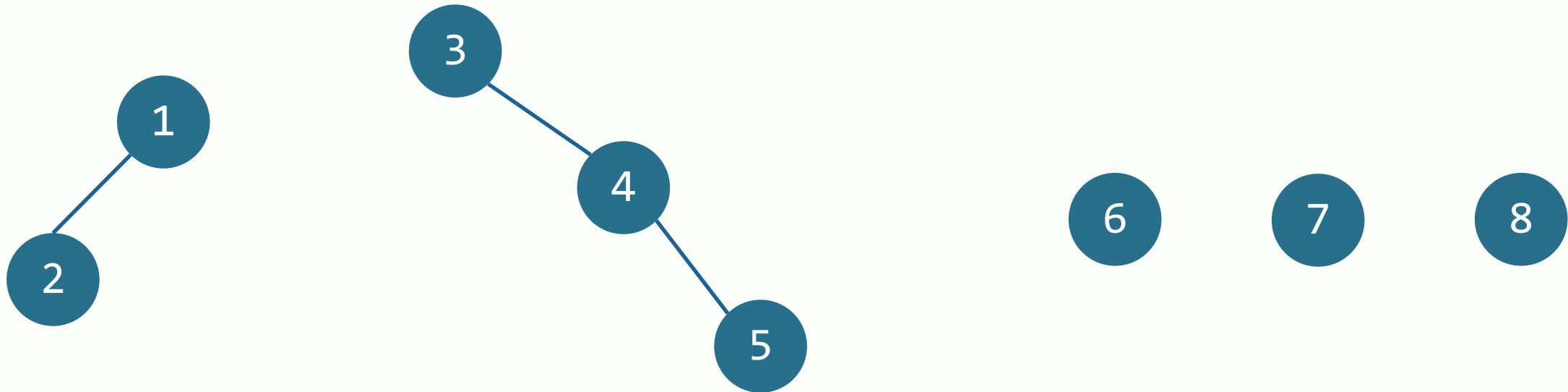
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(\text{find}(2), \text{find}(4))$



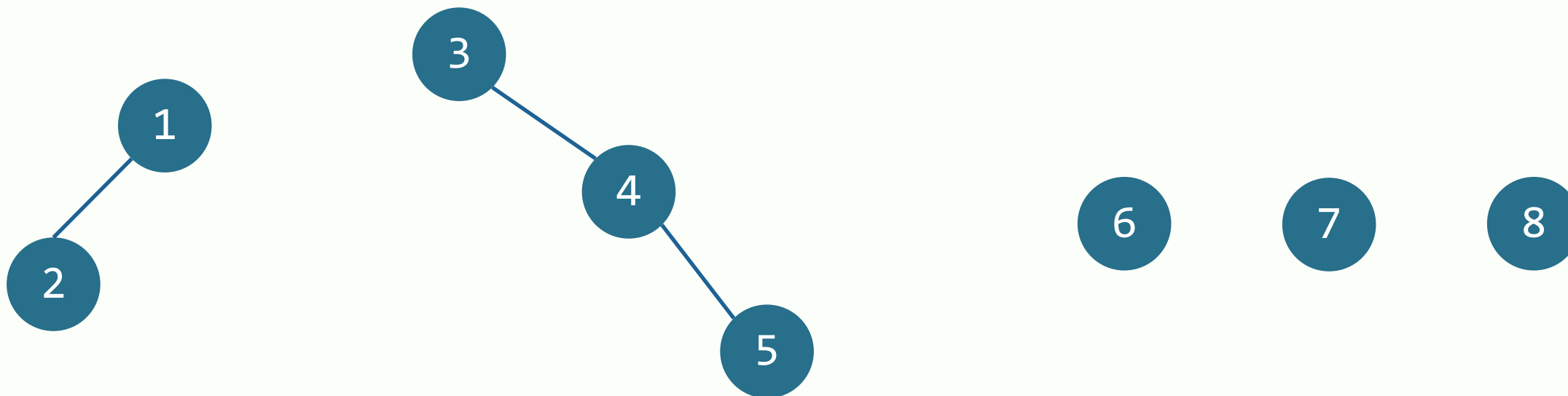
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(\text{find}(2), \text{find}(4)) \Rightarrow \text{link}(1, 3)$



