

고려대학교
프로그래밍
경시대회

KCPC 2020

문제 풀이 및 해설



Hello, KCPC!

Div2 A번

난이도: 브론즈 2


Alkor 김태훈

문제 내용

- 문자열이 있을 때,
- 정확히 “KCPC”인 부분이 등장하는
- 가장 오른쪽(뒤쪽) 위치를 찾는 것입니다.

더 오른쪽에도 있기
때문에 정답이 아님!

정답: 시작하는
위치인 10



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
?	?	K	C	P	C	?	?	?	K	C	P	C	?	?	K	C	P

풀이

- 뒤에서부터 보면서, “KCPC”와 동일한지 검사하면 됩니다.
- 만일 한 번도 그러하지 않으면, -1을 출력하면 됩니다.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
?	?	K	C	P	C	?	?	?	K	C	P	C	?	?	K	C	P

K	C	P	C
---	---	---	---

i = 15

풀이

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
?	?	K	C	P	C	?	?	?	K	C	P	C	?	?	K	C	P

K	C	P	C
---	---	---	---

$i = 14$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
?	?	K	C	P	C	?	?	?	K	C	P	C	?	?	K	C	P

K	C	P	C
---	---	---	---

$i = 10$

풀이

- 같다면, 해당하는 `i`를 출력하고 프로그램을 종료합니다.
- 만일 한 번도 그러지 않고 끝난다면, `-1`을 출력하면 됩니다.
- 대 소문자 구분과 문자열의 길이가 4보다 작을 때를 유의합니다.

```
#include <string>
#include <iostream>

int main() {
    std::string S;

    std::cin >> S;

    for(int i = S.size() - 4; i >= 0; i--)
        if(S[i] == 'K' && S[i + 1] == 'C' && S[i + 2] == 'P' && S[i + 3] == 'C')
        {
            std::cout << i+1 << std::endl;
            return 0;
        }

    std::cout << -1 << std::endl;

    return 0;
}
```



보일러 2

Div2 B번

난이도: 실버 2

ALPS 노세윤

문제

- i 번째 층 사람이 p_i 번째로 일어났다고 생각해봅시다.
- $p_i > p_{(i-1)}, p_i > p_{(i+1)}$ 이면 i 번째 층 사람은 보일러를 키지 않으며 그 외의 경우 i 번째 층 사람은 보일러를 켜야 합니다.
- 모든 가능한 순열 p_1, p_2, \dots, p_n 에 대하여 전체 일간 난방 비용의 기댓값을 구하는 것이 문제입니다.

풀이

- 전체 일간 난방 비용의 기댓값을 구하는 것이 아니라 각 층별로 난방 비용의 기댓값을 따로따로 구한 다음 더해도 됩니다.
- i 번째 층 사람이 보일러를 틀지 않을 확률을 구해봅시다.
- $i=1$ 이거나 $i=n$ 이라면 확률은 0입니다.
- $1 < i < n$ 인 경우 $p_i > p_{(i+1)}$, $p_i > p_{(i-1)}$ 일 확률을 구하면 됩니다.

풀이

- 3개 중에 3번째로 배치될 확률은 $1/3$ 입니다.
- 따라서 답은 $a_1 + a_n + (a_2 + a_3 + \dots + a_{n-1}) * 2/3$ 입니다.
- 시간복잡도: $O(n)$



보일러 1

Div2 C번, Div1 A번

난이도: 골드 5

ALPS 노세윤

문제

- N개의 칸으로 이루어진 파란색 배열을 주황색으로 칠할 것입니다.
- 이 때 다음 규칙을 지켜야 합니다.
 - 한 번에 한 칸만 칠할 수 있습니다.
 - 양 옆의 칸이 모두 주황색이라면 그 칸은 주황색으로 칠할 수 없습니다.
 - 더 이상 칠할 수 있는 칸이 없으면 종료합니다.
- 칠하는 순서를 잘 정해서 칠하기를 종료한 후 주황색으로 칠해진 칸에 있는 수의 합을 최소화하는 것이 문제입니다.

풀이



- 위와 같은 경우는 가능한 경우입니다.
- 어떤 경우가 불가능하고 어떤 경우가 가능한 걸까요?

풀이

- 파란색이 연속해서 존재할 수는 없습니다.
- 또한, 파란색이 연속하지 않은 다른 모든 경우는 가능합니다.
- 주황색 칸들을 인덱스가 증가하는 순서대로 칠하고 나면 파란색 칸은 칠할 수 없기 때문에 항상 유효한 construction이 있습니다.

풀이

- 이제 연속한 두 칸 중 하나는 무조건 골라야 할 때 고른 칸에 적힌 수의 합이 최소가 되도록 하는 문제가 되었습니다.
- DP로 풀 수 있습니다.
- $DP[i]$ =1부터 i 번째 칸 까지 고려했을 때 i 번째 칸을 칠했을 때 최소 합.
- $DP[i] = \min(DP[i-1], DP[i-2]) + a[i]$
- $DP[N]$ 을 출력하면 됩니다.
- 시간복잡도: $O(N)$

KCPC 2020

人古大 O

Div2 D번

난이도: 골드 4

Alkor 윤준혁

문제 내용

- M개의 정보가 주어졌을 때, 모순된 정보가 있는지 판단하는 문제입니다.
- 예를 들어 (1, 2), (2, 3), (3, 1)과 같이,
- 단 한 사람이라도 부먹과 찹먹의 성향을 모두 가지게 된다면 이 정보는 모순이라고 판단하면 되고,
- 모든 사람이 하나의 성향을 가질 수 있다면 모순이 아니라고 판단하면 됩니다.

