2019 숭.고.한 연합
Algorithm Camp

2019 숭고한 연합 Algorithm Camp Contest 풀이

대회 일자 : 2019년 8월 9일

풀이 배포 일자 : 2019년 8월 10일

숭고한 초급반 A

와글와글 숭고한

- 출제자 : 이상헌 / 분야 : 자유 주제 (입문)
- 제출 19회, 정답 15명
- 처음 풀린 시간 : 3분

풀이

와글와글 숭고한

- 세 정수를 입력받아, 문제 설명 그대로 하면 됩니다.
- 세 수의 최소를 찾는 건 `if (a < b and a < c)` 등의 조건문으로 확인 가능합니다.
- 문제의 모티브는 '와글와글 던전'이라는 보드게임입니다. 한 번 해보세요!

숭고한 초급반 B

보너스 점수

- 출제자 : 김민성 / 분야 : 자유 주제 (입문)
- 제출 19회, 정답 15명
- 처음 풀린 시간 : 7분

풀이

보너스 문제 – 로직

	O	O	X	O	X	O	O	O	O
기본 점수	1	2	0	4	0	6	7	8	9
보너스 점수	0	1	0	0	0	0	1	2	3
누적 점수	1	1+(2+1) = 4	4+(0+0) = 4	4+(4+0) = 8	8+(0+0) = 8	8+(6+0) = 14	14+(7+1) = 22	22+(8+2) = 32	32+(9+3) = 44

T번째 문제에서 얻는 기본 점수 = 맞았으면 T점, 틀렸으면 0점

T번째 문제에서 보너스 점수 = 이번에도 맞았고 저번에도 맞았으면 가장 최근 보너스 + 1 점, 아니면 0점 (더 쉽게 계산할 수도 있다! 풀이 참조)

T번째 문제까지의 누적 점수 = 여태까지 얻은 모든 기본 점수와 보너스 점수의 합

시간복잡도 = O(문제 개수)

풀이

보너스 문제 - 코드

1. 기본 점수(cnt), 보너스 점수(yes), 누적 점수(ans)가 모두 0인 상태로 시작.

2. 기본 점수가 1점 올라간다.

3-1. 문제를 맞췄다면 누적 점수가 (기본 점수 + 보너스 점수) 만큼 **올라간 뒤**, 보너스 점수가 1점 올라간다.

3-2. 문제를 틀렸다면 보너스 점수가 0점이 된다.

4. Step 2로 돌아간다.

```
length = input()
ans, yes, cnt = 0, 0, 0
for ch in input():
    cnt = cnt + 1
    if ch == 'O':
        ans = ans + cnt + yes
        yes = yes + 1
    else:
        yes = 0
print(ans)
```

숭고한 **초급반 C, 중급반 A**

이건 꼭 풀어야 해!

- 출제자 : 권욱제 / 분야 : PS 기초
- 초급반 : 제출 49회, 정답 15명
- 중급반 : 제출 57회, 정답 22명
- 처음 풀린 시간 : 16분(초급반), 3분(중급반)

풀이

이건 꼭 풀어야 해!

- 문제의 포인트는 크게 2개입니다.
  - 정렬 : 표준 정렬 라이브러리를 사용하면 됩니다. (시간 복잡도 O(N lg N))
  - 구간합 : prefix sum을 이용하여 구할 수 있습니다. (쿼리당 시간 복잡도 O(1))

- 해당하는 테크닉을 사용할 수 있는지를 묻는 문제였습니다.
  - 버블 정렬 구현해서 짜고 계신 분이 있었습니다. 출제진을 안타깝게 했습니다.
  - 지문 노트에 나이브하게 구간합을 구하면 느리다고 알렸음에도 그러시는 분들이 있었습니다. 역시 안타까웠습니다.

숭고한 초급반 D, 중급반 B

무한부스터

- 출제자 : 성창호 / 분야 : DP 초급
- 초급반 : 제출 22회, 정답 4명
- 중급반 : 제출 43회, 정답 18명
- 처음 풀린 시간 : 47분(초급반), 23분(중급반)

풀이

무한부스터

- 다양한 방식의 동적 계획법을 이용해 시간 복잡도 O(nm(n+m))에 풀 수 있습니다.
  - `dp[i][j]` : (1, 1)에서 (i, j)로 가는데 밟아야 하는 최소 칸 수, `dp[1][1] = 0`
  - `dp[i][j]` : (i, j)에서 (n, m)으로 가는데 밟아야 하는 최소 칸 수, `dp[n][m] = 0`

- 본질적으로는 최단 거리 문제이기 때문에 동적 계획법이면서도 BFS/DFS 기반 탐색의 꼴입니다.

- 생각보다 초급반에서 잘 안 풀렸습니다. 역시 동적 계획법은 많은 연습을 필요로 하는 것 같습니다.

풀이

무한부스터

- 두 번째 dp식의 경우, 우선 출발지점이 아닌 칸을 전부 10000 정도로 초기화를 합니다.
- 이후, (i, j)칸에서 획득할 수 있는 부스터의 개수를 k라고 합시다. 그럼
  - `dp[i+x][j] = min(dp[i][j]+1, dp[i+x][j]) (1 <= x <= min(k, n-i))`
  - `dp[i][j+x] = min(dp[i][j]+1, dp[i][j+x]) (1 <= x <= min(k, m-j))`

  의 두 식을 통해, 현재 칸에서 이동해서 갈 수 있는 칸의 정보를 갱신할 수 있습니다.
- 이래도 되는 이유! 한 번 방문한 칸은 다시 방문하기 않기 때문
- 반복문을 돌며 첫 번째 행부터 저렇게 정보를 전파하고, 두 번째 행, …, N번째 행까지 하면 됩니다.