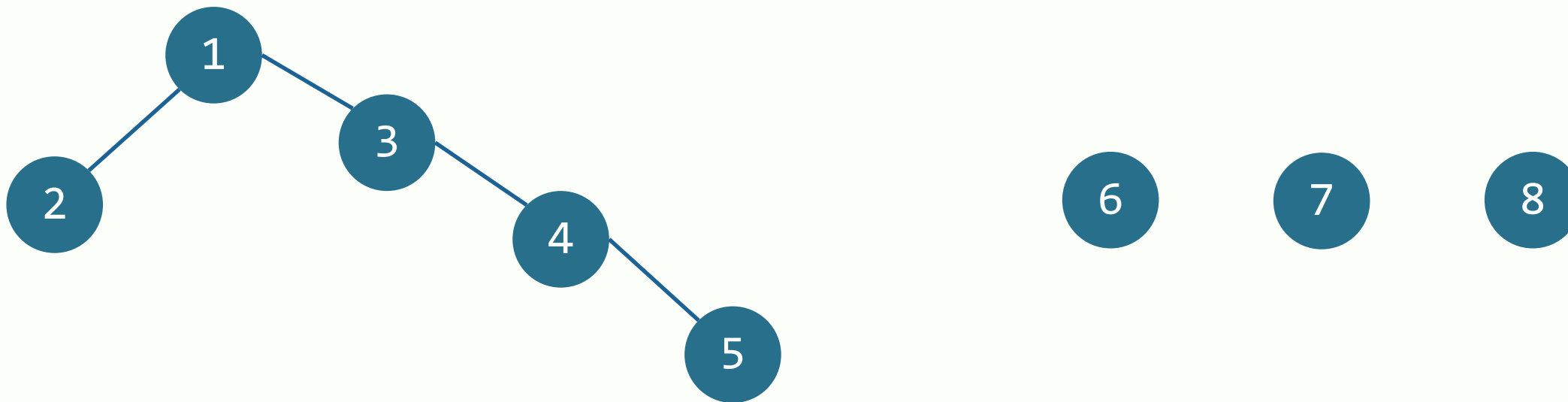
par

1	2	3	4	5	6	7	8
1	1	3	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(\text{find}(2), \text{find}(4)) \Rightarrow \text{link}(1, 3)$



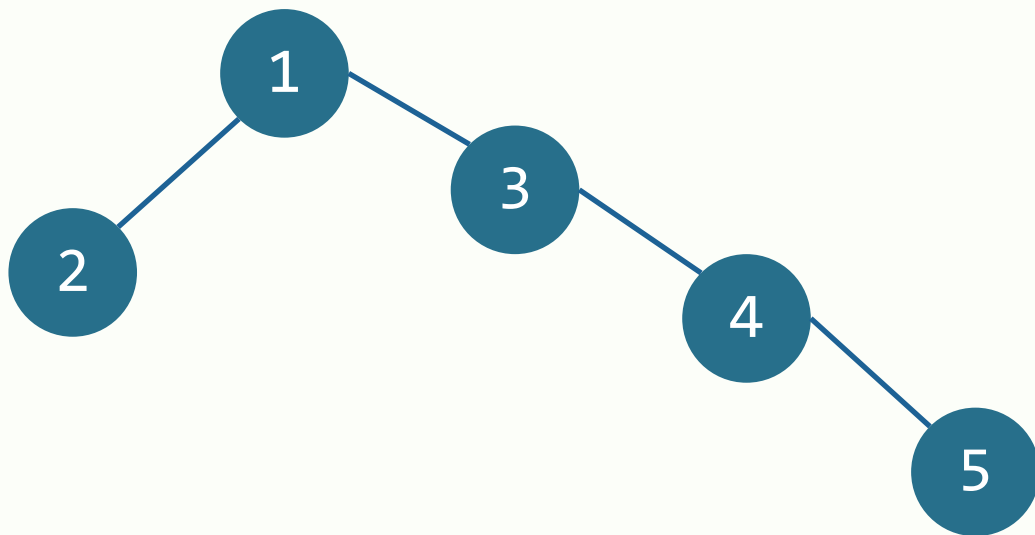
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{link}(2, 4) \Rightarrow \text{link}(2, \text{find}(4)) \Rightarrow \text{link}(\text{find}(2), \text{find}(4)) \Rightarrow \text{link}(1, 3)$



```
void link(int x, int y){
    par[find(y)] = find(x);
}
```

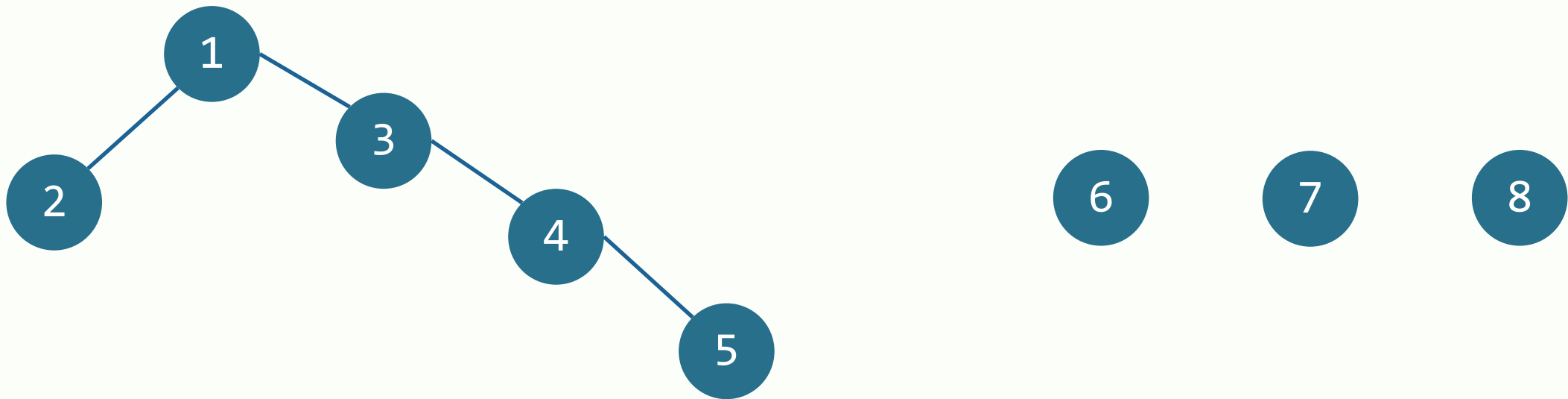
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- find(5)