풀이

- 핵심 알고리즘 dfs / bfs
- 간단한 풀이:
 - 주어지는 정보를 간선으로, 학생을 정점으로 하는 그래프를 생각합니다.
 - 그 그래프를 dfs 또는 bfs로 완전 탐색하여 모순이 있는지 찾습니다.

풀이

- 학생이 어떤 성향을 가지는지를 일차원 배열에 메모합니다.
 - 이때 일반성을 잃지 않고, 첫 학생의 성향은 임의로 지정해줄 수 있습니다.
- 주어지는 간선에 따라 dfs 또는 bfs로 모든 학생을 완전 탐색합니다.
- 방문한 정점과 연결된 다른 정점에 대해,
 - 그 정점이 이미 방문한 정점이라면, 다른 성향인지 확인합니다.
 - 서로 다른 성향이라면 정보에 모순이 없는 것입니다.
 - 만약 서로 성향이 같다면, 정보에 모순이 있는 것입니다.
 - 아니면 다른 성향으로 메모합니다.

정리

- 완전 탐색이므로 학생의 수와 정보의 수에 의해 시간 복잡도가 결정됩니다.
- 시간 복잡도 : $O(N+M)$
- 이와 같은 그래프를 그래프 이론에서는 “이분 그래프”라고 합니다.
- 이분 그래프의 정의는 아래와 같습니다.
 - 모든 정점을 빨강과 파랑으로 색칠하되, 모든 간선이 빨강과 파랑 정점을 포함하도록 색칠할 수 있는 그래프
 - https://en.wikipedia.org/wiki/Bipartite_graph



수로 설계

Div2 E번, Div1 B번

난이도: 골드 3

Alkor 윤준혁

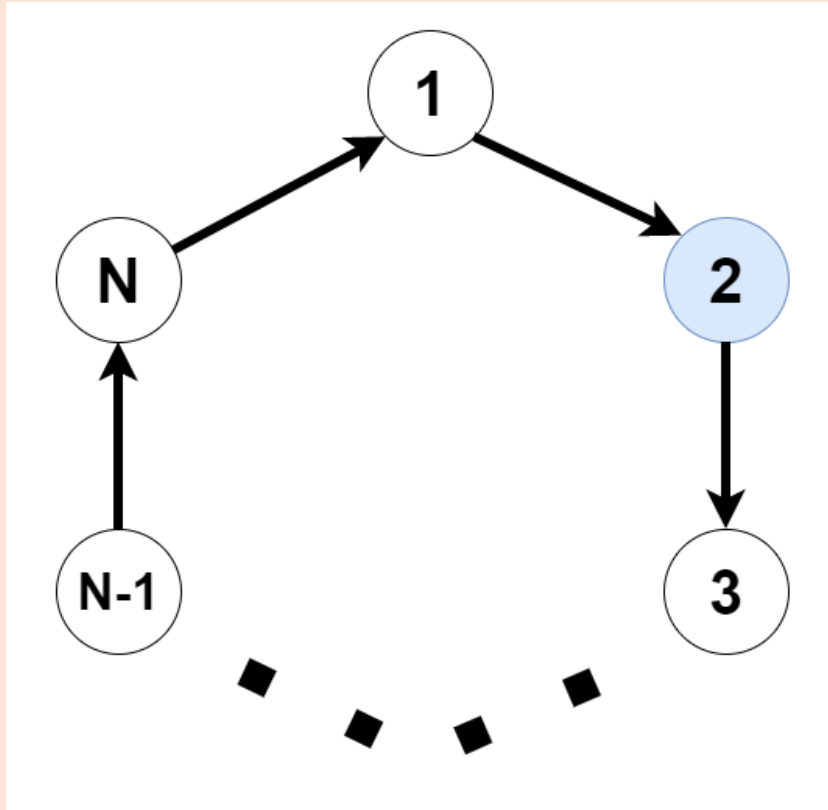
문제 내용

- 이 문제는 총 두 가지 소문제로 이루어져 있습니다.
 1. 방향 그래프에서 사이클 찾기 (이미 설치된 펌프에 의해 순환하는 수로 찾기)
 2. 사이클이 없도록 간선에 방향 부여하기 (펌프 설치)

풀이

- 핵심 알고리즘 : 위상 정렬
- 간단한 풀이:
 - 먼저 방향이 주어진 간선에 대해서 모든 정점을 위상 정렬 합니다.
 - 위상 순서에 따르지 않는 간선이 있다면 사이클이 존재하는 것입니다.
 - 방향이 정해지지 않은 간선에 대해서 위상 순서에 따르도록 방향을 정해줍니다.

풀이



- 위상 순서
- $2 \rightarrow 3 \rightarrow \dots \rightarrow N-1 \rightarrow N \rightarrow 1$
- $1 \rightarrow 2$ 는 위상 순서를 따르지 않는다.

풀이

- 방향이 주어진 간선을 따로 관리하여 그에 대해서만 모든 정점을 위상 정렬 합니다.
- 방향이 주어진 간선 중에 위상 순서에 따르지 않는 간선이 있는지 확인합니다.
- 그래프의 모든 간선의 방향이 위상 순서를 따르면, 사이클이 생기지 않음이 보장됩니다.
- 따라서 방향이 정해지지 않은 간선에 대해서는 위상 순서에 따르도록 방향을 정해주면 됩니다.