풀이

무한부스터

- 두 번째 dp식의 경우, 우선 출발지점이 아닌 칸을 전부 10000 정도로 초기화를 합니다.
- 이후, (i, j)칸에서 획득할 수 있는 부스터의 개수를 k라고 합시다. 그럼
  - `dp[i][j] = min(dp[i][j], dp[i+x][j]+1) (1 <= x <= min(k, n-i))`
  - `dp[i][j] = min(dp[i][j], dp[i][j+x]+1) (1 <= x <= min(k, m-j))`

  으로 조사할 수 있습니다.
  - (i, j)에서 (n, m)으로 가려면 위에서 조사하는 칸들을 거쳐가는 방법밖에 없기 때문
- 다만 `dp[i+x][j]`가 조사되어있지 않을 수 있습니다. 때문에 재귀적으로 memoization을 통해 저장해야 합니다.

숭고한 초급반 E

우울한 방학

- 출제자 : 공인호 / 분야 : 자유 주제 (초급)
- 제출 19회, 정답 5명
- 처음 풀린 시간 : 86분

풀이

우울한 방학

- 첫 번째 Greedy : 기분이 0 이상인 날에는 약속을 배치할 필요가 없습니다.
- 기대 행복 값이 H_i인 약속은, $H_i + 1$일 동안 불행함을 느끼지 않게 해주기 때문입니다.

풀이

우울한 방학

$H_1 + 1$

$H_2 + 1$

$H_3 + 1$

$\vdots$

$H_n + 1$

1	2	3	4	5	...	m-3	m-2	m-1	m

풀이

우울한 방학

- 아이디어 : 기분이 0 이상인 날에 약속을 배치하지 않으면, 우울함을 느끼게 되는 날짜의 수는 약속 배치와 무관하게 일정합니다.

- n개의 약속이 있으면, n+1개의 우울함을 느끼는 구간이 생깁니다.

- k번째 구간에서 우울함을 느끼게 되는 일 수를 S_k라고 하자. ($1 \leq k \leq n+1$)

- Total_S = $S_1 + S_2 + S_3 + \cdots + S_n + S_{n+1}$ = m - ($(H_1 + 1) + (H_2 + 1) + \cdots + (H_n + 1)$)

풀이

우울한 방학

- answer = min($(1^2+\cdots+S_1^2)+(1^2+\cdots+S_2^2)+\cdots+(1^2+\cdots+S_n^2)+(1^2+\cdots+S_{n+1}^2)$)
- $S_1+S_2+S_3+\cdots+S_n+S_{n+1}$ 의 값은 Total_S로 고정

- 두 번째 Greedy : $S_1, S_2, S_3, \cdots, S_n, S_{n+1}$가 최대한 균등할수록 최적입니다.
- x = Total_S/(n+1)라 하면, S_i 에 x 또는 x+1을 분배하면 됩니다.

- 이유는 $y = x^2$ 의 볼록성 때문으로, 임의의 $a(\geq 1)$, $k(\geq 2)$ 에 대하여 다음 식이 성립합니다.
- $(1^2+\cdots+a^2)+(1^2+\cdots+(a+k)^2) \geq (1^2+\cdots+(a+1)^2)+(1^2+\cdots+(a+k-1)^2)$

숭고한 초급반 F

다이나믹 롤러

- 출제자 : 조민석 / 분야 : 수학, 이분 탐색
- 제출 36회, 정답 8명
- 처음 풀린 시간 : 74분

풀이

다이나믹 롤러

- 각각의 위치에서 칠할 수 있는 최대 범위를 구하자!
- 칠할 수 있는 최대 범위는 잉크지수보다 같거나 작은 최대 위치
- 각 타일의 점도 지수는 이미 정렬되어 있기 때문에, 이분 탐색을 사용해 최대 위치를 구할 수 있다.
- C++의 경우, `std::upper_bound`를 사용할 수도 있습니다.

숭고한 초급반 G

핑거 스냅

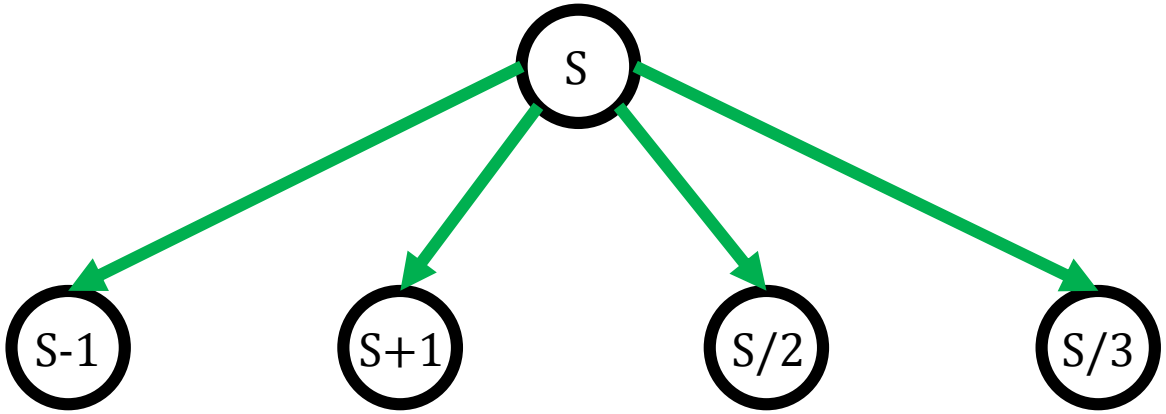
- 출제자 : 박정호 / 분야 : 트리, 그래프, 탐색
- 제출 13회, 정답 1명
- 처음 풀린 시간 : 70분

풀이

핑거 스냅

- 기본적으로 소수인지 판별하기 위해서 에라토스테네스의 체를 사용해서 1부터 100,000까지의 소수를 구해 놓습니다.