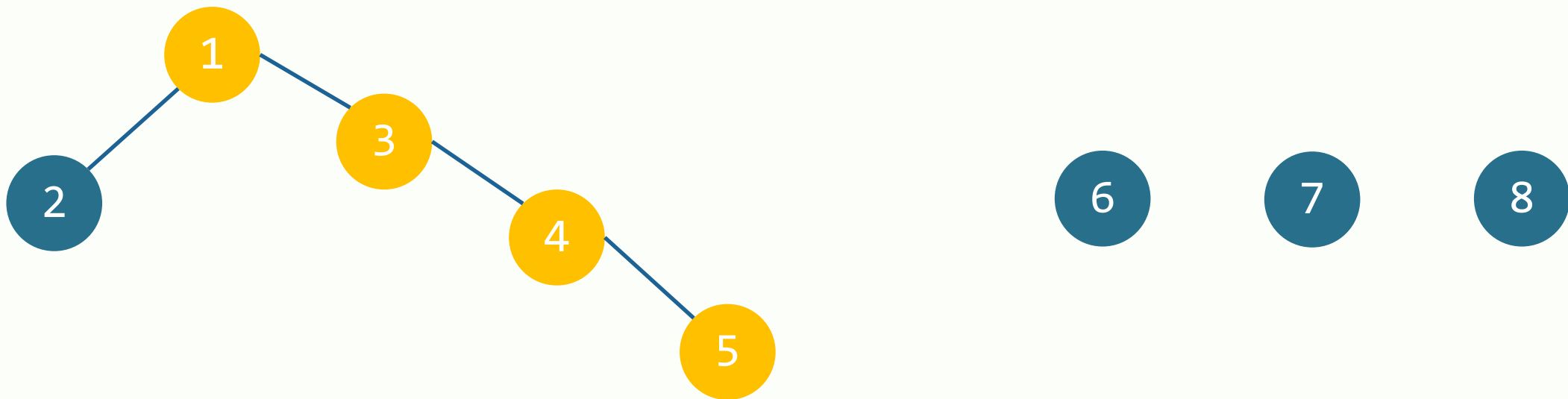
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$



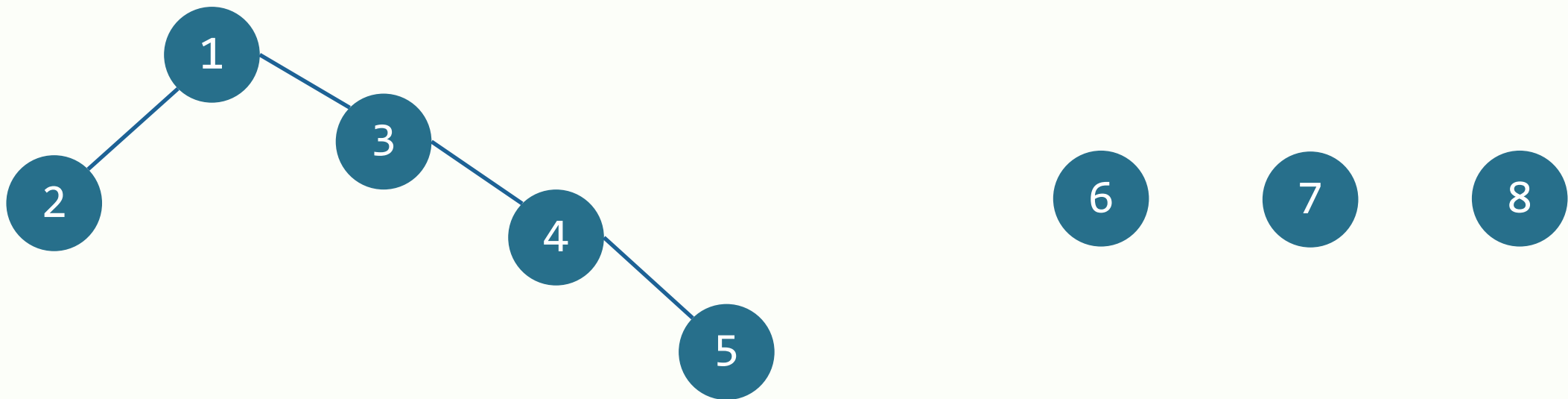
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- again, find(5)



