알고리즘 문제해결기법

제1주

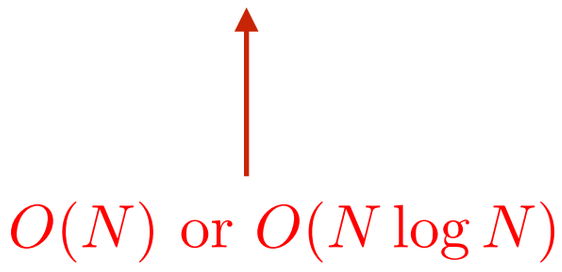
Problem 01: Chocolates

We are given a sequence of N positive integers where each value represents number of chocolates in a packet. Each packet can have variable number of chocolates. There are M students. The task is to distribute chocolate packets such that :

(1) Each student gets exactly one packet.

(2) The difference between the number of chocolates in packet with maximum chocolates and packet with minimum chocolates given to the students is minimum.

N and M are at most 1,000,000 and M ≦ N



Problem 01: Chocolates

이항계수와 그것이 어느 정도로
큰 값인지 정도는 알고 있어야 !!!

- N개의 정수중에 M개를 뽑는 문제. 경우의 수는?

$$\binom{N}{M} = \frac{N!}{(N-M)!M!}$$

정렬을 하더라도 해가 바뀌지 않는 경우라면
"정렬을 하면 어떨까?" 생각해 본다.

- 정렬을 한다면?
  - 연속된 M개의 정수를 선택하는 것이 최대값과 최소값의 차이를 최소로 만든다.
  - 경우의 수는 N-M+1 가지로 줄어든다.

해의 특성에 대한 observation은
탐색할 해공간의 크기를 줄여 준다.

정렬

- Bubble sort
- Insertion sort
- Selection sort

$O(N^2)$

- Quicksort
- Merge sort
- Heap sort

$O(NlogN)$, 물론 quicksort는 최악의 경우 $O(N^2)$

- Radix sort

$O(N)$, but not comparison sort

Sort in C, C++, Java

- C++
  - https://www.youtube.com/watch?v=pIvGGvPwsvI
- Java
  - https://www.youtube.com/watch?v=ESwVlixFtak&list=PL52K_8WQO5oUuH06MLOrah4h05TZ4n38l&index=18
- C
  - qsort 함수

자신이 사용하는 언어의 표준 라이브러리가 제공하는
정렬 함수를 능숙하게 구사해야 한다.

Problem 01: Chocolates

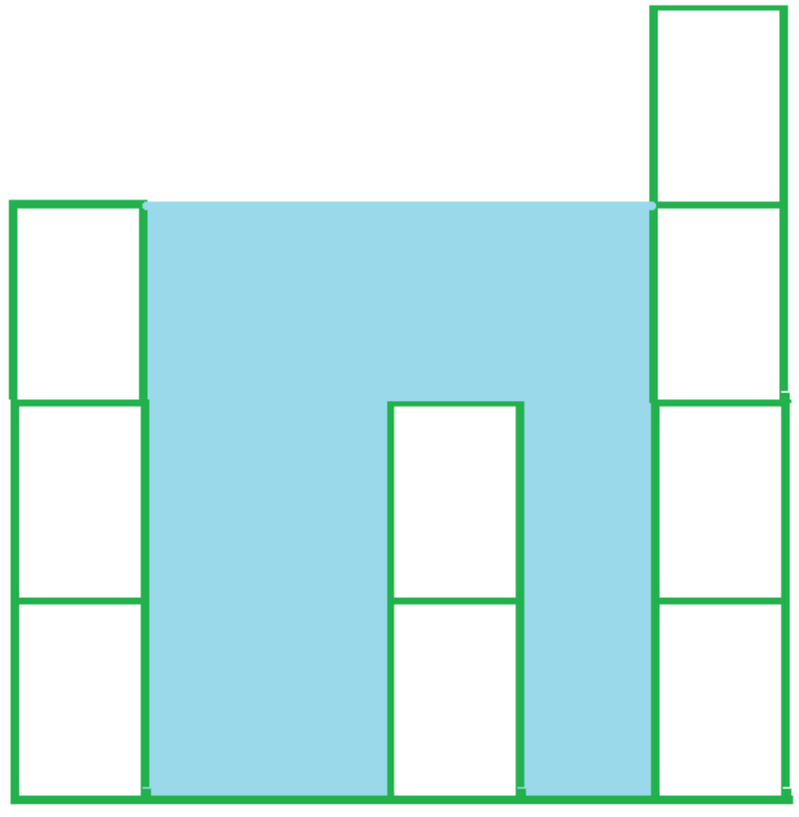
```
sort(data);
int min = data[M-1]-data[0];
for (int i=1; i<N-M+1; i++)
    min = (min > (t=data[i+M-1]-data[i]) ? t : min);
print(min);
```

O(NlogN)

```
sort(data)
min_diff = min([data[i+m-1]-data[i] for i in range(n-m+1)])
```

Problem 03: Water

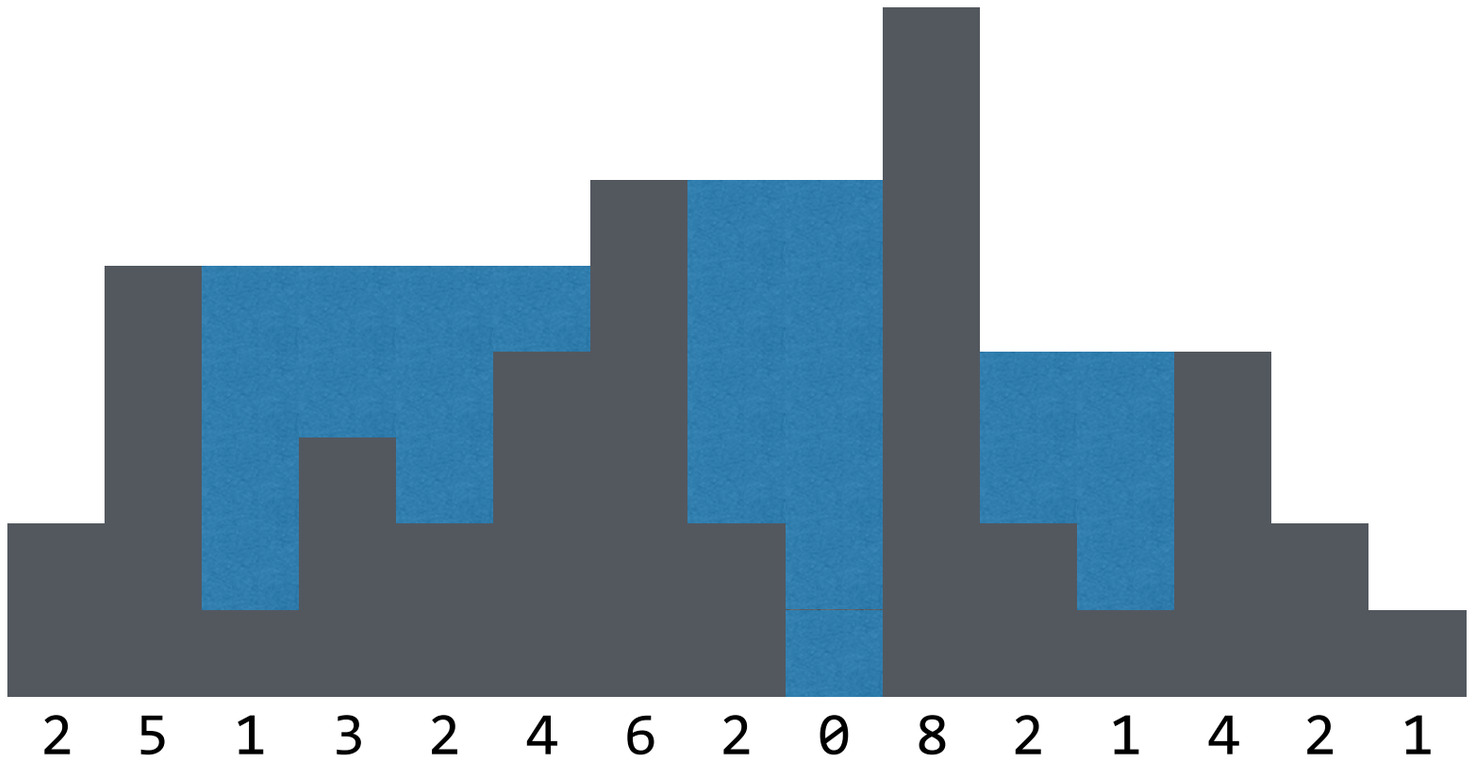
Given N≦1,000,000 non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining. The following figure depicts the situation where given integers are {3, 0, 0, 2, 0, 4}. The total amount of trapped water is 10 as illustrated in the figure.