- 그리고, 시작점에서부터 BFS를 돌리는데, 여기서 그래프는 다음과 같이 만들어집니다.
- 물론, 방문 여부 체크는 필수입니다.



풀이

핑거 스냅

- BFS를 돌리는 과정에서 구간 [a, b] 안의 소수를 만난다면, 바로 핑거 스냅 횟수를 출력한 후, BFS 순회를 종료하면 됩니다. 이후 초기화도 잘 해주셔야 합니다.
  - 소수 판단은 처음에 구한 에라토스테네스 배열을 이용하면 됩니다.

- BFS의 탐색 범위는 최대 1,000,000까지만 하면 됩니다.
  - 소수의 범위가 100,000까지이며, 수 증가가 '+1' 밖에 없음
  - 증가시켜도 빼거나 나누면 스냅 횟수에 차이가 없음

풀이

핑거 스냅

2로 나누는 것과의 케이스 분석입니다.

1. N = 2a 일 때

   +1 후 /2 -〉 (2a+1)/2 = a (/2한 결과와 같기에 손해)

2. N = 2a+1 일 때

   +1 후 /2 -〉 (2a+2)/2 = a+1 (/2한 후 +1한 것과 같기에 굳이 먼저 할 이유가 없음)

숭고한 중급반 C, 고급반 A

이진수 변환

- 출제자 : 김민성 / 분야 : 자유 주제 (중급)
- 중급반 : 제출 28회, 정답 8명
- 고급반 : 제출 30회, 정답 13명
- 처음 풀린 시간 : 86분(중급반), 15분(고급반)

풀이

이진수 변환

Observation. 제일 큰 n-1자리의 '1' 비트는 반드시 혼자 변해야 한다.

1 0 1 0 0 1 1 1 1 0 1 1 1 (n=6)

각 비트의 변화량 = 4096, 1024, 128, 64, 32, 16, 4, 2, 1
맨 앞자리 하나 바꾸는 게 뒷자리 싹 다 바꾸는 것보다 더 크다.

$$2^0 + 2^1 + 2^2 + \ldots + 2^n < 2^{n+1}$$

n번의 변화 중에서 반드시 어떤 한 변화는 **최상위 비트**(예시에서는 4096에 해당한다)를 고르게 되는데,
그 변화는 **무조건 최대 변화량**이 되어버린다.
더 나아가서, 각각의 변화량들의 서열은 **오직 그 변화량의 최상위 비트의 위치에 의해서 결정**된다.

그렇다면 최대 변화와 최소 변화의 차이를 최소화하기 위해서는, 최상위 비트를 고르는 n-1개의 변화에는 **제일 높은 '1' 들을 각각 단 1개만 할당해야 하고, 나머지 1개의 변화에 남은 '1'들을 모두를 몰빵해야 한다.**

숭고한 **중급반 D, 고급반 B**

백도어

- 출제자 : 이유섭 / 분야 : 최단 거리
- 중급반 : 제출 63회, 정답 6명
- 고급반 : 제출 76회, 정답 16명
- 처음 풀린 시간 : 169분(중급반), 14분(고급반)

풀이

백도어

- 다익스트라 알고리즘을 통해 0번 분기점에서 N-1번째 분기점까지의 최단거리를 구할 수 있으며, 상대의 시야에 걸리는 분기점 및 간선은 사용할 수 없기 때문에 무시해줍니다.
  - 예외는 상대 팀 넥서스가 위치한 N-1번째 분기점으로, 시야에 걸리지 않는 것으로 고려

- 많이 풀 수 있을 줄 알았는데 생각보다 많이 고전했습니다. 어느 정도의 구현 난이도가 있을 뿐더러, 알고리즘을 약간 다르게 이해하여 틀린 경우도 있었고, 64비트 정수형 연산에 있어 많은 실수가 포착되었습니다.

- 다익스트라 알고리즘은 잘못 구현하기 쉽습니다. 올바른 구현체를 숙지하도록 합시다.

숭고한 **중급반 E**

FLEX

- 출제자 : 공인호 / 분야 : DP 중급
- 제출 19회, 정답 2명
- 처음 풀린 시간 : 155분

풀이 FLEX

- 관찰 : 돈을 배분할 때, 하루 지출이 10만원을 넘길 필요가 없습니다.
  - 만약 돈이 남는다면, 마지막 날에 몰아주면 된다.

- DP(n,c,m): n번째 날에 c만원을 지출하고 m만원이 남았을 때, n일까지의 박탈감의 최소 합

`DP(n, c, m) = min(DP(n-1, i, m+c-a[n]) + (i-c)*(i-c), DP(n, c, m)) (i > c)`

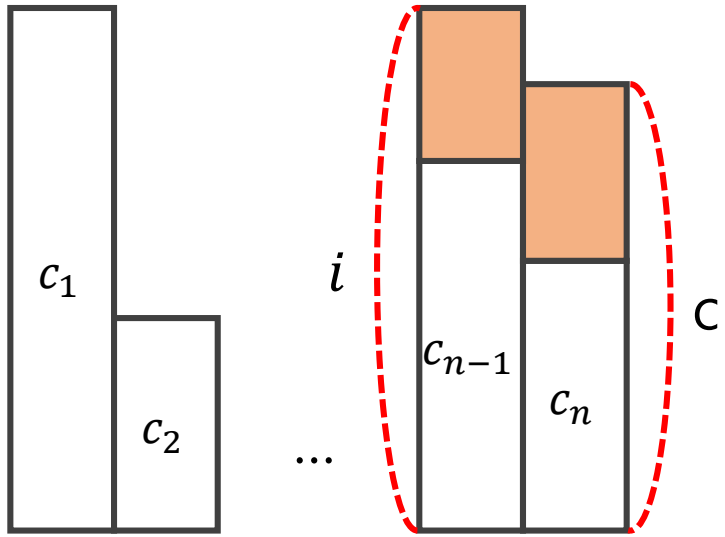
`DP(n, c, m) = min(DP(n-1, i, m+c-a[n]), DP(n, c, m)) (a[n-1] <= i <= c)`

`a[i] : i번째 날의 기본 지출 금액, a[n-1] <= i <= 10, a[n] <= c <= 10`

- 유효한 쌍에 대해서만 테이블 값을 채워놓아야 합니다.