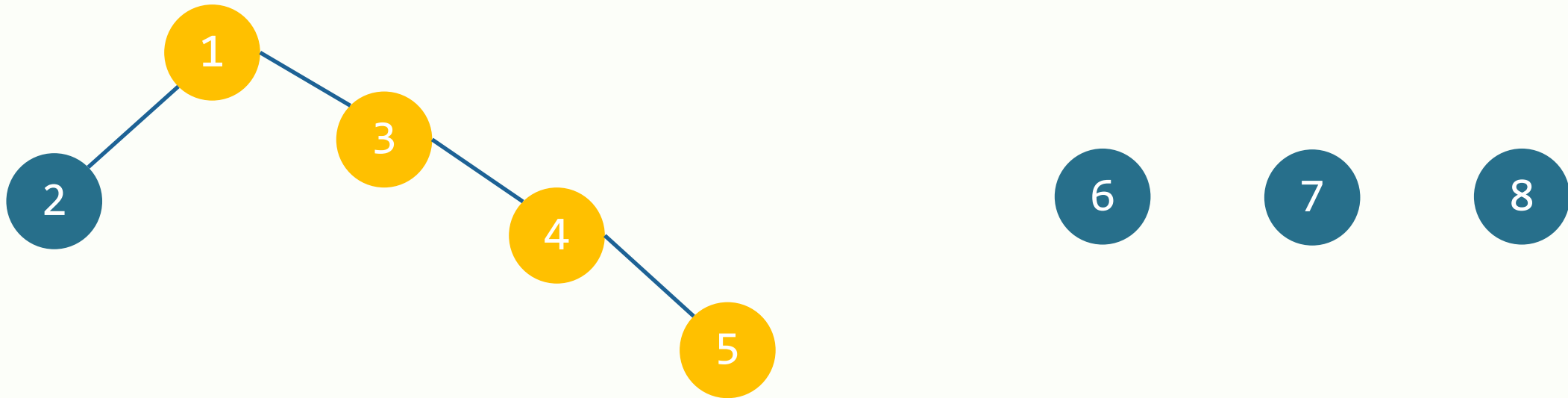
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- again, find(5)



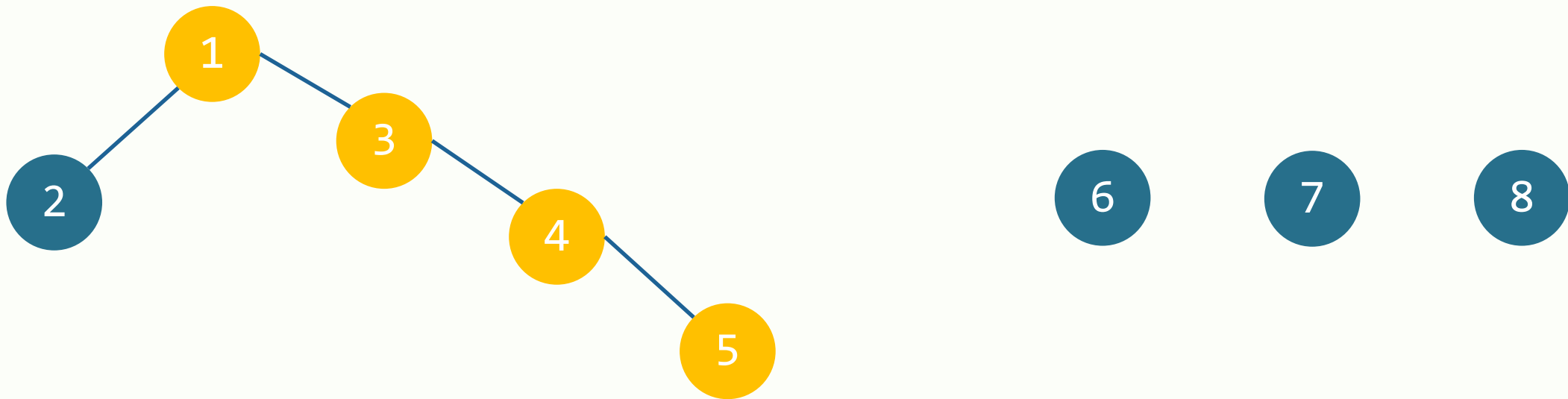
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$



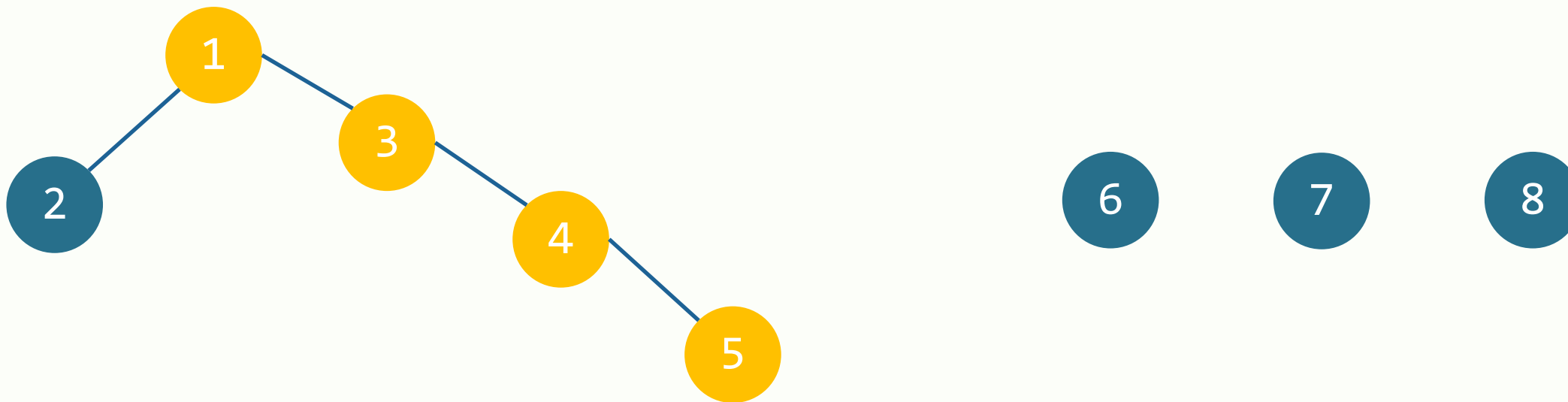
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(1) \Rightarrow 1$