Bars for input {3, 0, 0, 2, 0, 4}
Total trapped water = 3 + 3 + 1 + 3 = 10

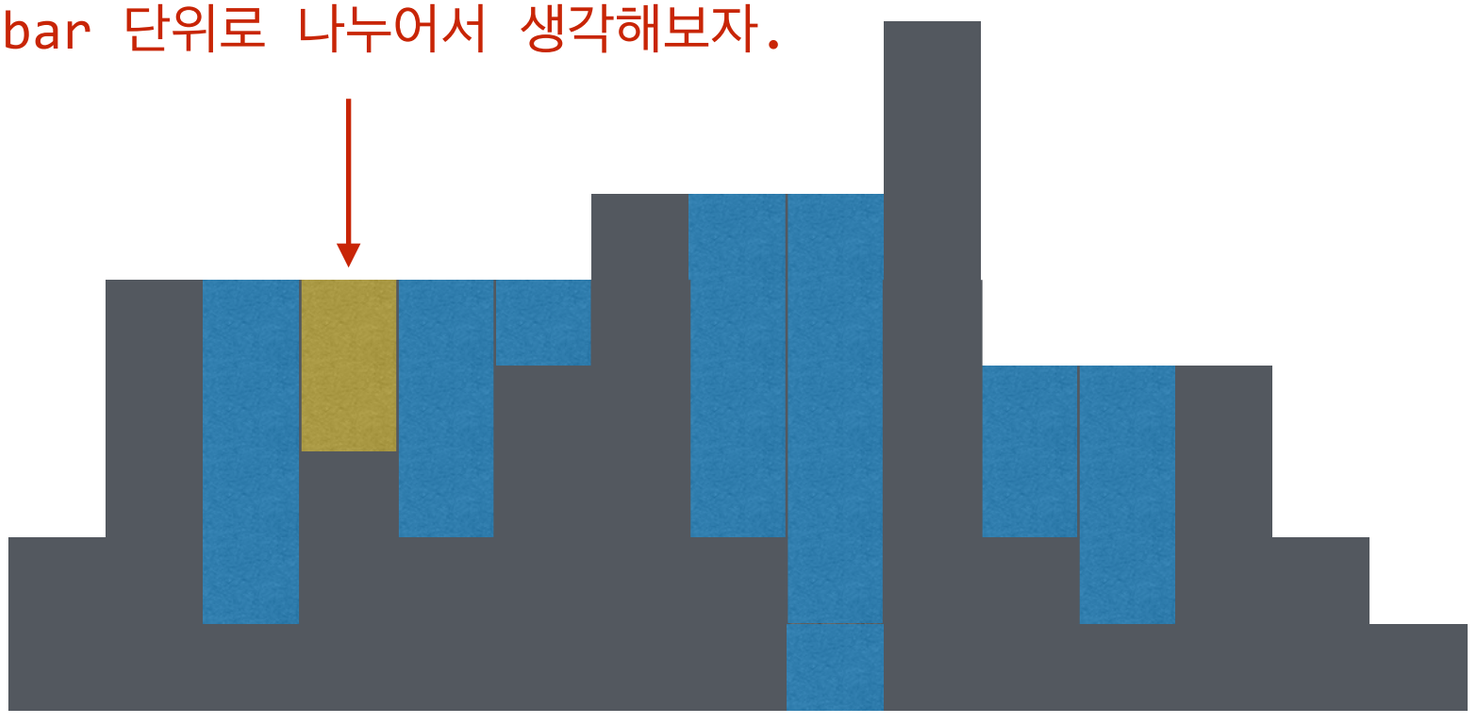
Problem 03: Water

예를 만들어 본다.



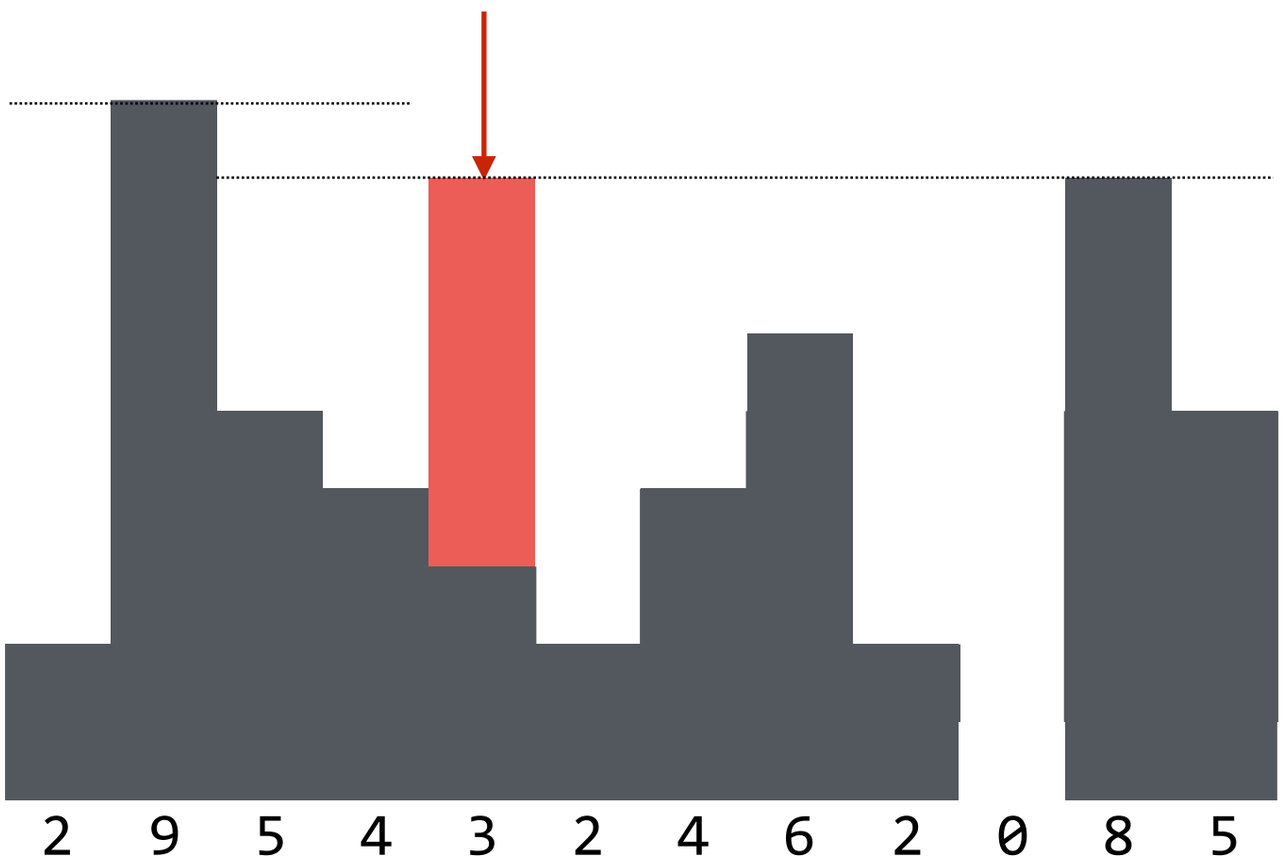
Problem 03: Water

각각의 bar 단위로 나누어서 생각해보자.