FLEX

- (n, m, c)를 고정하고 가능한 i를 전부 확인해보아 현재 값을 알 수 있습니다.
- (n, m, c)의 상태는 (n−1, x, y)의 값과 관련이 있기 때문에, 일수를 증가시키며 동적 계획법을 진행하면 됩니다. 공간 복잡도는 $O(c_{max}nm)$이며 정해 시간 복잡도는 $O(c_{max}^2 nm)$이나, 시간복잡도 $O(c_{max} nm^2)$도 통과하게 제한을 주었습니다.



숭고한 중급반 F

통신망 분할

- 출제자 : 주영준 / 분야 : 서로소 집합
- 제출 16회, 정답 0명
- 처음 풀린 시간 : -

풀이

통신망 분할

- 주어진 쿼리 그대로 진행하면 O(NQ)이기 때문에 생각을 바꾸어야 합니다.
- 쿼리를 뒤에서부터 보면, 삭제 쿼리가 병합 쿼리로 변경됩니다.
- 그렇게 되면 일반적인 Disjoint-set 문제가 되어, 삭제되지 않는 연결 관계만 남기고 거꾸로 병합 해나가면 됩니다. 각 집합의 크기 관리는 배열 하나만 추가해주면 구할 수 있습니다.
- 정답이 32비트 정수형의 표현 범위를 벗어날 수 있습니다.

- 내부 대회에서는 풀리지 않았지만, Open Contest에서는 FLEX보다 많이 풀렸습니다.

숭고한 **중급반 G, 고급반 E**

트리의 외심

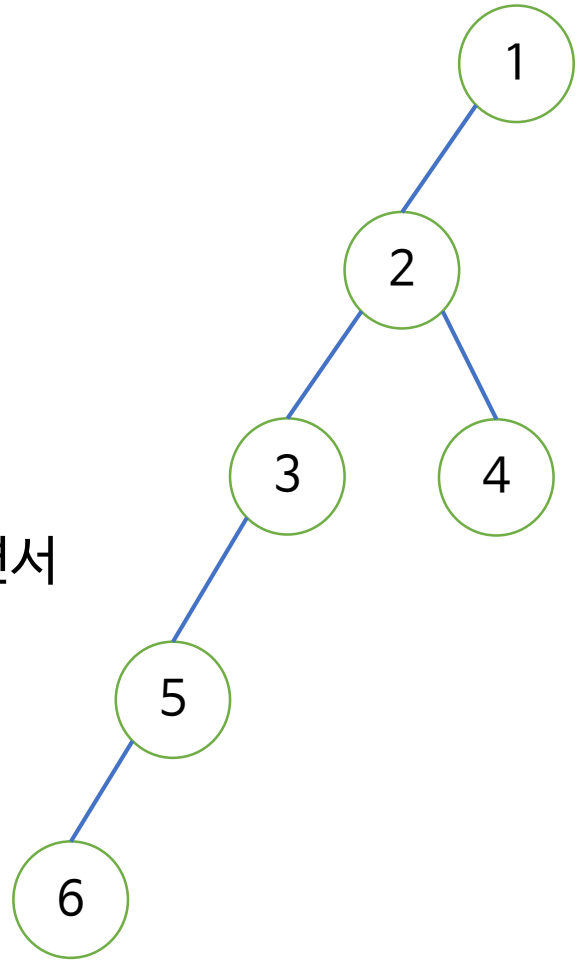
- 출제자 : 이동관 / 분야 : 최저 공통 조상
- 중급반 : 제출 1회, 정답 0명
- 고급반 : 제출 18회, 정답 4명
- 처음 풀린 시간 : -(중급반), 147분(고급반)

풀이

트리의 외심

- LCA를 이용하여 트리의 외심을 구할 수 있습니다.

- 결론부터 말하자면 A,B,C의 외심은
  - A-B의 중점, A-C의 중점, B-C의 중점
- 중 하나이며, 이들이 외심이 아닐 경우 존재하지 않습니다.
  - 여기서 트리의 두 정점의 중점이란, 두 정점으로부터 거리가 같으면서 그 거리가 최소가 되는 정점을 의미합니다.
- 외심이 저 세 조건을 만족하는 건 자명합니다.
  그 역은 어떻게 보일까요?