정리

- 위상 정렬은 dfs로 구현할 수 있기 때문에 dfs의 시간 복잡도에 따릅니다.
- 시간 복잡도 : $O(N+M)$



Finding Nemo

Div2 F번

난이도: 플래티넘 5

Alkor 이경렬

문제 내용

- N개의 점이 주어졌을 때, 모든 변이 x축 혹은 y축과 평행한 정사각형의 개수를 구하는 문제입니다.

풀이

- 먼저 다음과 같은 두 vector의 배열을 만들어 놓습니다.

$x[i]$: y좌표가 i 인 점들의 x좌표들을 넣는 vector

$y[i]$: x좌표가 i 인 점들의 y좌표들을 넣는 vector

그 다음, set에 입력으로 주어지는 좌표들을 넣어 특정 점이 있는지
검색하기 쉽도록 합니다.

풀이

- $x[i]$ 와 $y[i]$ 들을 모두 정렬하면 $x[i]$ 와 $y[i]$ 는 다음과 같이 됩니다.

$x[i]$: y 좌표가 i 인 좌표들이 x 좌표 순으로 정렬

$y[i]$: x 좌표가 i 인 좌표들이 y 좌표 순으로 정렬

풀이

- 입력으로 주어진 좌표들을 정렬한 다음 하나씩 순회하면서 이 점이 정사각형의 북동쪽 꼭짓점에 있는 정사각형의 개수를 세려면 다음 작업을 시행합니다.

현재 관찰하고 있는 좌표를 (now.x, now.y)라고 할 때,
x[now.y]에서 now.x보다 x좌표가 작은 점의 개수 num_x와
y[now.x]에서 now.y보다 y좌표가 작은 점의 개수 num_y를
이분탐색(lower_bound)으로 셉니다.

풀이

만약 num_x 가 num_y 보다 클 때, 즉 현재 관찰하고 있는 점에 대해 y 좌표는 같고 x 좌표는 작은 점의 개수가 x 좌표는 같고 y 좌표는 작은 점의 개수보다 많을 때, 정사각형을 탐색하려면 x 좌표는 같고 y 좌표는 작은 점들만 순회해야 시간단축을 할 수 있습니다.

반대의 경우에는 y 좌표는 같고 x 좌표가 작은 점들을 순회합니다.

풀이

num_x가 num_y보다 큰 상황에서는 현재 관찰하고 있는 점 기준으로 x좌표는 같고 y좌표는 작은 점들을 순회하면서 이 점과 현재 관찰하고 있는 점으로 정사각형을 만들 수 있는지 확인하려면 아까 만들어둔 set으로 이 점들로 정사각형을 만들기 위해 필요한 점들이 있는지 검색하면 됩니다.

num_x가 num_y보다 작아도 비슷하게 시행하면 됩니다.

풀이

- 이 풀이의 시간복잡도는

$$\Sigma(\min(\text{x좌표는 같고 y좌표는 작은 점의 개수}, \\ \text{y좌표는 같고 x좌표는 작은 점의 개수})) * \log N$$

으로 표현할 수 있습니다.

이를 최종적으로 계산하면 $O(N^{1.5} * \log N)$ 가 됩니다.



코로나 검사

Div1 C번

난이도: 플래티넘 4

ALPS 박홍빈

아이디어

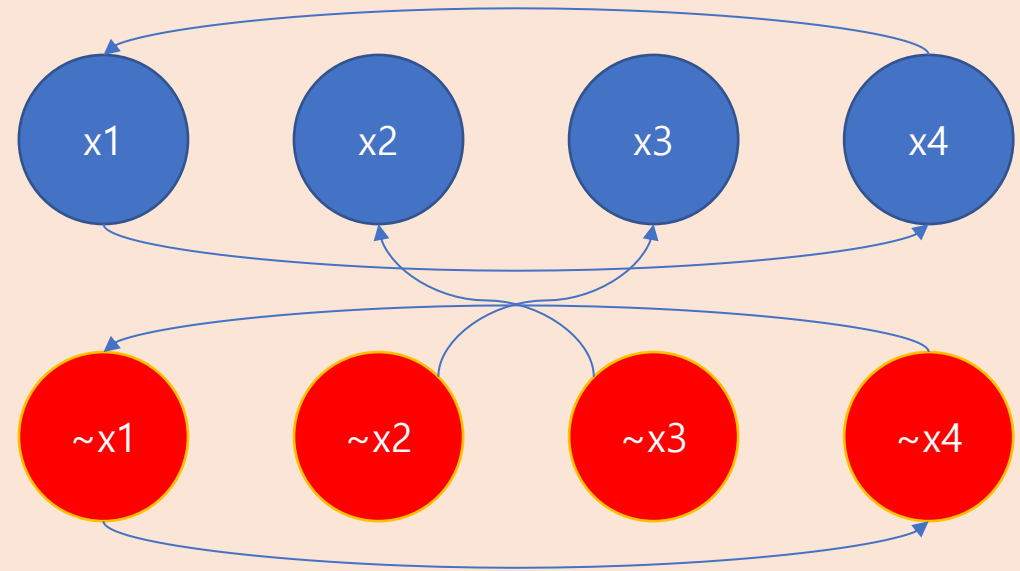
- 두 사람의 결과 중 하나는 반드시 참이므로 각 사람마다 bool 변수로 만들어 CNF를 구성
- 문제 조건을 통해 만든 CNF를 2-SAT 알고리즘을 이용하여 해결