Key Observation

이 칸에 쌓일 물의 양은 왼쪽의 최대값과
오른쪽 최대값 중 작은 값에 의해서 결정



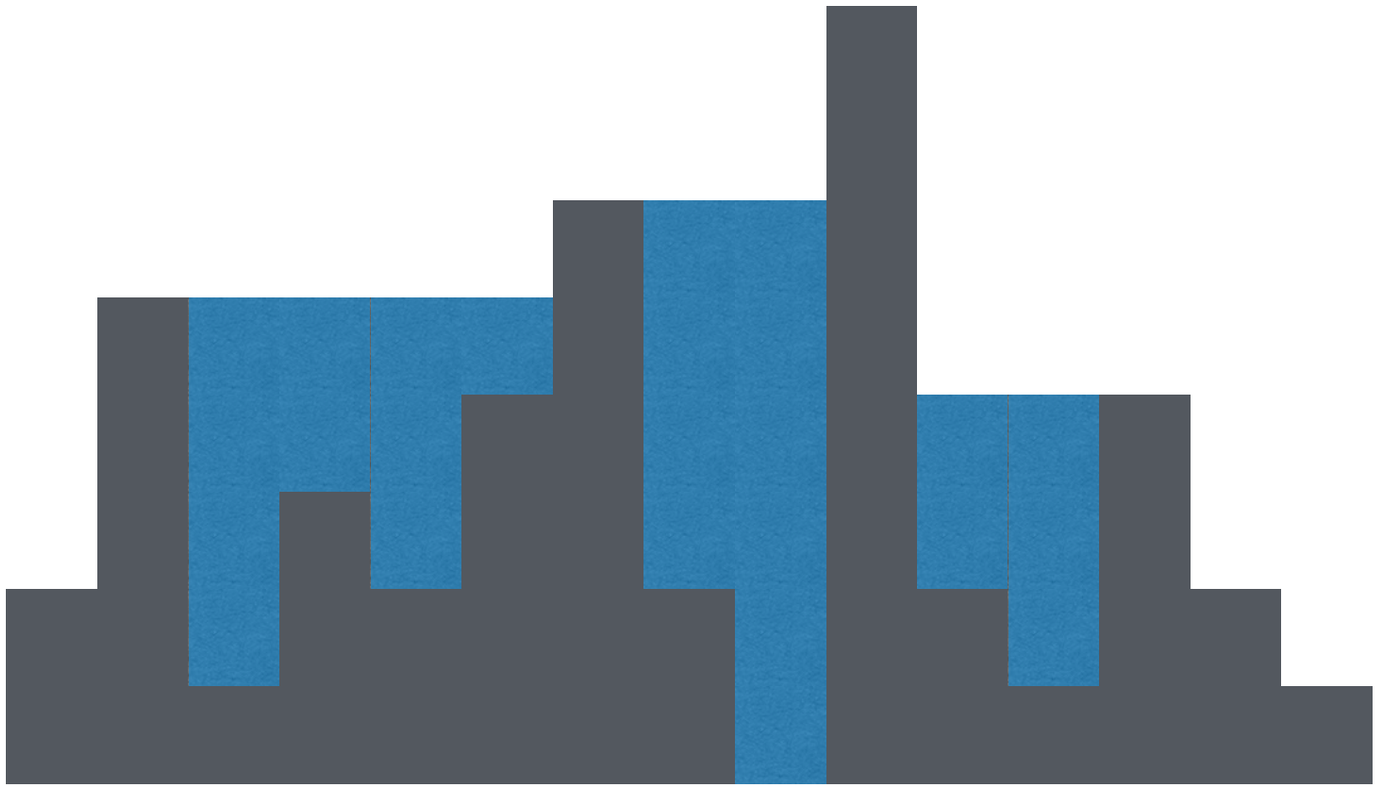
적절한 표기를 이용하여
이렇게 수식으로
표현을 해보는 것
은 중요하다.

$$w(i) = \min\{\text{maxLeft}(i), \text{maxRight}(i)\} - h(i)$$

i번째 칸에 고인 물의 양

자신을 포함하여 왼쪽/오른쪽의 최대값

Problem 03: Water



h	2	5	1	3	2	4	6	2	0	8	2	1	4	2	1
maxLeft	2	5	5	5	5	5	6	6	6	8	8	8	8	8	8
maxRight	8	8	8	8	8	8	8	8	8	8	4	4	4	2	1
w	0	0	4	2	3	1	0	4	6	0	2	3	0	0	0

O(N)

preprocessing
(해를 구하기 위해서
필요한 어떤 값들을 미리 구해두는 일)

Problem 03: Water

```
left_max = [h[0]]
right_max = [h[n-1]]
for i in range(n-1):
    left_max.append(max(h[i+1], left_max[-1]))
    right_max.append(max(h[n-i-2], right_max[-1]))

print(sum([(min(left_max[i], right_max[n-i-1])-h[i]) for i in range(n)]))
```

풀려고 하지 말고 관찰할 것

Problem 02: Platforms

Given arrival and departure times of all trains that reach a railway station, find the minimum number of platforms required to accommodate all the trains. Suppose that we have 6 trains and their arrival and departure times are as follows:

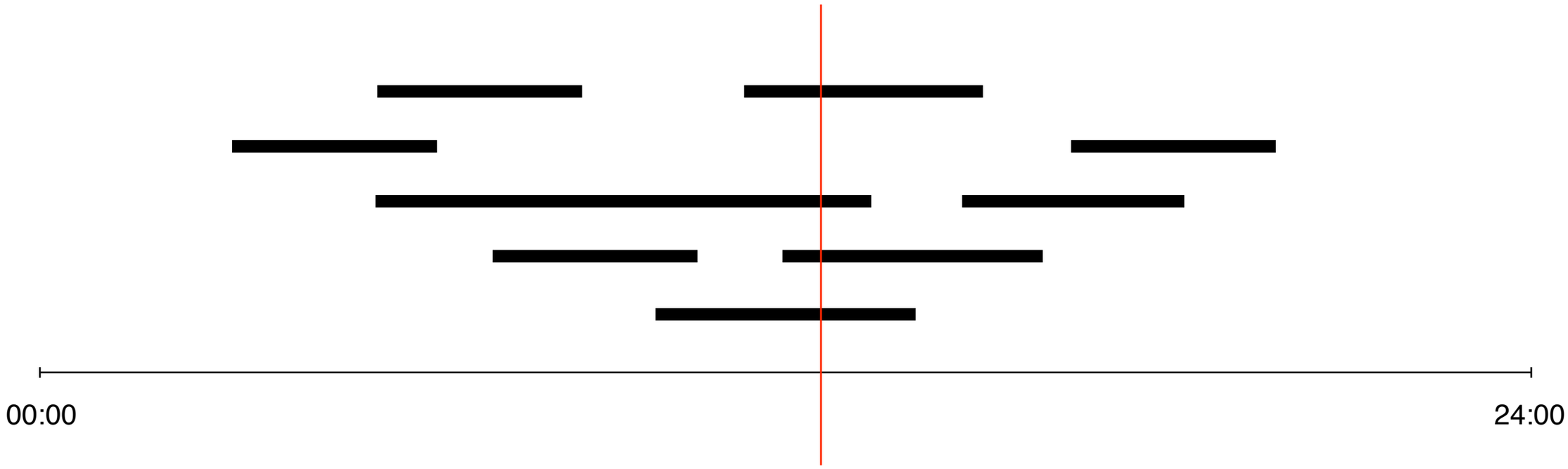
(9:00, 9:10), (9:40, 12:00), (9:50, 11:20), (11:00, 11:30), (15:00, 19:00), (18:00,20:00)

Two trains can share a platform if their time intervals do not overlap. In the above example, the station should have at least three platforms since three trains (2nd, 3rd, and 4th train) stay at the station between 11:00 and 11:20.

Given arrival and departure times of N trains, write a program to compute the minimum number of required platforms. Each arrival and departure time is between 00:00 and 24:00. The number of trains N is at most 1,000,000.

Platforms

막연하게 머리로 생각하지 말고 구체적인
예를 가지고 생각할 것.
그리고 예를 어떻게든 시각화 할 것

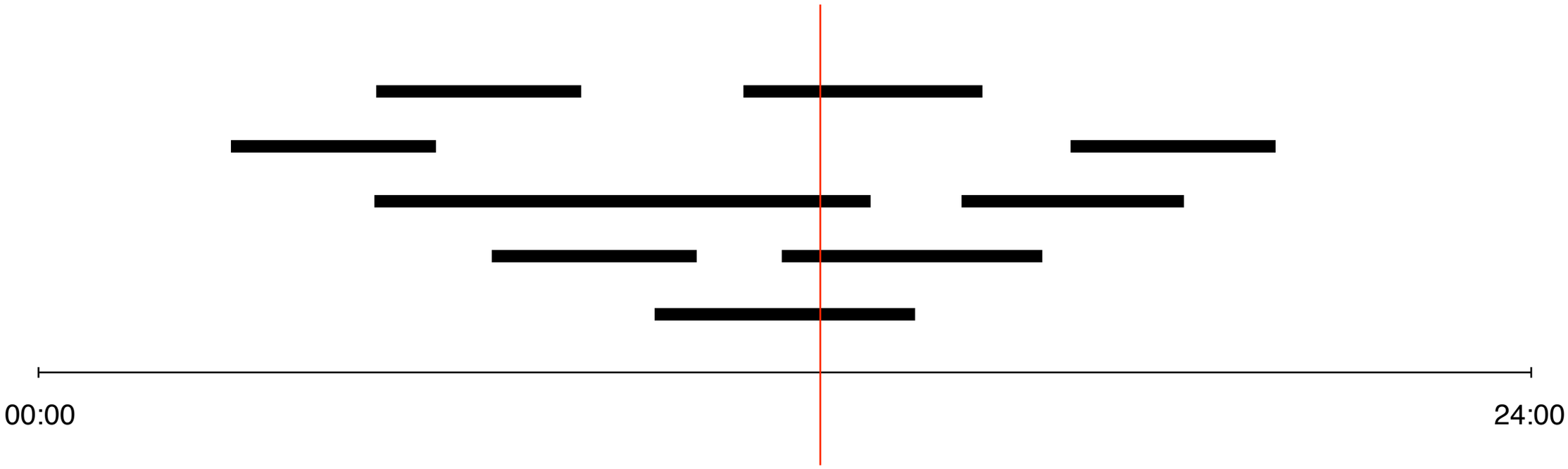


이 시점에 동시에 4대의 기차가 머무른다.

observation

따라서 **적어도** 4개의 플랫폼이 필요하다.