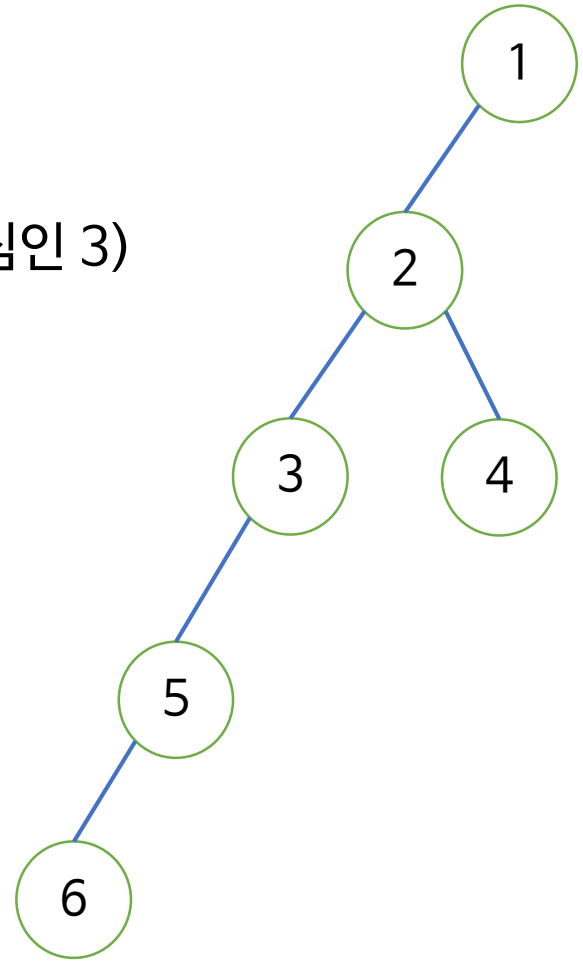


풀이

트리의 외심

- 옆에 있는 트리를 예시로 들면,
  1,4,6의 외심은 (1,4의 중심인 2) or (1,6의 중심인 3) or (4,6의 중심인 3)
- 이 경우 3이 외심이 됩니다.

- A,B의 중점은 “트리와 쿼리 2” 에서 했던 방식과 일치하는 방식으로
  LCA를 구해서 찾을 수 있습니다.



풀이

트리의 외심

- 세 정점이 같다면 그 정점이 답이 되며, 이는 두 정점의 중점 (거리 0)으로도 볼 수 있습니다.
- 세 정점 중 두 정점만 같다면 서로 다른 두 정점의 중점이 외심이 됩니다.
- 세 정점 (A, B, C)가 다른 경우, A-B / B-C / C-A를 연결하는 간선만 고려하도록 하겠습니다.
  그럼 이 서브트리는 일직선이거나, Y자 형태를 띄게 됩니다.
- 일직선일 때는 외심이 존재하지 않으므로, Y자 형태만 고려하면 됩니다.
  이 서브트리를 G라 하겠습니다.

풀이

트리의 외심

- 보조정리. 외심이 존재한다면, G 위에 존재한다.
  - G 위에 있지 않은 정점 X가 외심이라고 가정합시다. 트리이기 때문에, X와 G를 연결하는 유일한 경로가 존재합니다. 이 경로를 따라 X를 G쪽으로 이동시켜도 해당 경로에 있는 모든 정점은 A, B, C와의 거리가 같으며, 거리는 X의 것보다 짧습니다. 이는 가정에 모순이므로 외심이 존재한다면 G 위에 존재합니다.
- G 위에 존재하는 게 뭐 그리 대단해보일 수 있지만, G 위에서 A와 B로부터 거리가 같은 정점은 (존재할 시) 유일합니다. B-C도, C-A도 마찬가지입니다.

풀이

트리의 외심

- 때문에 외심의 후보가 두 점의 중점들인 3개의 정점으로 추려집니다.
- 외심이 존재하면 유일하다는 것도 비슷하게 증명할 수 있기에, 이 세 중점들만 검사하면 됩니다.

- 중급반 최고난도 문제였으나, 고급반에 자유 주제 개념으로 내도 괜찮을 것 같다는 의견이 등장했고, 이어 고급반에서도 최고난도 문제다라는 의견이 우세해졌습니다. 실제로도 내부 고급반 대회에서 가장 적게 (그러나 예상보다는 많이) 풀렸습니다.

숭고한 고급반 C

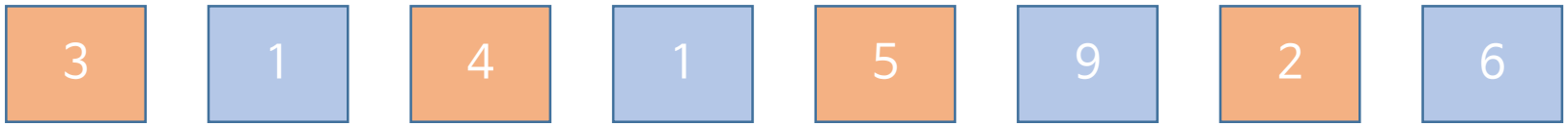
깃발춤

- 출제자 : 이상헌 / 분야 : 세그먼트 트리
- 제출 44회, 정답 18명
- 처음 풀린 시간 : 35분

풀이

깃발춤

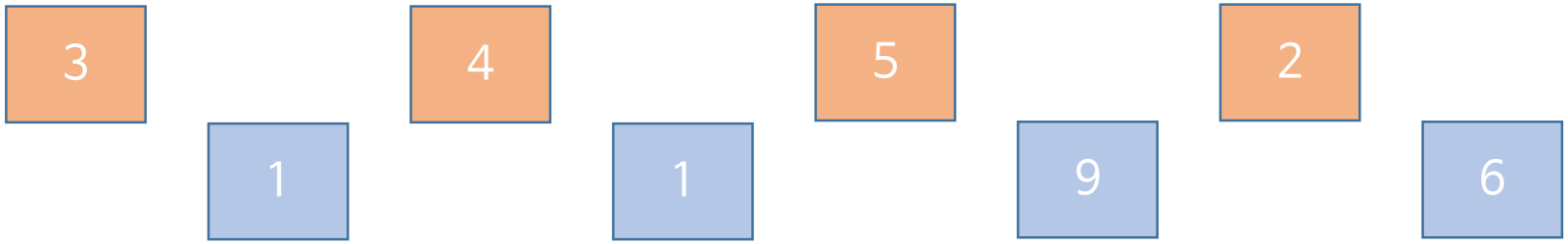
- 구간을 교대하면서 +/-를 하려면 어떻게 해야 할까요?