풀이

- $3 - 6 \rightarrow (3 \mid -6)$
- $4 - 2 \rightarrow (4 \mid -2)$
- $(3 \mid -6) \ \& \ (4 \mid -2)$
- 확실한 감염자, 비감염자: $(2 \mid 2), (-3 \mid -3)$
- $F(x) = (x_1 \mid \sim x_4) \ \& \ (x_2 \mid x_3) \ \& \ (x_4 \mid \sim x_1)$

풀이

- $F(x) = (x1 \mid \sim x4) \& (x2 \mid x3) \& (x4 \mid \sim x1)$
- $(a \mid b) == (\sim a \rightarrow b) \& (\sim b \rightarrow a)$
- $F(x) = (x4 \rightarrow x1) \& (\sim x1 \rightarrow \sim x4)$
 $\& (\sim x2 \rightarrow x3) \& (\sim x3 \rightarrow x2)$
 $\& (x1 \rightarrow x4) \& (\sim x4 \rightarrow \sim x1)$
- 그래프를 SCC를 이용하여 해결





지하철 요금

Div1 D번

난이도: 플래티넘 1

Alkor 김태훈

아이디어 1/2

- 문제가 조금 길지만, 다음과 같이 요약할 수 있습니다.
- 양방향 그래프와, 시작점 도착점이 있습니다.
- 시작점에서 도착점까지 못 가도록 만들고 싶습니다.
- 이 때 최소 비용으로 정점을 선택하고 싶습니다.
- 여기서 이 문제가 Minimum-Cut임을 떠올릴 수 있습니다.

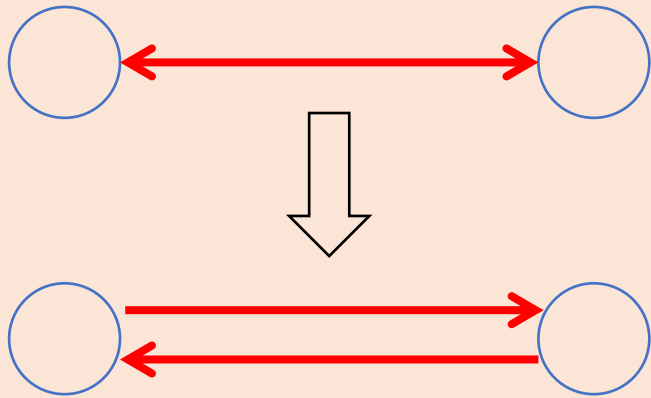
아이디어 2/2

- 근데, 각 정점을 막는데 드는 비용은 얼마일까요?
- Altair가 있는 정점에서 출발하여 막고 싶은 정점까지 가야 합니다.
- 따라서 각 정점의 비용은, Altair가 있는 점에서 최단경로 알고리즘을 통해 구한 최단비용입니다.

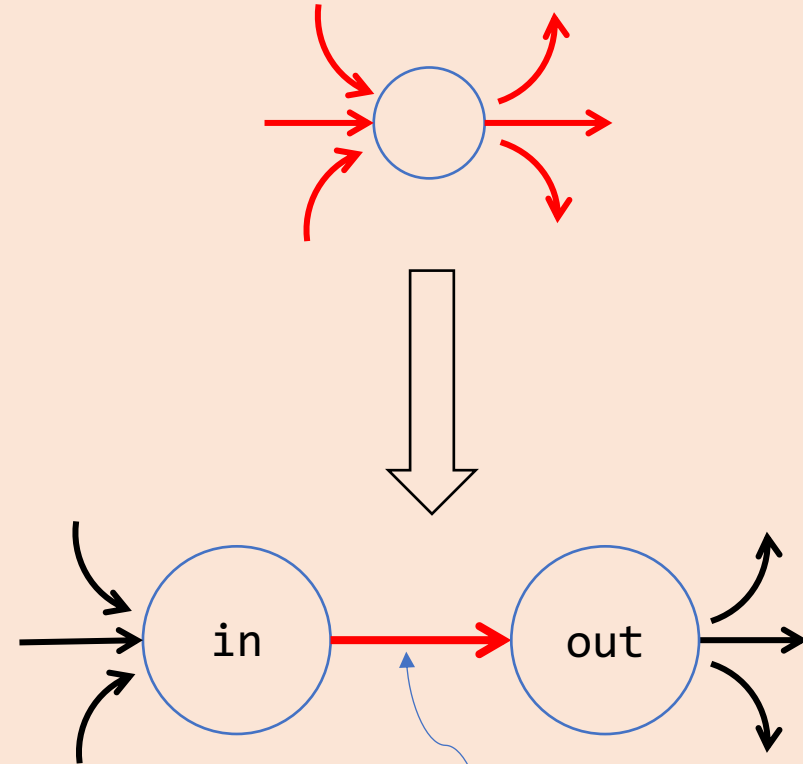
풀이

- 따라서, 먼저 Altair가 있는 정점에서 최단경로 알고리즘을 통해 각 정점의 비용을 알아낸 뒤,
- 정점을 막는 Min-cut을 구하고 싶기 때문에 Max-flow와 Min-cut이 같음을 이용하면 됩니다.
- 이 때 간선이 아닌 정점을 막는 것이기 때문에 정점을 in-node와 out-node로 쪼개는 것과, 양방향 간선이기 때문에 이를 단방향 2개로 나누어 생각하는 것이 필요합니다.

in-node와 out-node



먼저 양방향 간선을 이렇게
단방향으로 쪼갭시다.