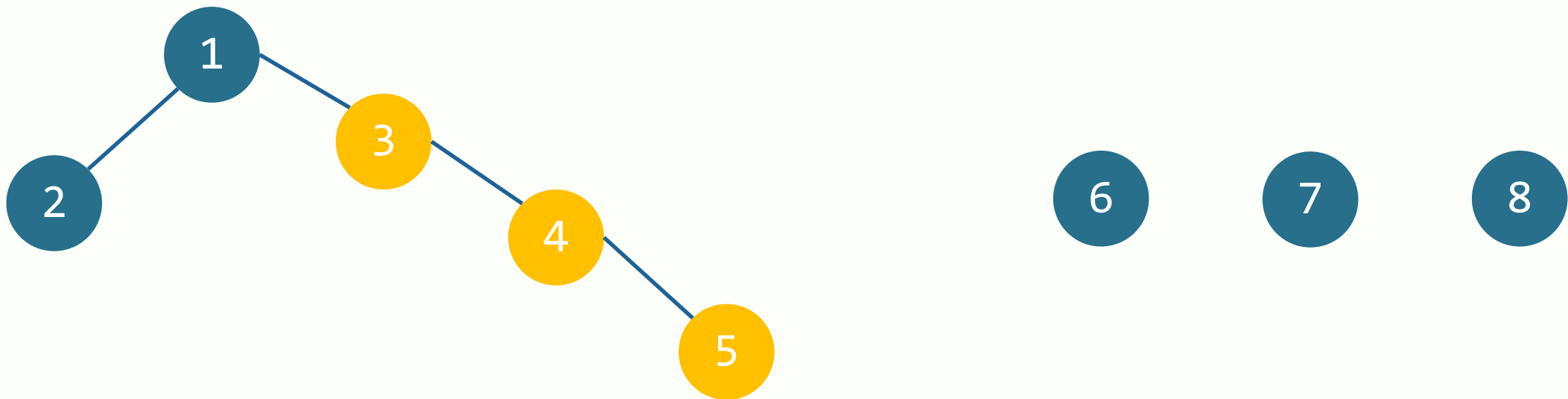
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(3) \Rightarrow \text{find}(1) \Rightarrow 1$



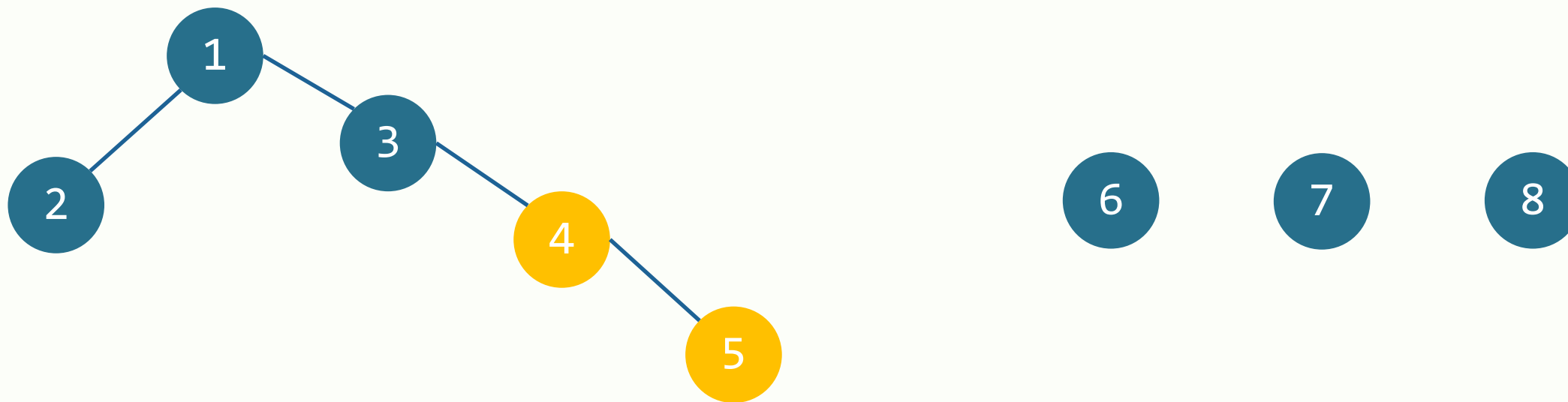
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(4) \Rightarrow \text{find}(3) \Rightarrow 1$