풀이

깃발춤

- 구간을 교대하면서 +/-를 하려면 어떻게 해야 할까요?



- 홀수 번째 원소끼리와 짝수 번째 원소끼리 묶은 다음, 따로 세그먼트 트리나 펜윅 트리를 만들어 합을 관리하고, 포함되는 구간을 나누어 관리하면 됩니다.
- 약간 다르게 값을 설정하면 세그먼트 트리/펜윅 트리 1개로도 풀 수 있습니다.

- 원래는 lazy를 쓰는 문제였는데, 안 쓰는 버전으로 수정되었습니다. 좋은 선택이었습니다.

숭고한 고급반 D

일하는 세포

- 출제자 : 장형준 / 분야 : DP 고급
- 제출 27회, 정답 11명
- 처음 풀린 시간 : 71분

풀이

일하는 세포

- 그래프를 행렬 T로 표현하면, T의 n제곱의 의미는 다음과 같습니다.
  - i행 j열의 값은 i번 정점에서 시작하여 n개의 간선을 거쳐 j번 정점에 도착하는 가짓수입니다.
- 때문에 이 문제는 행렬곱으로 풀 수 있습니다.
- 답 : $(\text{지도}_1 \times \cdots \times \text{지도}_T)^{D/T} \times (\text{지도}_1 \times \cdots \times \text{지도}_{D\ \%\ T})$
  - 첫 번째 지도는 1초 후를 의미하고, 여기에 두 번째 지도를 의미하는 행렬을 곱하면 첫 번째 그래프를 통해 이동하고, 두 번째 그래프를 통해 이동한 가짓수를 의미하게 됩니다.
  - 그렇게 T번째 지도까지 곱하는 걸 D/T번 반복하고, 1번째 지도부터 D%T번째 지도까지 곱하면 답이 됩니다. (행렬곱은 교환법칙이 성립하지 않습니다!!)
  - 행렬 제곱은 일반적으로 제곱을 빠르게 하는 방법을 이용하여 구할 수 있습니다.

숭고한 **고급반 F**

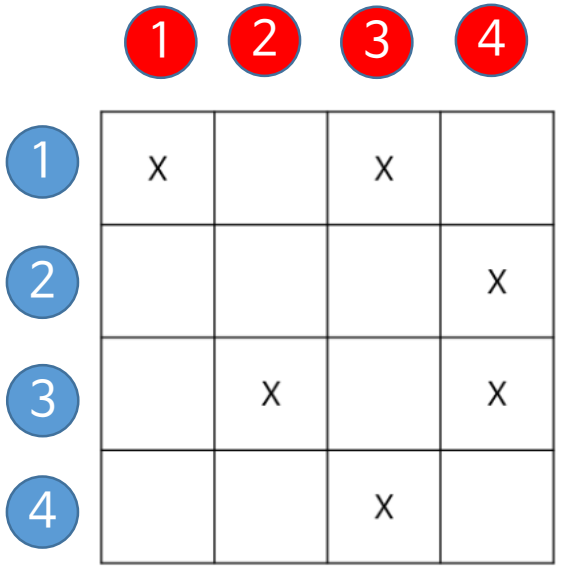
시간 끌기

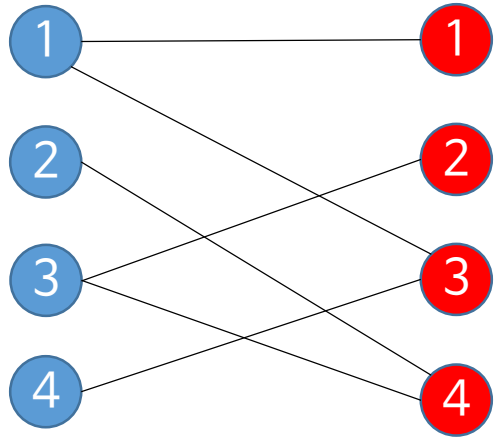
- 출제자 : 박홍빈 / 분야 : 네트워크 플로우
- 제출 20회, 정답 7명
- 처음 풀린 시간 : 54분

풀이

시간 끌기

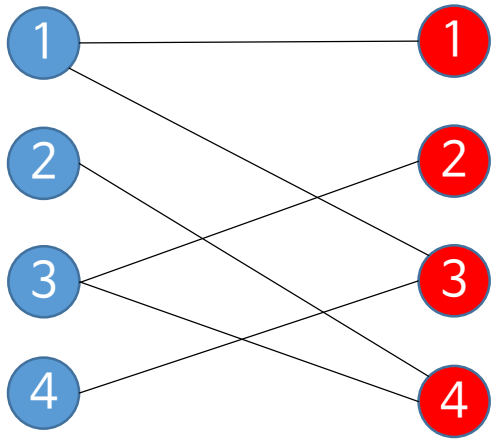
이분그래프를 만드는데 각각 행과 열을 노드로 하고 그 행과 열에 해당하는 칸에 X가 있다면 두 노드를 잇는다.
그럼 아래와 같이 바꿀 수 있다.





풀이

시간 끌기



이제 문제는 오른쪽의 그래프에서 아래 조건을 만족하는 최대 정점을 고르는 문제로 바꿀 수 있다.

- 고른 정점들 중 어느 두 개를 골라도 그 둘을 잇는 간선이 없다.

위의 조건을 바꾸어 생각하면 어떤 간선을 선택해도 양 끝 정점 중에서 적어도 한 정점은 고르지 않아야 한다.

즉 고르지 않는 정점들이 vertex cover이며 고른 정점들의 수가 최대가 되어야 하므로 고르지 않은 정점들이 minimum vertex cover가 된다.