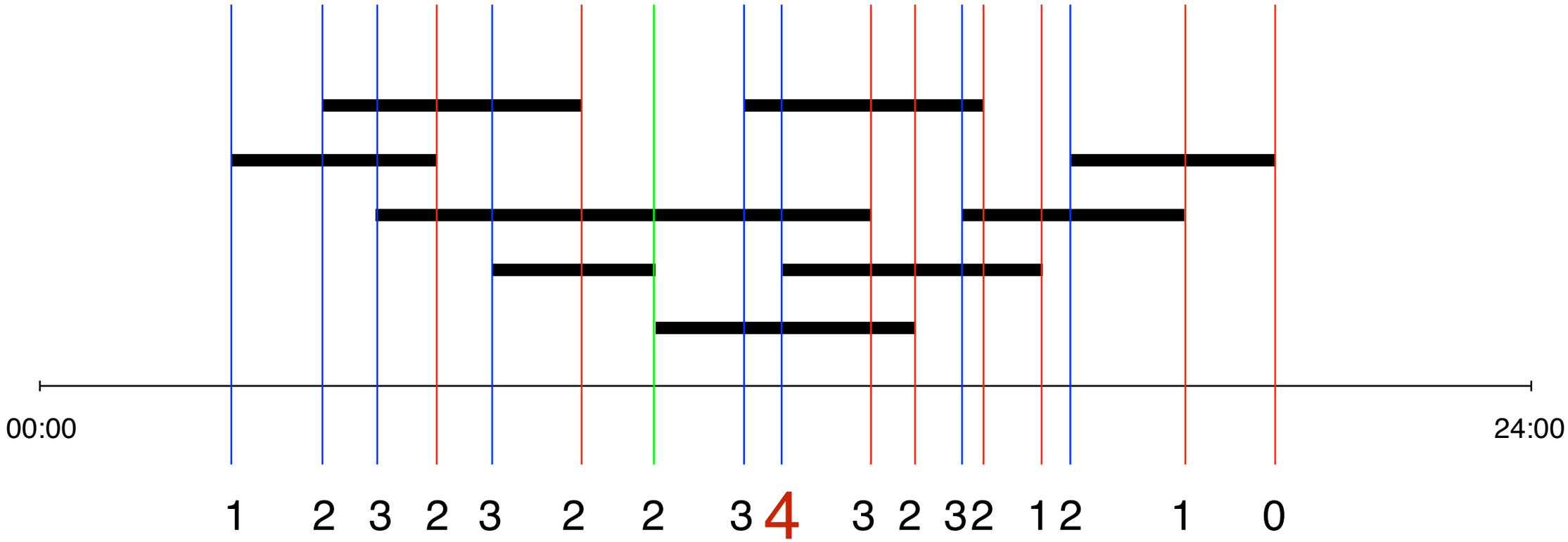
Platforms



동시에 최대 4대의 기차가 머무른다.

그렇다면 4개면 충분한가? 왜?

동시에 머무는 기차수 카운트하기



Plaforms

```
int solve(int n, int [] arrival, int [] departure)
{
   Arrays.sort(arrival);
   Arrays.sort(departure);
   int i=0, j=0, count = 0, maxCount = 0;
   while(i<n && j<n)  {
      if (arrival[i] < departure[j])  {
         count++; i++;
         if (count > maxCount)  maxCount = count;
      }
      else if (arrival[i] > departure[j]) {
         count--; j++;
      }
      else {
         i++; j++;
      }
   }
   return maxCount;
}
```

O(NlogN)

Alternative Solution

문제에서 데이터가 비교적 작은 정수일 경우
이런 방식의 알고리즘이 존재할 가능성이 크다.

```
int solve(int n, int [] arrival, int [] departure)
{
    int [] table = new int [2401];
    for (int i=0; i<n; i++) {
        table[arrival[i]]++;
        table[departure[i]]--;
    }
    int cur=0, max=0;
    for (int t=0; t<=2400; t++) {
        cur += table[t];
        if (cur > max)  max = cur;
    }
    return max;
}
```

0으로 초기화되어 있다고 가정한다.

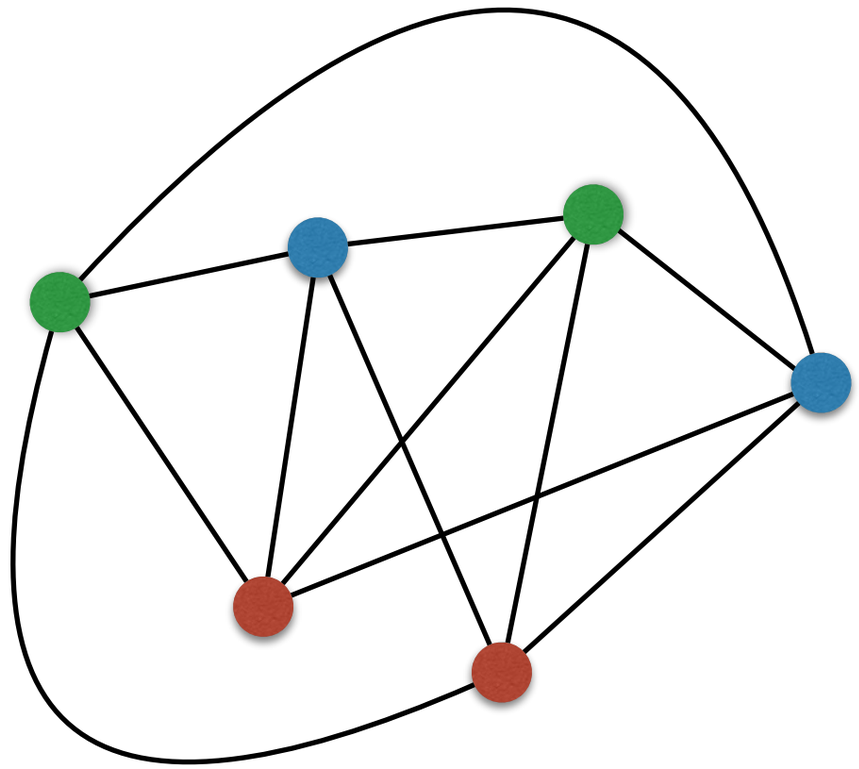
문제에 등장할 수 있는 서로 다른 시각은
총 2401가지에 지나지 않는다는 점을 이용했다.
시간이 매우 큰 임의의 정수로 표현된다면 이 방법은 사용할 수 없다.

$O(N)$

심화하기

Graph Coloring

Graph Coloring Problem



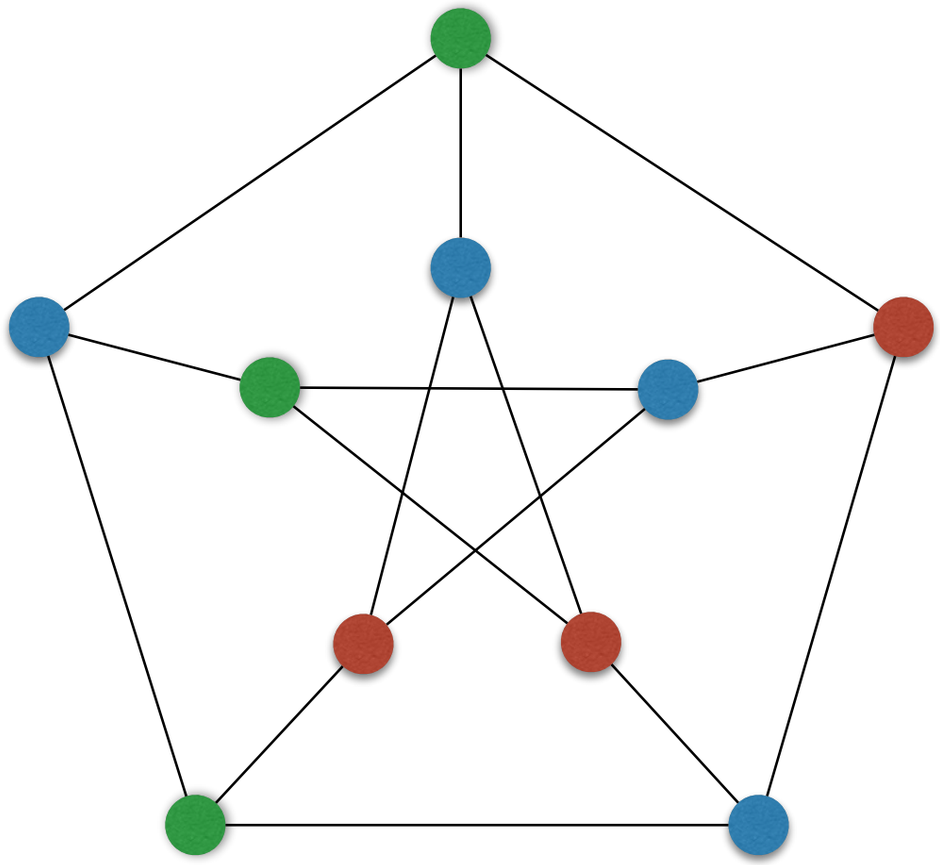
- 그래프의 각 정점을 색칠한다. 단, 인접한 정점은 서로 다른 색으로 칠해야 함
- k-coloring 문제
  - k개의 color로 모든 정점을 칠할 수 있는가?
  - **Decision problem** (답이 yes or no인 문제)
- **Min-coloring 문제**
  - 필요한 최소 color 갯수는?
  - 최적화(optimization) 문제
- **k-coloring 문제를 다항시간에 풀 수 있으면 min-coloring 문제도 다항시간에 풀 수 있다.**

어떻게?