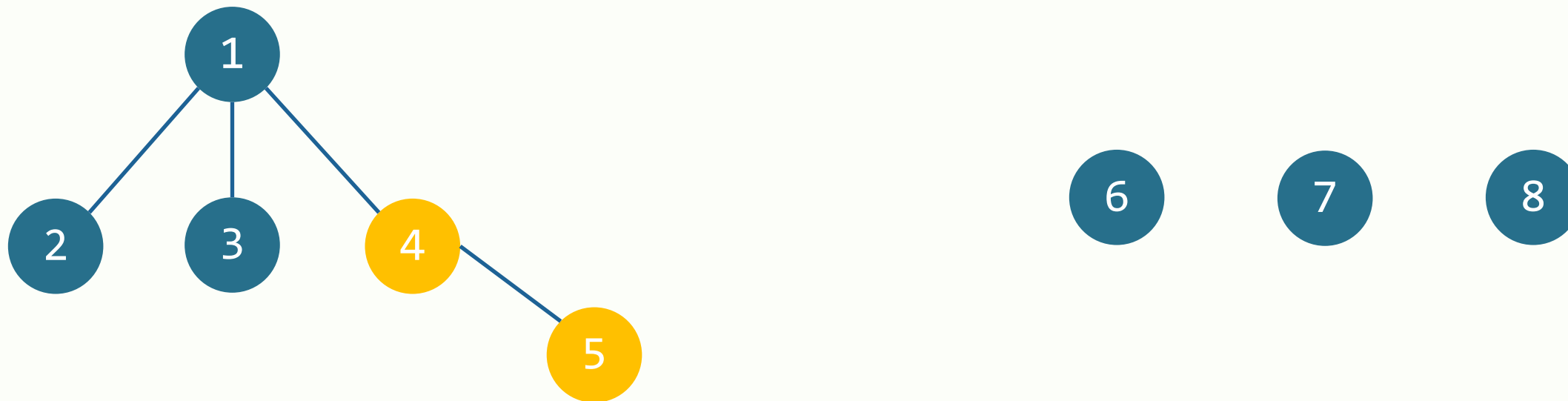
par

1	2	3	4	5	6	7	8
1	1	1	3	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(4) \Rightarrow 1$



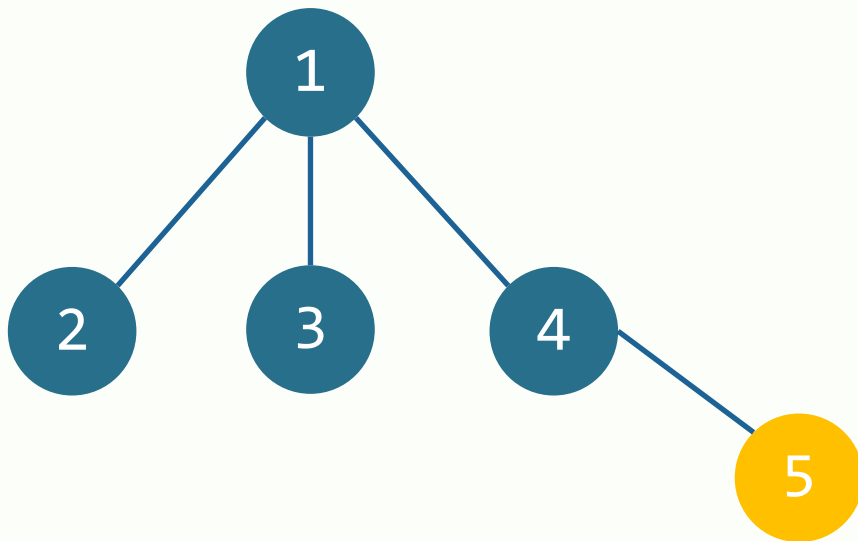
par

1	2	3	4	5	6	7	8
1	1	1	1	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow \text{find}(4) \Rightarrow 1$