따라서 답은 (전체 정점 개수) - (minimum vertex cover)이며 이분 그래프이므로 minimum vertex cover는 이분 매칭이나 flow를 이용하여 구할 수 있다.

풀이

시간 끌기

- 이분 매칭의 유명한 유형으로, Kőnig's Theorem과 independent set, vertex cover에 대한 개념 숙지와 이분 매칭 구현이 필요한 문제였습니다.

- 내부 대회에서는 템플릿 코드에 위의 개념 및 정리 설명을 매우 상세히 해주었으나, 프리패스가 되는 걸 원하지 않아 이분 매칭 코드는 생략하였습니다.

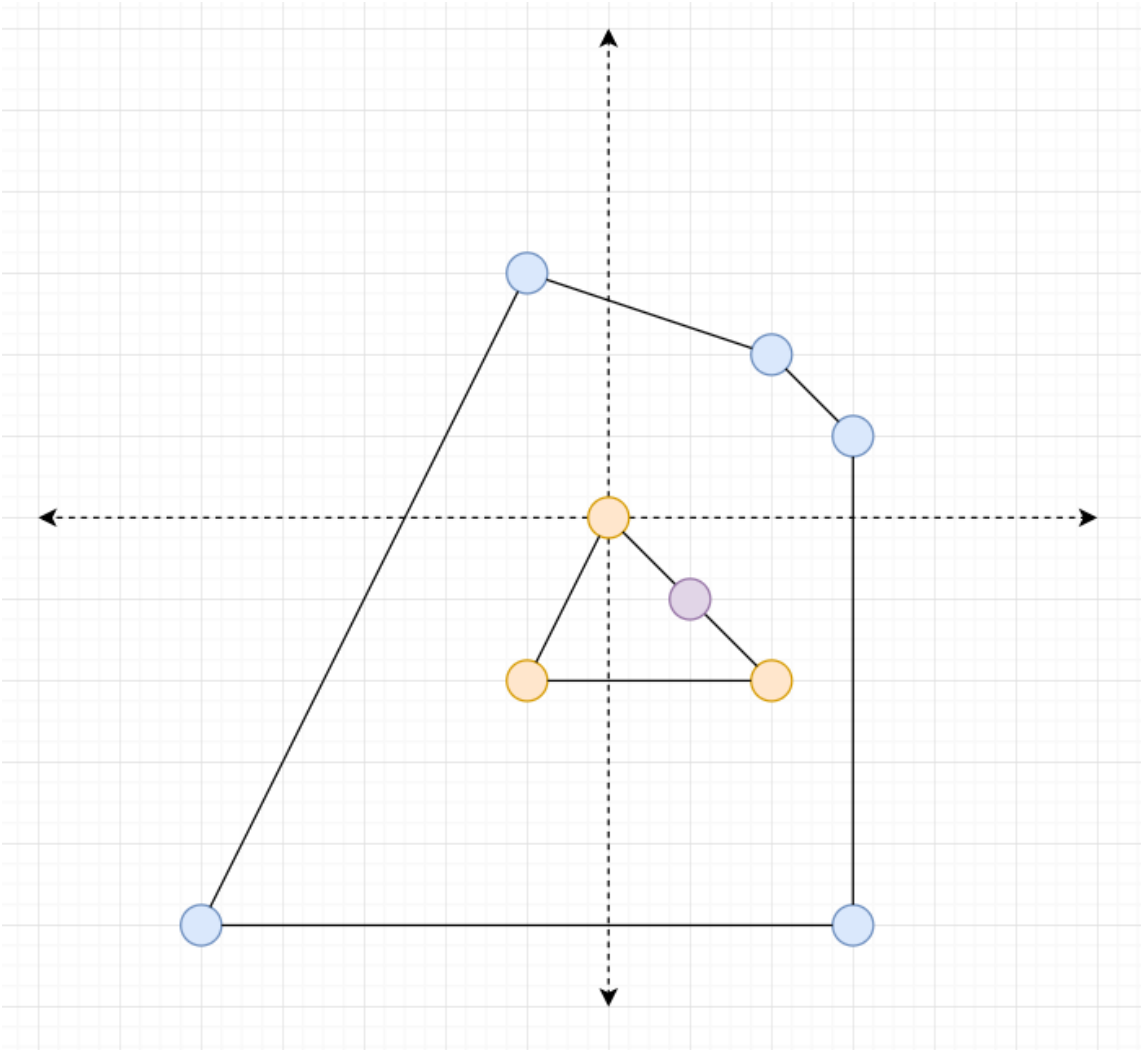
숭고한 고급반 G

가장 높고 넓은 성

- 출제자 : 김민성 / 분야 : 기하
- 제출 41회, 정답 15명
- 처음 풀린 시간 : 17분

풀이

가장 높고 넓은 성 – 로직



Convex Hull을 만들고 남은 점들 가지고 다시 Convex Hull을 만들고 다시 남은 점들 가지고 Convex Hull을 만들고...