최적화문제 대신 decision 문제를
먼저 생각해 본다.

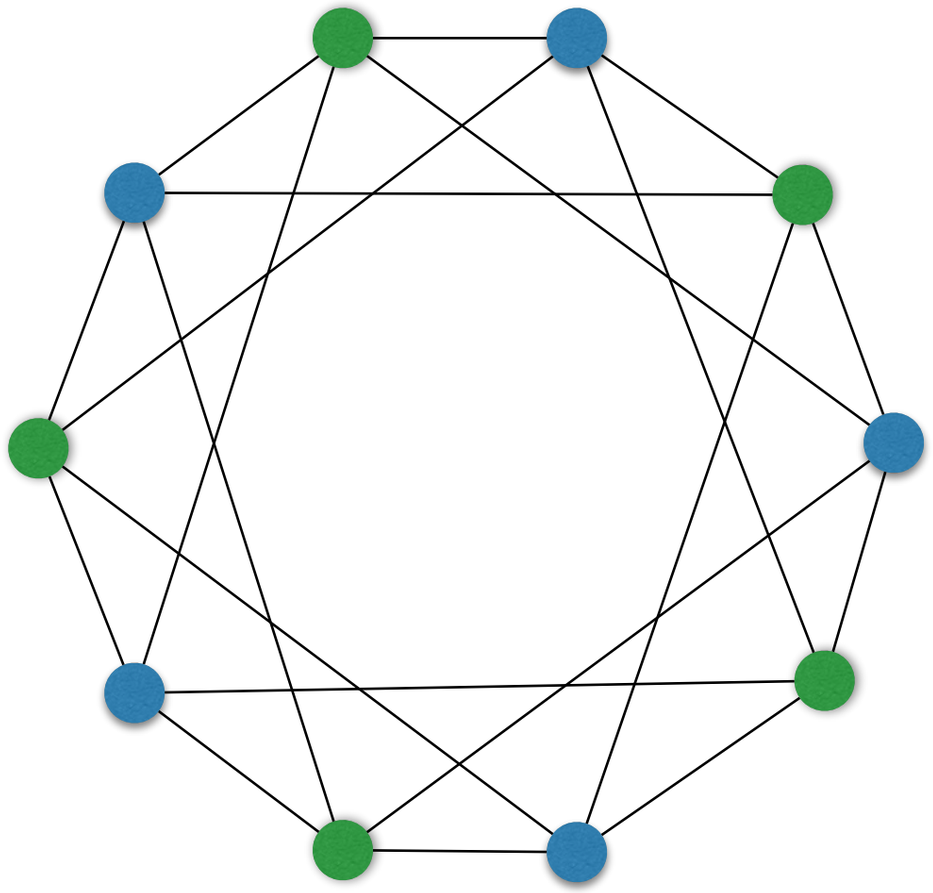
어려운 문제인가?

Greedy Method를 비롯하여 이런저런 시도를 해보자.
Optimal Substructure를 생각해보자.



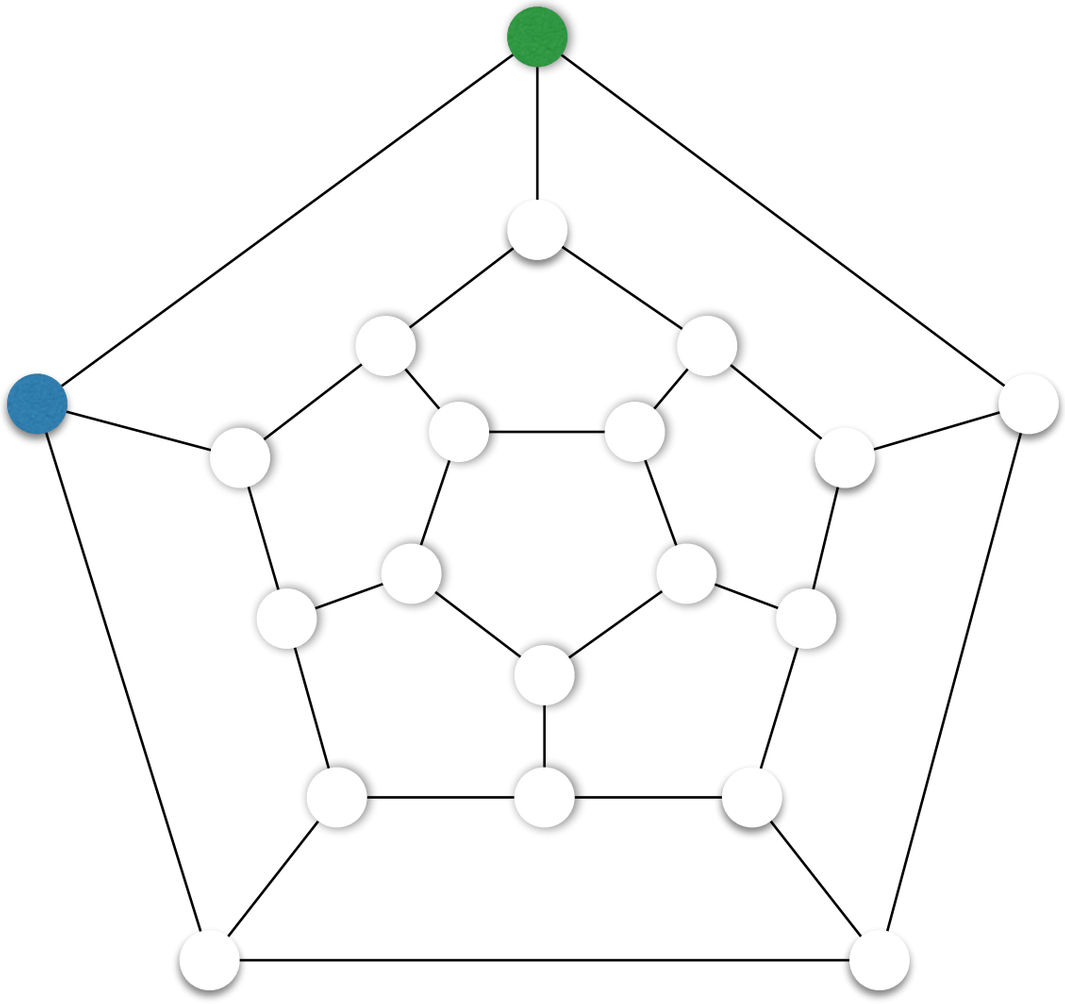
어려운 문제인가?

Greedy Method를 비롯하여 이런저런 시도를 해보자.
Optimal Substructure를 생각해보자.



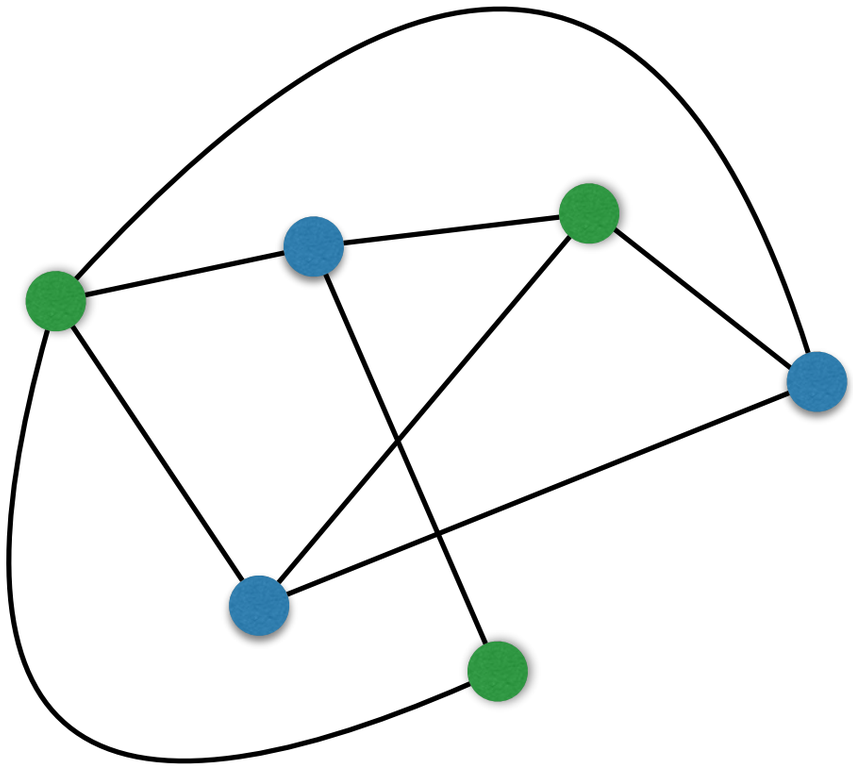
어려운 문제인가?