정점까지 가는데
드는 최소비용

정리

- 1. Altair가 있는 점에서 시작하여 Dijkstra를 수행합니다.
- 2. 각 정점을 in-node와 out-node로 쪼개고, 그 사이를 1에서 구한 비용의 용량을 가지는 간선으로 잇습니다.
- 3. 원래 있었던 간선들에 대해 각각을 2개의 단방향 간선으로 생각하고, 2에서 쪼갠 정점에 맞추어 무한대의 용량을 가지도록 만듭니다.
- 4. SOURCE 노드를 시작점의 in-node와, SINK 노드를 도착점의 out-node와 무한대의 용량으로 잇습니다.
- 5. 적절한 Max-Flow 알고리즘을 돌려 포상금 C에서 빼줍니다.
 - 이 때, max flow가 C보다 크면, 0을 출력해야 함에 유의합니다.



종이접기

Div2 E번

난이도: 플래티넘 1

Alkor 이경렬

문제 내용

- 종이가 접히는 것을 구현하고, 접힌 종이 구간의 두께 합을 빠르게 계산하는 문제입니다.

풀이

기본적으로 “2 a b”의 형태로 들어오는 쿼리를 처리하기 위해서 구간합 배열 테크닉을 활용합니다. 즉, 이 문제의 핵심은 종이 접히는 과정을 구간합 배열에 어떻게 반영할 것인가에 대해서 관찰을 해야 합니다. 우선적으로 현재 어느 구간에 종이 존재하는지 표현하기 위해 종이의 가장 왼쪽 지점 ‘l’과 가장 오른쪽 지점 ‘r’을 관리해야 합니다.

풀이

문제에서 어디를 접을지 혹은 어느 구간의 두께 합을 계산할지 정해주는 인덱스는 항상 0에서 시작하지만, 실제 종이는 '1'에서 시작하기 때문에 이에 유의해야 합니다.

만약, $\text{sum}[i] = \text{sum}[i-1] + \text{arr}[i]$ 의 형태로 만들어진 구간합 배열을 가지고 있을 때, '일반적인 상황에서' 구간 (a,b) 의 두께 합을 계산하려면 $\text{sum}[l+b] - \text{sum}[l+a]$ 를 계산해야 합니다.

풀이

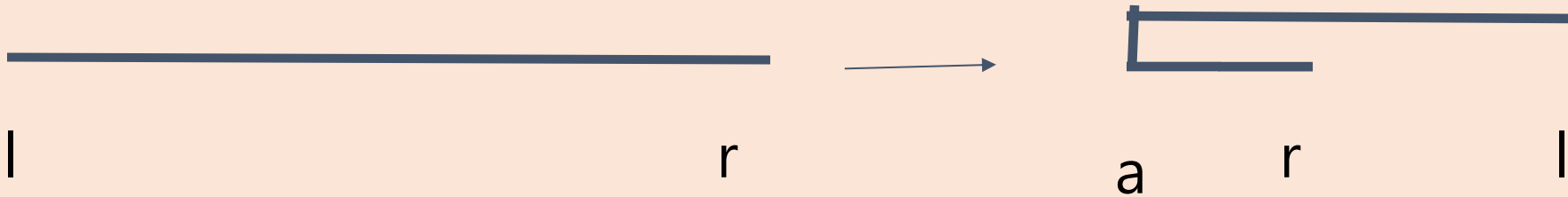
여기서 'l'은 'r'보다 항상 왼쪽에 있지 않음에 주목해야 합니다. 접는 행동을 수행할 때, 접어야 하는 위치가 종이의 중간보다 오른쪽이라면, 'l'과 'r'의 위치 관계는 역전됩니다. 이를 관리하기 위해 변수 flag를 선언하고, 접는 위치 'a'가 $(r-l)/2$ 보다 크면 1과 xor하는 방식으로 flag를 관리하면 편합니다.

풀이

$(r-l)/2 < a$ 를 chk 라고 할 때, $flag \wedge chk$ 가 참인 경우,
즉 $flag$ 가 1이고 chk 가 0이거나 // $flag$ 가 0이고 chk 가 1인 경우에
 a 지점에서 종이를 접어야 한다면,
 (a, r) 구간의 $arr[i]$ 값을 a 지점을 기준으로 대칭되는 구간에
더해야 합니다.

풀이

- 즉, 이를 그림으로 표현하자면



다음과 같습니다. (flag=0, chk=1)

이 때, 구간 (a, r) 의 값을 구간 $(2a-r, a)$ 구간에 더해야 합니다.

그리고 이를 구간합 배열 $sum[i]$ 에 반영하고 r 을 a 로 바꿉니다.

풀이

$\text{flag}=1, \text{chk}=0$ 인 경우에도 마찬가지로 구간 (a, r) 의 값을 구간 $(2a-r, a)$ 구간에 더해야 한다는 사실을 관찰할 수 있습니다.