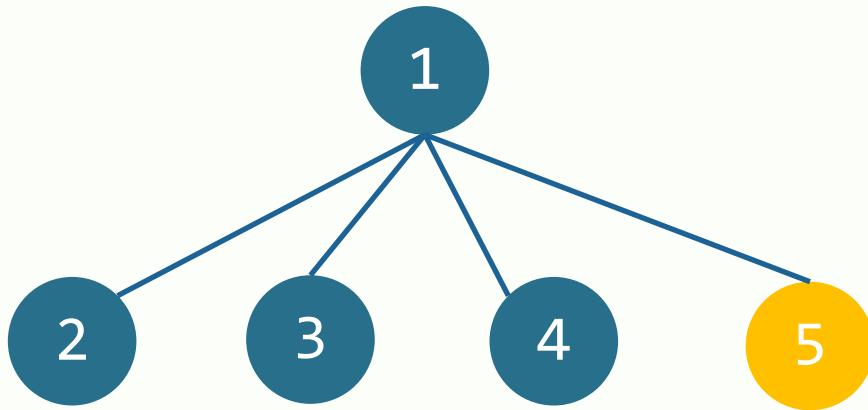
par

1	2	3	4	5	6	7	8
1	1	1	1	4	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$



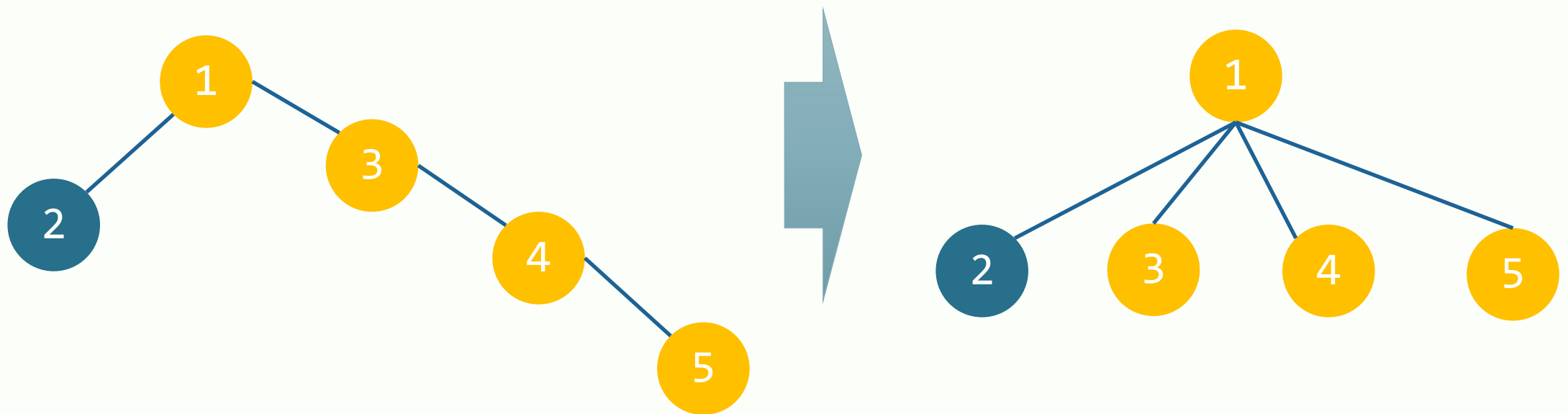
par

1	2	3	4	5	6	7	8
1	1	1	1	1	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$



par

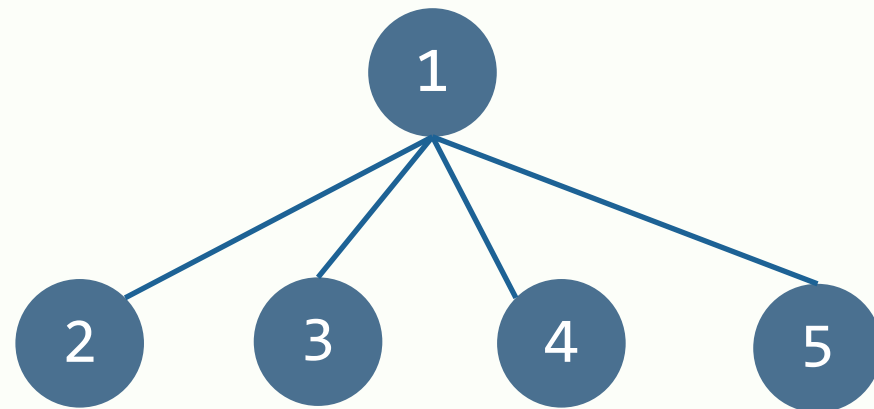
1	2	3	4	5	6	7	8
1	1	1	1	1	6	7	8

Disjoint-set

집합의 표현

- $\text{find}(5) \Rightarrow 1$

```
int find(int x){  
    if(par[x] == x) return x;  
    else return par[x] = find(par[x]);  
}
```



par	1	2	3	4	5	6	7	8
	1	1	1	1	1	6	7	8

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- 0 ~ N의 조상을 자신으로 초기화
- Op0: link(x, y)
- Op1: print(find(x) == find(y))

집합의 표현

Problem: <https://www.acmicpc.net/problem/1717>

- C/C++:

<https://gist.github.com/Acka1357/bfb3330c488c379996a096229abeb835>

- JAVA:

<https://gist.github.com/Acka1357/b37ab2661cc5937ddda23ebdd5111c83>

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- N개의 숫자가 있다.
 - A와 B를 고른다.
 - A의 소인수 X를 고른다.
 - $A \neq X, B \neq X$
-
- 위 연산을 반복해 N개의 숫자의 최대공약수를 최대로 한다.
 - 이 때 가능한 가장 큰 최대공약수와 필요한 연산의 최소횟수

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{8, 24, 9\}$ 가 있을 때

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{8, 24, 9\}$ 가 있을 때
- 1) $A = 9, B = 8, X = 3$
- $\Rightarrow \{24, 24, 3\}$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{8, 24, 9\}$ 가 있을 때
- 1) $A = 9, B = 8, X = 3$
- $\Rightarrow \{24, 24, 3\}$
- 2) $A = 24, B = 3, X = 2$
- $\Rightarrow \{12, 24, 6\}$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{8, 24, 9\}$ 가 있을 때
 - 1) $A = 9, B = 8, X = 3$
 - $\Rightarrow \{24, 24, 3\}$
- 2) $A = 24, B = 3, X = 2$
- $\Rightarrow \{12, 24, 6\}$
- 3) $A = 24, B = 6, X = 2$
- $\Rightarrow \{12, 12, 12\}$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\text{GCD}(12, 12, 12)$
- $\Rightarrow 12$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- N개의 수를 모두 소인수분해하고
- 각 소인수 개수의 합을 구한다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $8 = 2^3$
- $24 = 2^3 * 3^1$
- $9 = 3^2$