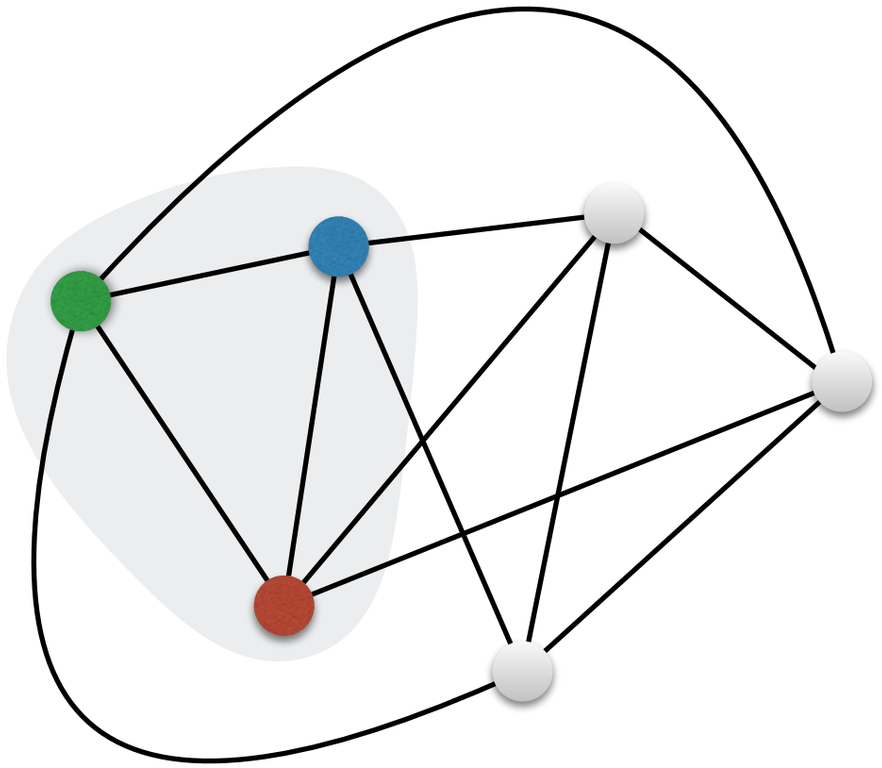
Greedy Method를 비롯하여 이런저런 시도를 해보자.
Optimal Substructure를 생각해보자.



Graph Coloring은 어려운 문제인가 ?

2-coloring의 경우





2-coloring은 불가능하다. 왜?

과감한 conjecture

2-coloring의 경우

“삼각형이 없으면 항상 2-colorable하다.”

?

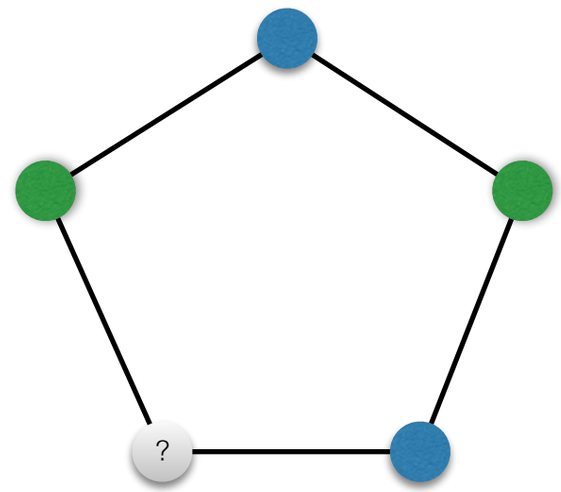
다시 과감한 conjecture

2-coloring의 경우

odd cycle이라고 부른다.

↓

“길이가 홀수인 사이클이 없으면 항상 2-colorable하다.”

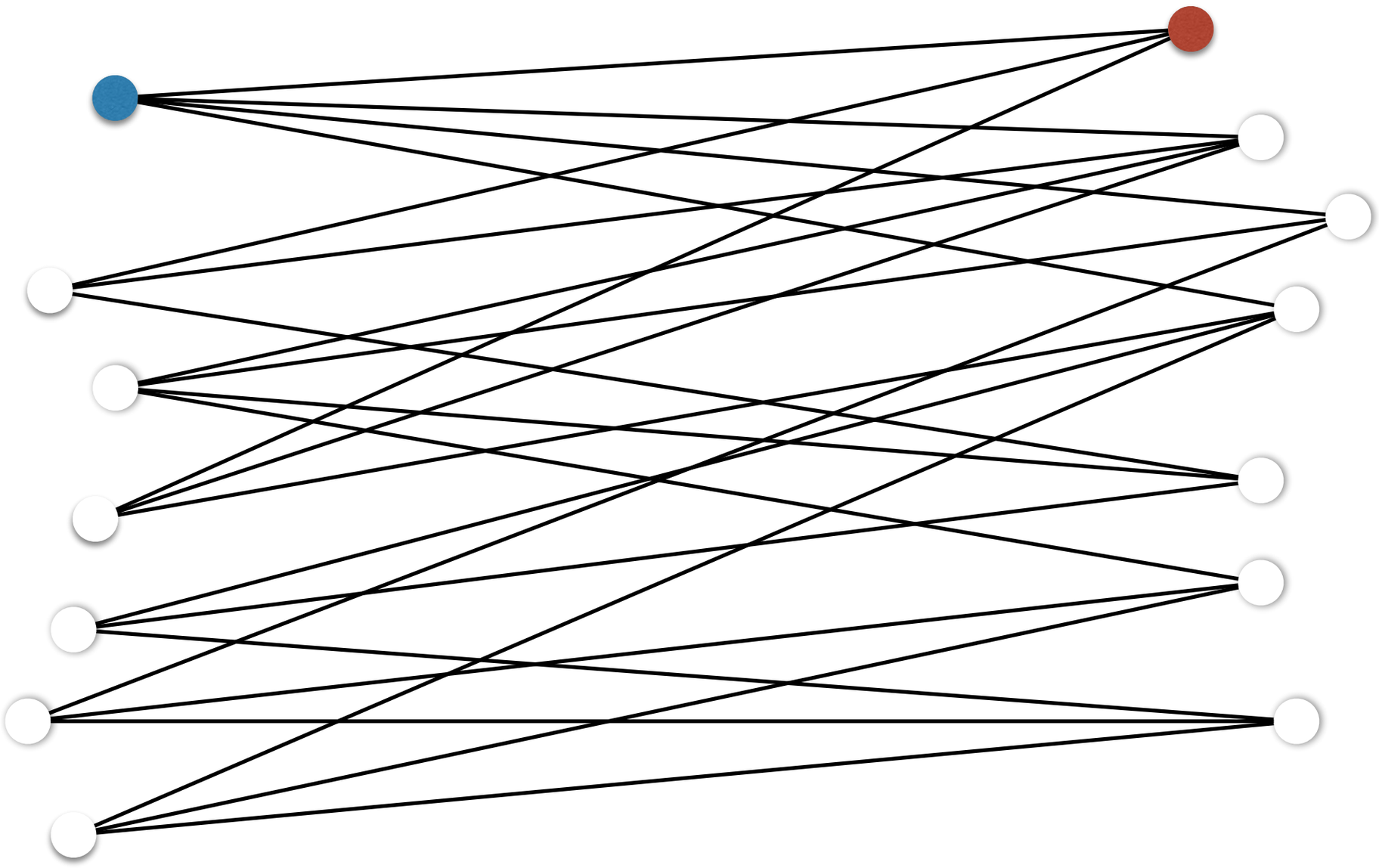


?