$\text{flag}=\text{chk}$ 인 경우에 종이가 어떻게 접혀야 하는지를 관찰하면 구간 (l, a) 의 값을 구간 $(a, 2a-l)$ 구간에 더해야 하고, 이를 구간합 배열에 반영하고 l 을 a 로 바꾸면 됩니다.

풀이

이 풀이의 시간복잡도를 계산하자면,
2번 행동을 시행할 때에는 단순히 구간합 배열의 차이를 계산
하면 되므로 $O(1)$ 만큼 걸립니다.

1번 행동을 시행할 때에는 $O(N)$ 만큼 걸려 $O(N*Q)$ 처럼 보일 수
있으나 1번 행동을 하면 할수록 N 의 크기가 줄어들기에
시간복잡도를 계산하면 $O(N)$ 이 나오게 됩니다.



팀 차이

Div1 F번

난이도: 다이아몬드 4

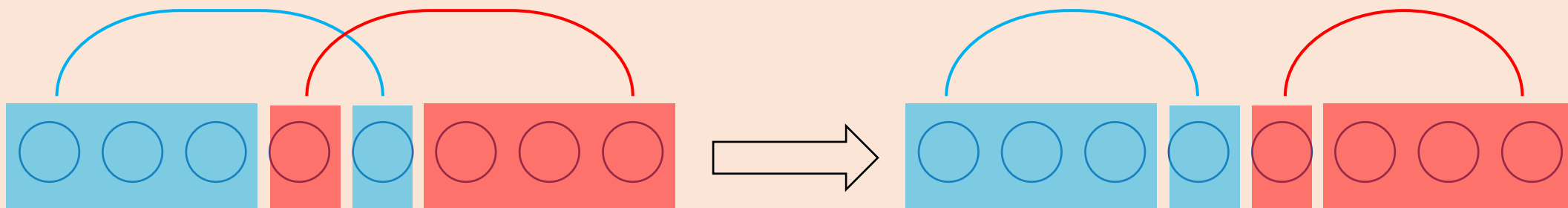
Alkor 김태훈

문제

- N개의 정수가 있습니다.
- 이를 K개로 중복없고 빠짐없이 나눌 겁니다.
 - 각 묶음에는 최소 2개의 정수가 들어가야 합니다.
- 각 묶음마다, 최대값과 최소값의 차를 구합니다.
- 이 차이들을 더한 값을 최소화 하고 싶습니다.
- 어떻게 해야 할까요?

관찰 1/3

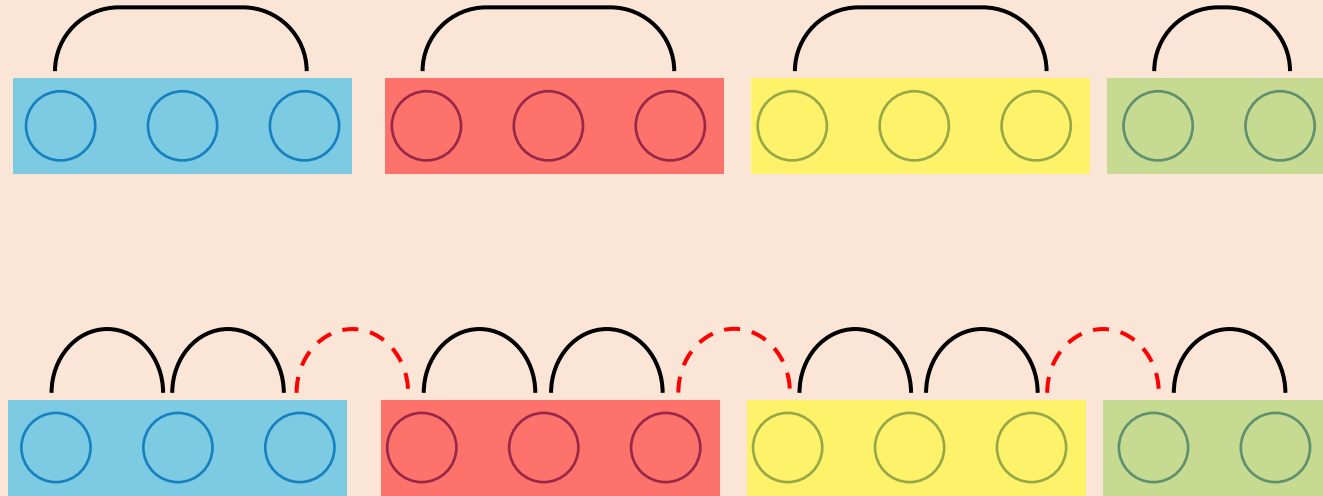
- 먼저, 정렬을 하는 것이 좋아 보입니다.
- 그러면, 모든 묶음은 정렬된 “순서대로” 뽑혀야 함을 알 수 있습니다.
- 예를 들어 왼쪽보다는 오른쪽이 Greedy하게 좋습니다.