- $\{8, 24, 9\} \rightarrow \{2^3, 2^3 * 3^1, 3^2\}$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{2^3, 2^3 * 3^1, 3^2\}$
- 모든 수의 최대공약수가 최대가 되려면
- 최대한 많은 소인수를 공유해야 한다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{2^3, 2^3 * 3^1, 3^2\}$
- 전체 2의 개수: 6개
- 전체 3의 개수: 3개

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{2^3, 2^3 * 3^1, 3^2\}$
- 전체 2의 개수: 6개
- 전체 3의 개수: 3개
- 3개의 숫자가 위 소인수를 최대한 많이 공유하려면
- 각각 2를 2개씩, 3을 1개씩 가질 수 있다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $\{2^3, 2^3 * 3^1, 3^2\}$
- 3개의 숫자가 위 소인수를 최대한 많이 공유하려면
- 각각 2를 2개씩, 3을 1개씩 가질 수 있다.
- 전체 숫자가 1개 이상씩 공유할 수 있는 소인수가 더 없으므로
- 답: $2^2 * 3^1$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- 소인수 분해를 하려면?
- => 소수를 먼저 구해야 한다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- 양의 정수 X 가 소수인지를 판별하기 위해서는
- $2 \sim \sqrt{X}$ 까지의 정수로 나누어 떨어지는지 확인한다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- 수의 범위를 M이라고 하면
- 1~M까지 모든 소수를 구하는 시간복잡도는 아래와 같다.
- 시간복잡도: $O(M * \sqrt{M})$
 - $1 \leq M \leq 1,000,000$

Sieve of Eratosthenes

소수를 보다 빠르게 구할 수 있는 알고리즘

Sieve of Eratosthenes

소수 구하기

- 구하고자 하는 범위의 모든 수를 쓴다.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 2부터 시작해, 가장 작은 소수의 배수를 지운다.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 2부터 시작해, 가장 작은 소수의 배수를 지운다.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 다음으로 남은 소수는 3

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 다음으로 남은 소수는 3

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 다음으로 남은 소수는 7

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Sieve of Eratosthenes

소수 구하기

- 이를 배열을 이용해 다음과 같이 코드로 나타낼 수 있다.

```
int notP[1000001] = { true, true, }, pcnt, p[PRIME_COUNT];

void set_prime(){
    for (int i = 2; i < 1000000; i++){
        if (notP[i]) continue;
        p[pcnt++] = i;
        for (int j = 2 * i; j <= 1000000; j += i)
            notP[j] = true;
    }
}
```

Sieve of Eratosthenes

소수 구하기

- 조금 더 효율적으로

```
int notP[1000001] = { true, true, }, pcnt, p[PRIME_COUNT];

void set_prime(){
    for (long long i = 2; i < 1000000; i++){
        if (notP[i]) continue;
        p[pcnt++] = i;
        for (long long j = i * i; j <= 1000000; j += i)
            notP[j] = true;
    }
}
```


수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $fcnt[i]$: N개의 숫자를 소인수분해 했을 때, i번째 소수 개수의 합
- N개의 수의 최대공약수가 최대가 되기 위해서는
- 각 소인수를 N개의 수에 균등히 분배한 것과 같다.

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- $fcnt[i]$: N개의 숫자를 소인수분해 했을 때, i번째 소수 개수의 합
 - N개의 수의 최대공약수가 최대가 되기 위해서는
 - 각 소인수를 N개의 수에 균등히 분배한 것과 같다.
-
- $max_gcd = \text{모든 } prime[i]^{(fcnt[i] / N)} \text{의 곱}$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- 필요한 최소 연산의 횟수
 - 입력으로 주어진 수의 소인수 중
 - max_gcd 의 소인수 개수보다 작은 것이 있다면
 - 더 큰 수로부터 받게 된다.
-
- 각 수가 max_gcd 가 되기 위해 받아야만 하는 소인수 개수의 총합

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- 시간복잡도: $O(M * \log M)$
 - $O(\text{소수구하기}) + O(N) * O(\text{소인수분해})$
 - $O(\text{소수구하기}): O(M * \log M)$
 - $O(\text{소인수분해}): O(\text{소수개수}): 98498$
 - $O(M * \log M) + O(N) * O(\text{소수개수})$
 - $1 \leq M \leq 1,000,000$

수학은 너무 쉬워

Problem: <https://www.acmicpc.net/problem/2904>

- C/C++:
<https://gist.github.com/Acka1357/d3bfca476ea381eca498312ae58d597e>
- JAVA:
<https://gist.github.com/Acka1357/ddb4d1987af4b028c575bc3bd7582b10>