다시 과감한 conjecture

2-coloring의 경우

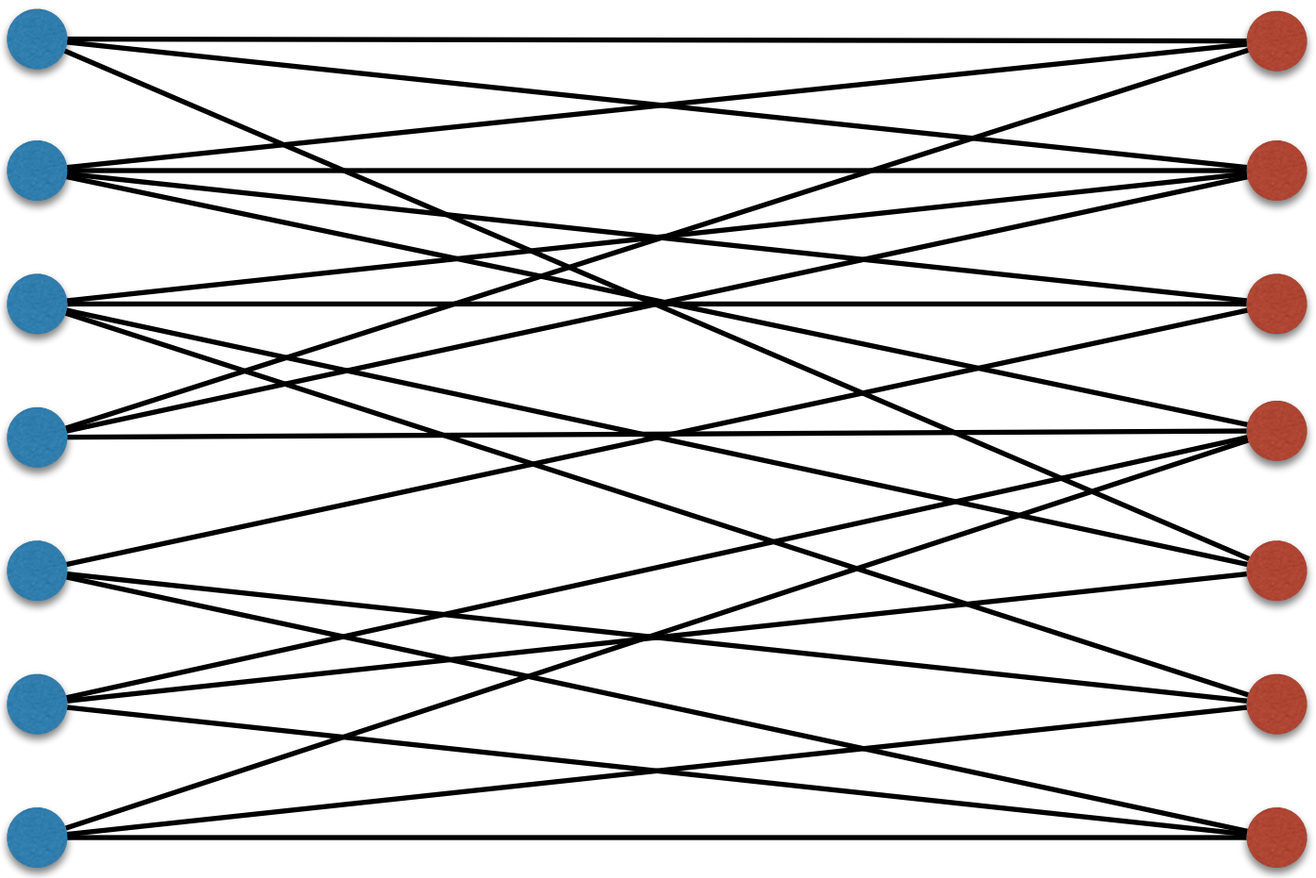
"홀수 사이클이 없으면 항상 2-colorable하다."



다시 과감한 conjecture

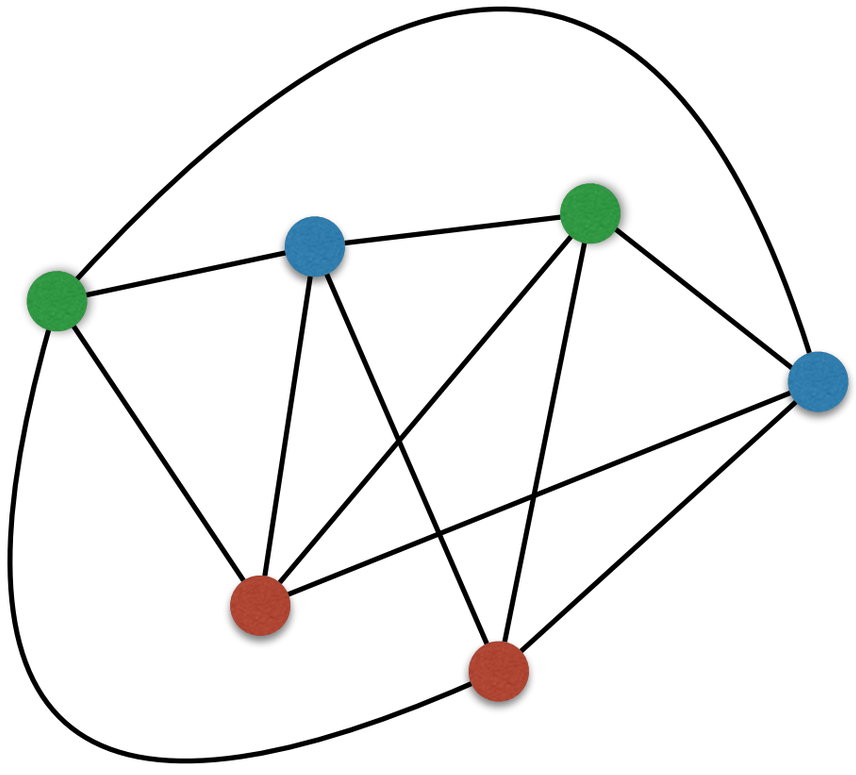
2-coloring의 경우

"2-colorable하면 홀수 사이클이 없다."



2-colorable한 그래프 ≡ 홀수 사이클이 없는 그래프
≡ Bipartite Graph

Graph Coloring Problem



최대 차수(degree): 각 정점에 인접한 정점의 개수의 최대값

- 그래프 G를 색칠하기 위한 최소 color수를 G의 chromatic number라고 부르고 $\chi(G)$로 나타낸다.

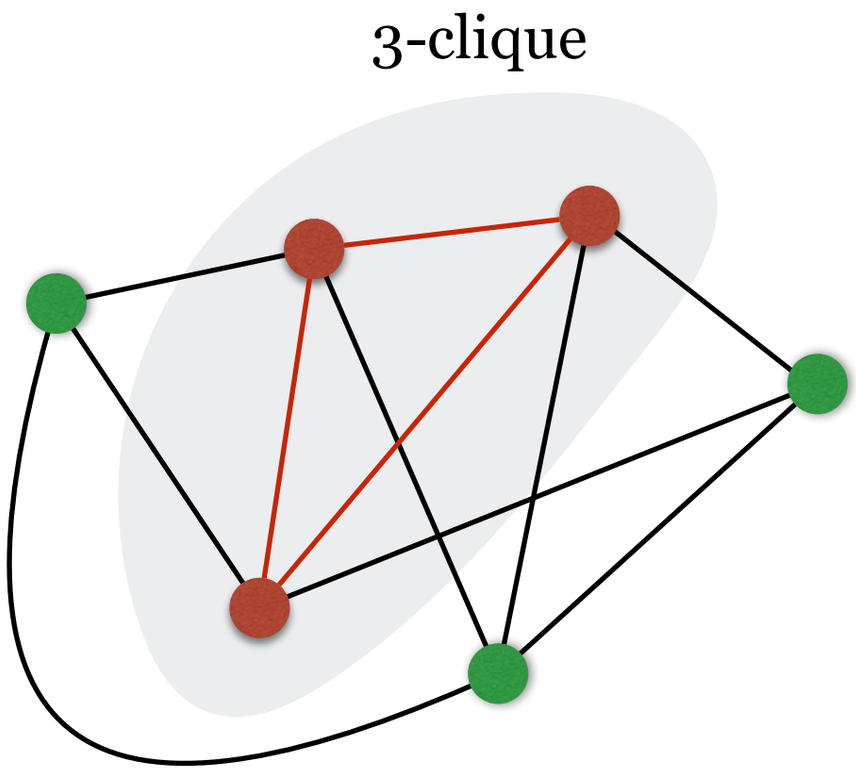
"카이(Chi)"라고 읽는다.

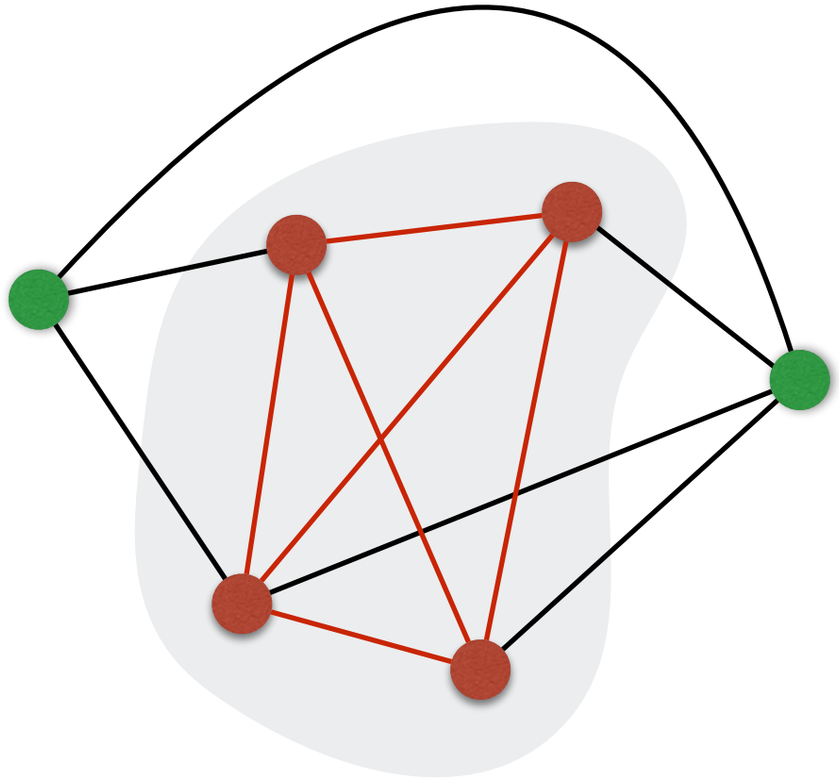
- $\chi(G) \leq d_{max} + 1$?
- K-coloring 문제는 K≧3인 경우 NP-complete 이다.