관찰 1.5 / 3

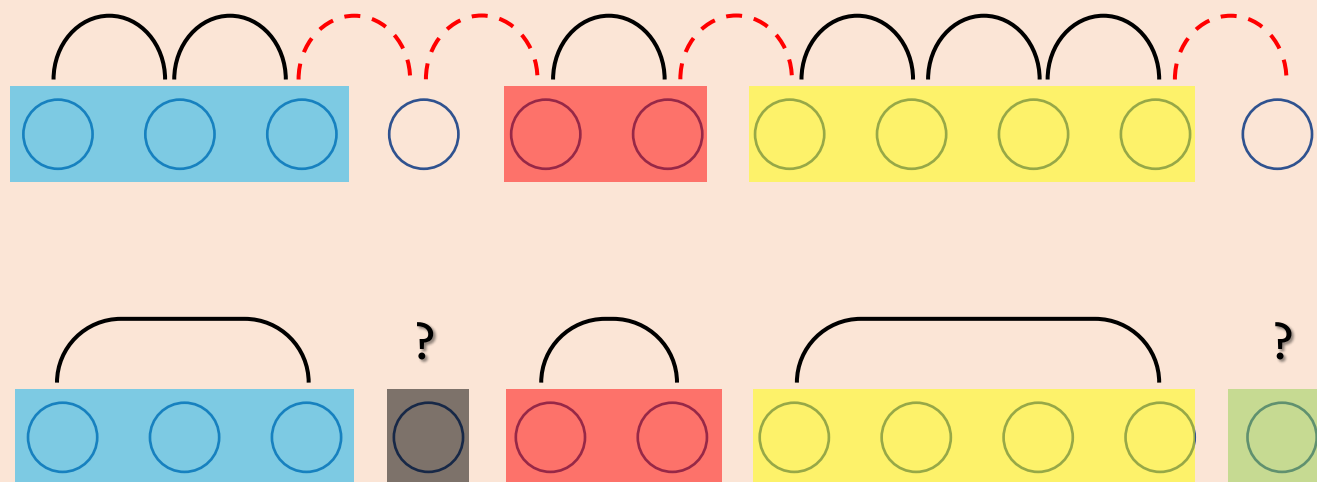
- 그렇다면? $O(N^2)$ DP를 생각해볼 수 있습니다.
- i 번째 index까지를,
 - j 개로 나눌 때의 정답을 구할 수 있습니다.
- $dp[i][j] = \min(dp[i - 1][j], dp[i - 2][j - 1]) + (arr[i] - arr[i - 1])$
- 원래는 이와 같은 N^2 DP로 낼 계획이었습니다...
 - 하지만 여기서는 시간 초과를 받습니다. 또다른 관찰이 필요합니다.

관찰 2/3



- 구하려는 값은 원래 수열의 (최대 – 최소) 에서,
- 인접한 수의 차이 $(N - 1)$ 개 중 $(K - 1)$ 개를 골라 뺀 값입니다.

관찰 3/3

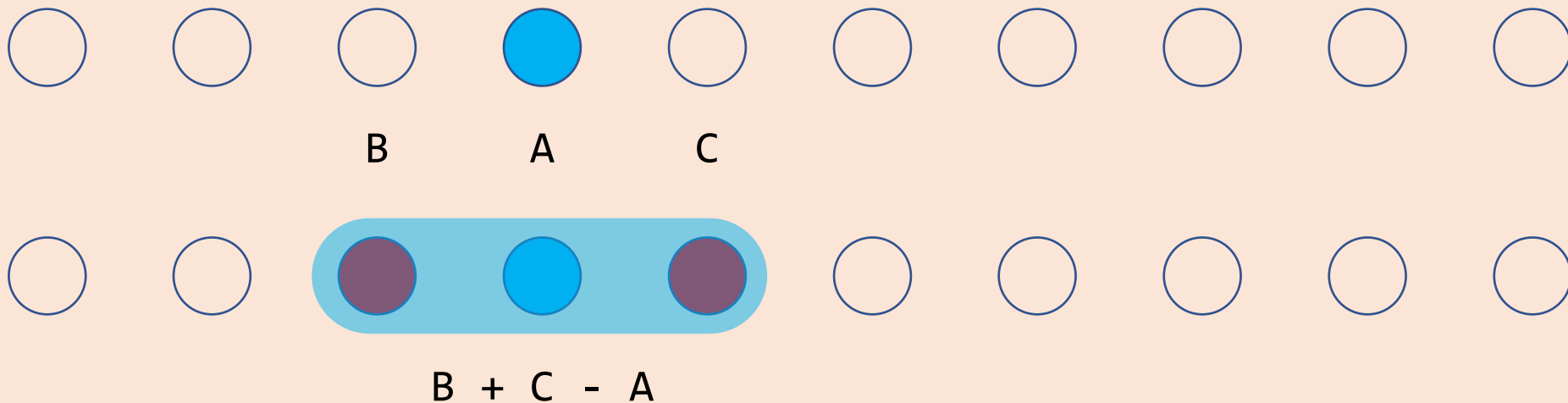


- 그런데 한 묶음에는 최소 2개의 정수가 있어야 하기 때문에,
- 삭제할 “차이”는 인접하여 선택할 수 없습니다.
- 또한 양 끝의 차이는 선택할 수 없습니다.

문제 변환

- 최소를 구해야 하기 때문에, 삭제할 $(K - 1)$ 개의 차이는 최대한 골라야 합니다.
- 즉, $(N - 3)$ 개의 정수 중 인접하지 않도록 $(K - 1)$ 개의 정수를 합이 최대가 되도록 고르는 문제로 바뀌었습니다.
- 이는 백준 1150번에도 있는 Greedy 문제입니다.
 - Alien Trick으로도 풀립니다. (DP 최적화이기 때문에..)