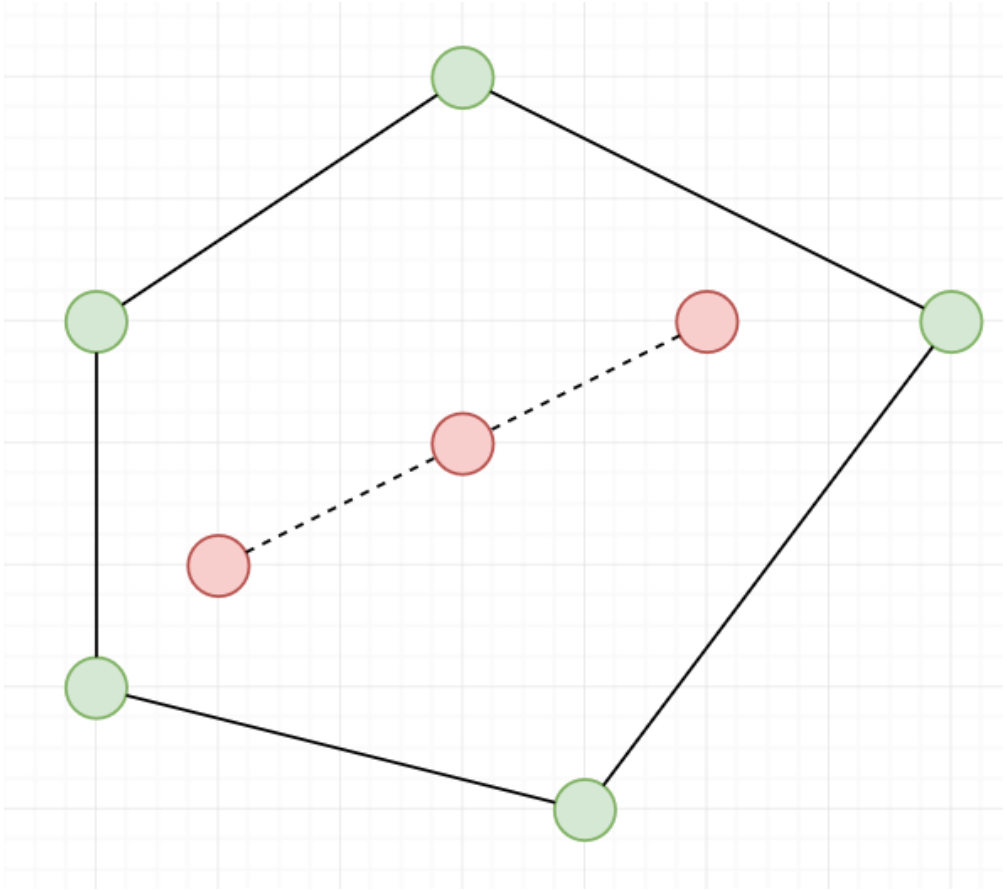
더 이상 못 만들 때까지 무한반복하면 된다. 그래야만 가장 높고 넓은 성을 동시에 만족시킬 수 있다.

단, 구현 방식에 따라 조심해야 할 수도 있는 케이스가 존재한다.

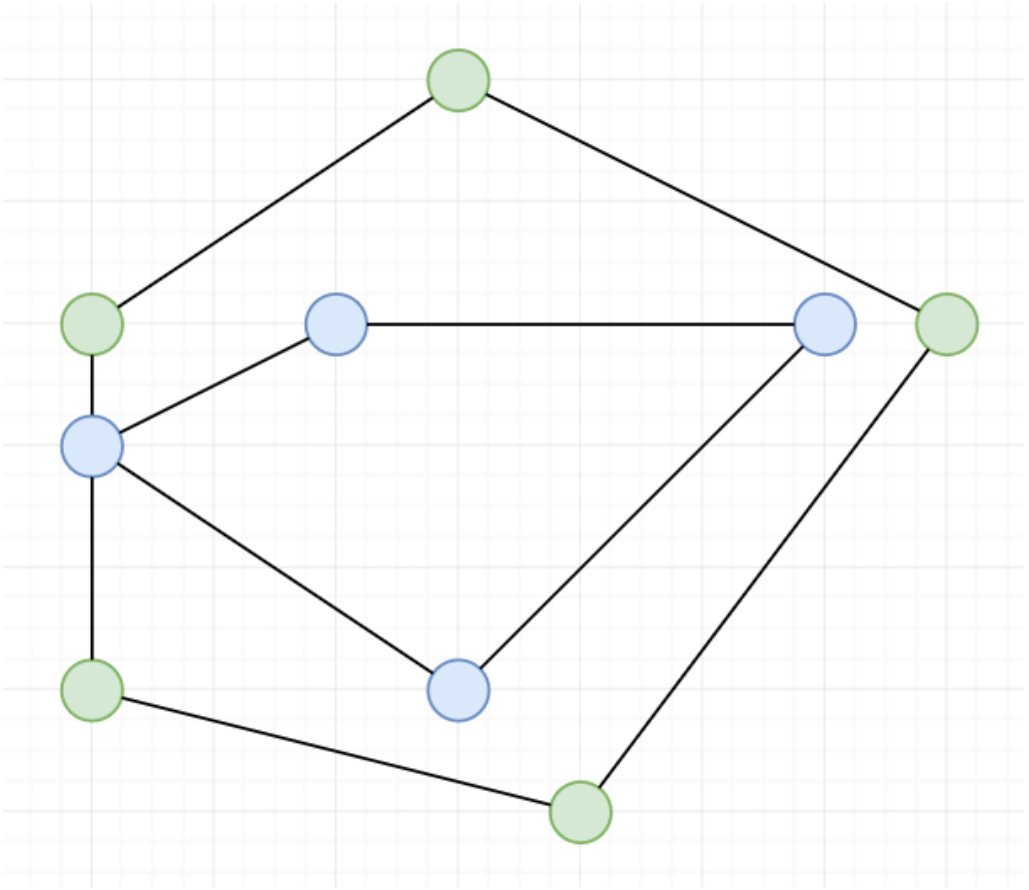
시간복잡도는 O(n^2) ~ O(n^2 log n)

풀이

가장 높고 넓은 성 – 예외



남은 점들이 3개 이상임에도 불구하고
컨벡스헐을 못 만드는 경우
(이 예시는 1층이 전부다)



외부 레이어에 걸친 점이
내부 레이어로 들어오는 경우