학기말 쯤에 이게 무슨 뜻인지 다룰 계획이다.
지금은 그냥 원래 어려운 문제 정도로 생각하자.

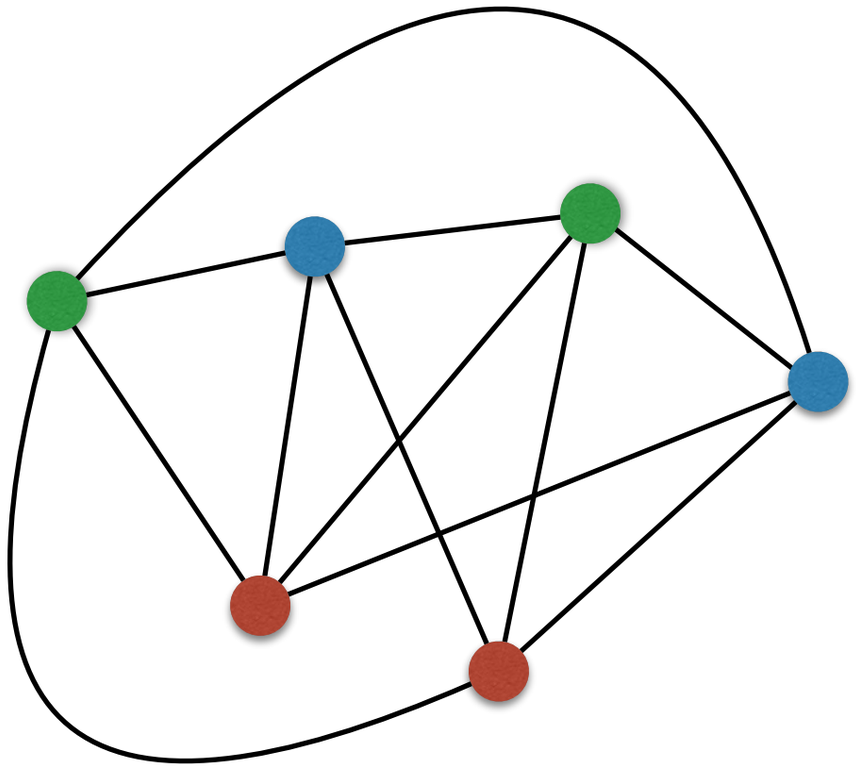
완전(complete) 그래프

- 완전 그래프는 모든 정점 간에 에지가 있는그래프이다.
- 어떤 그래프에서 정점의 개수가 k개인 완전 부그래프(subgraph)를 k-clique이라고 부른다.





Maximum Clique Problem

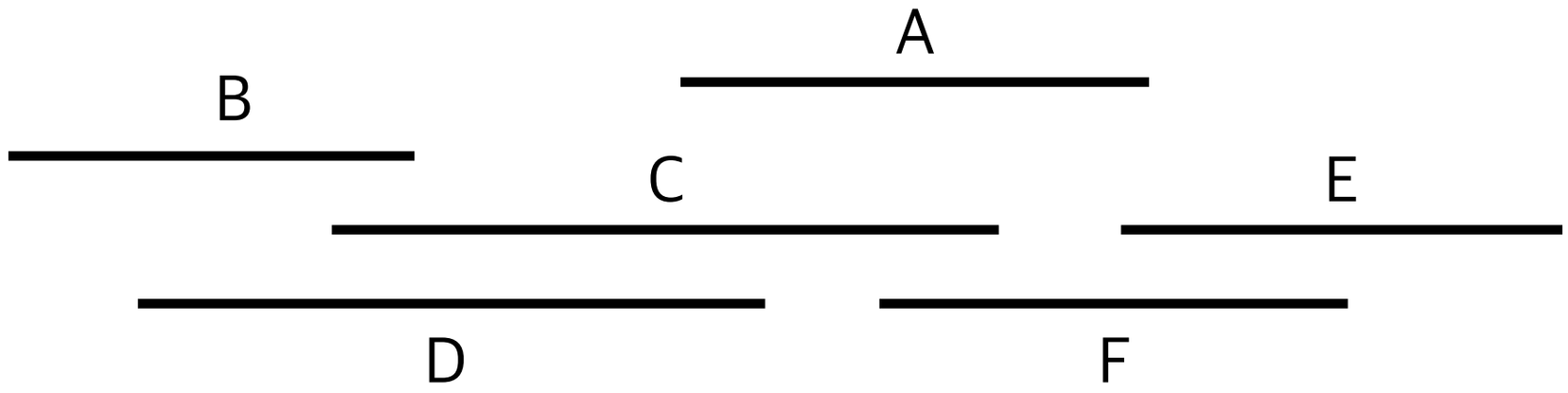


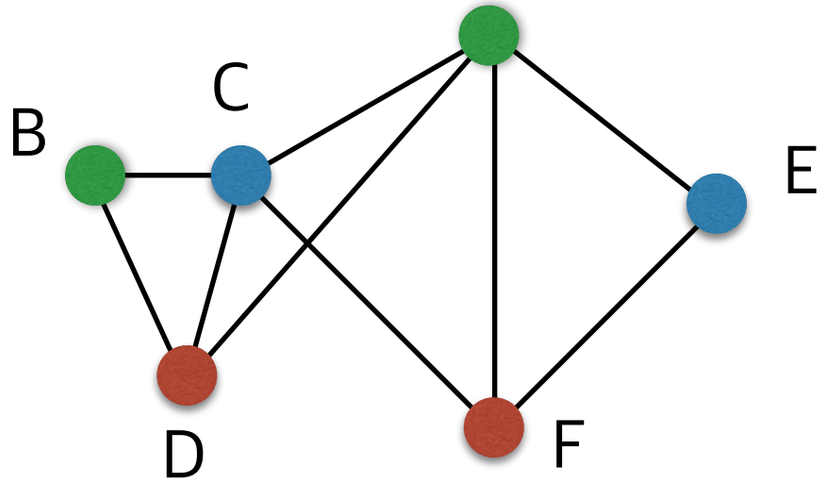
- 그래프 G에서 가장 큰 clique를 찾는 문제
- k-clique 문제는 크기가 k인 clique이 존재하는지 묻는 문제이다.
- 일반적인 그래프에서 k-clique문제는 NP-complete이다.

$\chi(G) \geq$ size of maximum clique

Platforms

이렇게 문제를 그래프와 같은 수학적 구조를 이용하여
표현해보는 것을 문제를 수학적으로 모델링한다 혹은 정형화한다고 말한다.





각 interval을 하나의 정점으로 표현하고 두 interval이 겹치면 해당하는 두 정점을 에지로 연결하여 만들어진 그래프를 interval graph라고 부른다.

**열차들을 최소 개수의 플랫폼에 배정하는 문제는
interval graph의 min coloring 문제와 동일하다.**

Interval Graph

- Interval 그래프에서 $\chi(G)$ = size of maximum clique 이다.
- Interval 그래프에서 coloring 문제는 다항시간 알고리즘을 가진다. Interval 그래프를 다항시간에 인터벌들의 집합으로 표현할 수 있다면…

예

- IT융합응용공학과에는 N명의 학생이 있다. 조별 과제를 하기 위해서 M개의 조가 짜여져 있다. 모든 학생들은 적어도 하나의 조에 속해 있고, 한 명의 학생이 여러 개의 조에 속해 있을 수 있다. 대학에서 주최하는 IT 전시회에 각각의 조들이 자신들의 작품을 전시해야 한다. 전시회는 K일 동안 진행된다. 각 조는 정확히 하루 동안 자신의 작품을 전시하며, 작품이 전시되는 동안 모든 조원이 전시 부스를 지켜야 한다. 따라서 어떤 학생이 2개의 조에 동시에 속해있으면 그 두 조는 서로 다른 날 전시를 해야 한다. 모든 조들이 무사히 전시를 마칠수 있는지 검사하는 프로그램을 작성하라.
- 모든 조가 정확히 2명으로만 구성된다면?