풀이 1/3



- 만일 이 묶음을 선택한다면?
 - A만 선택 \rightarrow B, C를 선택
 - 선택하는 개수는 +1, 비용은 $A \rightarrow B + C$ 이므로 추가로 드는 비용은 $B + C - A$
- 따라서 계속 최대한 정수를 Greedy 하게 고르면 됩니다.

풀이 2/3



B

A

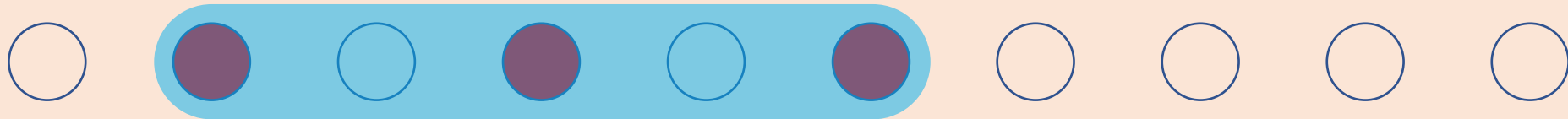
C



D

$B + C - A$

E



$D + E - B - C + A$

풀이 3/3

- 정수들은 Set으로 관리하고, 고를 때마다 양옆의 node와 합쳐야 합니다.
 - 합칠 때 제거가 필요하기 때문에, set을 써야 합니다.
- set 2개로 관리해도 좋고, set 하나와 양 끝을 저장하는 배열을 사용해도 좋습니다.
- 참고) Alien Trick은 묶음이 쪼개질 때마다 가중치 x 를 추가로 더해, 단순 $O(N)$ DP를 할 때 K 개의 묶음이 가장 적합해지는 x 를 이분탐색으로 찾는 방법입니다.

정리

- Greedy나 Alien Trick 모두 $O(N \log N)$ 이 걸립니다.
- 1. 수열을 먼저 정렬하고,
- 2. 최대 – 최소를 저장해두고,
- 3. 인접한 차이들을 구한 뒤 따로 빼서
- 4. Greedy나 Alien Trick을 돌립니다.
- 5. 2에서 구한 값에서 빼면 됩니다.



수열 쏘개기

Div1 G번

난이도: 다이아몬드 2

ALPS 노세윤

문제

- 수열 A_1, A_2, \dots, A_n 이 주어질 때
- $\text{Cost}[i][j]$ = 부분수열 $A[i \dots j]$ 에 존재하는 서로 다른 수의 개수
- $\text{DP}[i][j]$ = 부분수열 $A[1 \dots j]$ 를 i 개의 수열로 쪼갬을 때 최대 가치
- $\text{DP}[k][n]$ 을 계산하는 문제입니다.

풀이

- 아래와 같은 간단한 점화식을 얻을 수 있습니다.
- $DP[i][j] = \min(DP[i-1][l] + cost[l+1][j]) \quad l < j$
- 이를 단순히 계산하면 $O(n^3)$ 입니다.
- 최적화를 해 봅시다.

풀이

- $a \leq b \leq c \leq d$ 인 정수 a, b, c, d 에 대하여
- $\text{Cost}[a][d] + \text{Cost}[b][c] \leq \text{Cost}[a][c] + \text{Cost}[b][d]$ 가 성립합니다. (사각부등식)
- 따라서 Alien Trick을 쓸 수 있습니다.
- 점화식이 $\text{DP}[i] = \min(\text{DP}[j] + \text{Cost}[j+1][i] - X) \quad j < i$ 로 바뀌었습니다.
- 점화식을 $O(n^2)$ 에 계산할 수 있으므로 $O(n^2 \log n)$ 에 풀 수 있습니다.

풀이

- 더 빠르게 풀 수 있을까요?
- 임의의 i, j 에 대한 $Cost[i][j]$ 를 persistent segment tree를 이용해 $O(\log n)$ 에 계산할 수 있으므로 Monotone Queue Optimization 을 적용해 $O(n * \log^3 n)$ 에 풀 수 있지만 전부 상수가 큰 로그라 어렵도 없습니다.

풀이

- 세그먼트 트리를 도입합니다.
- 최대값 세그먼트 트리의 j 번째 리프노드에 $DP[j-1]+Cost[j][i]$ 를 넣어놓으면 루트노드를 확인함으로써 $DP[i]$ 를 바로 계산할 수 있습니다.
- 문제는 i 가 $i+1$ 로 바뀌면 $Cost[j][i]$ 도 바뀐다는 것입니다.
- 그러나 관찰을 조금 해보면 $Cost[j][i] \neq Cost[j][i+1]$ 인 j 는 연속적으로 분포함을 알 수 있습니다.

풀이

- $a[j]=a[i]$ 인 i 보다 작은 최대 j 를 P_i 라고 합시다.
- 구간 $(P_i, i]$ 에는 $a[i]$ 가 단 하나 존재합니다.
- 따라서 $(P_i, i]$ 에 존재하는 j 에 대하여 $Cost[j][i-1]+1=Cost[j][i]$ 입니다.
- 또한 $[1, P_i]$ 에 존재하는 j 에 대하여 $Cost[j][i-1]=Cost[j][i]$ 입니다.
- 즉 $DP[i]$ 을 계산하기 전에 구간 $(P_i, i]$ 에 lazy propagation으로 1을 더해주면 됩니다.
- 시간복잡도: $O(n \cdot \log^2 n)$