2018 서울대학교 프로그래밍 경시대회

September 11, 2018

수고하셨습니다

- 총 참가자: 54명 (Div 2: 29명, Div 1: 25명)
- 총 제출 횟수: 971 (Div 2: 583, Div 1: 388)
- 총 정답 횟수: 268 (Div 2: 115, Div 1: 153)

Div2A. 5차 전직

- 제출 횟수 : 94
- 맞은 참가자 수 : 22
- 정답률 : 23.404%
- 처음 맞은 참가자 : 정유석 (0:05)
- 출제자 : kipa00

Div2A. 5차 전직

- a_i가 주어지고, $b_i = \min(i-1, k)$일 때,

$$\max_{\sigma \in S_n} \sum_{i=1}^{n} a_i b_{\sigma(i)}$$

  의 값을 구하는 문제
  - S_n은 모든 permutation의 집합
- 일반성을 잃지 않고 $a_1 \le a_2 \le \cdots \le a_n$이라 합니다
- $|S_n| = n!$이기에 이것을 전부 계산하면 $O(n \cdot n!)$로 시간 초과!

Div2A. 5차 전직

- 정답 배치의 permutation을 생각: σ^* 이라고 합시다
- $j > k$인 임의의 j와 k에 대해 permutation σ_{jk}를 다음과 같이 정의:

$$\sigma_{jk}(i) := \begin{cases} \sigma^*(j) & i = k \\ \sigma^*(k) & i = j \\ \sigma^*(i) & \text{otherwise} \end{cases}$$

- σ^* 이 최적 배치이기 때문에

$$\sum_{i=1}^{n} a_i b_{\sigma^*(i)} \geq \sum_{i=1}^{n} a_i b_{\sigma_{jk}(i)}$$

$$a_j b_{\sigma^*(j)} + a_k b_{\sigma^*(k)} \geq a_j b_{\sigma_{jk}(j)} + a_k b_{\sigma_{jk}(k)}$$

$$(a_j - a_k)\left(b_{\sigma^*(j)} - b_{\sigma^*(k)}\right) \geq 0$$

Div2A. 5차 전직

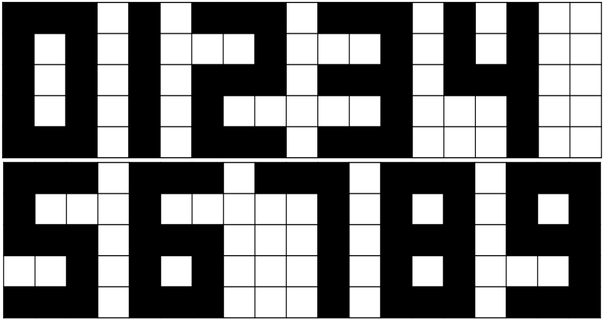
$$\left(a_j - a_k\right)\left(b_{\sigma^*(j)} - b_{\sigma^*(k)}\right) \geq 0$$

- $j > k$이므로 $a_j \geq a_k$이고, $a_j > a_k$일 때 $b_{\sigma^*(j)} \geq b_{\sigma^*(k)}$
- 즉 $b_{\sigma^*(1)} \leq b_{\sigma^*(2)} \leq \cdots \leq b_{\sigma^*(n)}$이면 되고, 이런 σ^*는 $\sigma^*(i) = i$
- 따라서, 정렬한 후 각각의 a_i에 $b_i = \min(i-1, k)$를 곱해서 더한 값
  - $9 \times 10^{18} < 2^{63}$이라서, C++의 `long long int` 범위 안에 들어갑니다
- 시간 복잡도: 정렬에 $O(n \log n)$, 값 계산에 $O(n)$

Div2B. 시그널

- 제출 횟수 : 85
- 맞은 참가자 수 : 23
- 정답률 : 27.059%
- 처음 맞은 참가자 : 김무환 (0:25)
- 출제자 : 16silver

Div2B. 시그널



- 한 열씩 읽어 가면서 0부터 9까지의 숫자를 구분하면 됩니다.
- 1 양 옆의 열이 비어 있음에 주의해야 합니다.
- 1이 맨 앞 / 맨 뒤에 있는 경우를 처리할 수 있어야 합니다.
- 시그널에는 숫자가 최대 10000개까지 있을 수 있습니다.

Div2C. 화살표 연산자

- 제출 횟수 : 129
- 맞은 참가자 수 : 25
- 정답률 : 19.380%
- 처음 맞은 참가자 : 이재찬 (0:12)
- 출제자 : doju

Div2C. 화살표 연산자

- 프로그램을 직접 실행해 보면 많은 도움이 됩니다.
- 감소 연산자가 사용된 경우 ($N \geq 2$)
  - N이 홀수일 경우 컴파일에 실패합니다.
  - N이 짝수일 경우 쉽게 답을 계산할 수 있습니다.

Div2C. 화살표 연산자

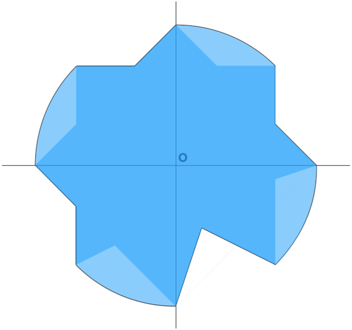
- $N = 0$인 경우: `while(x > 0)`
  - X가 0보다 크다면 무한 루프를 돌고, 그렇지 않다면 즉시 종료됩니다.
- $N = 1$인 경우: `while(-x > 0)`
  - X가 0보다 작다면 무한 루프를 돌고, 그렇지 않다면 즉시 종료됩니다.

Div2D. 팬이에요

- 제출 횟수: 108
- 맞은 참가자 수: 12
- 정답률: 11.111%
- 처음 맞은 참가자: 이정민 (0:41)
- 출제자: cki86201

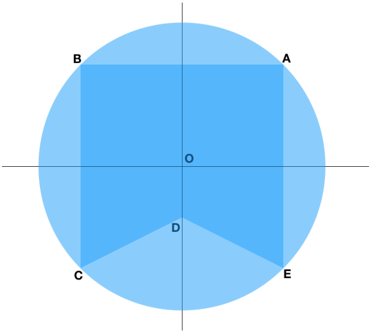
Div2D. 팬이에요

- 원점을 기준으로 돌아가는 도형의 자취가 언제 원이 되는지 판별하는 문제입니다.



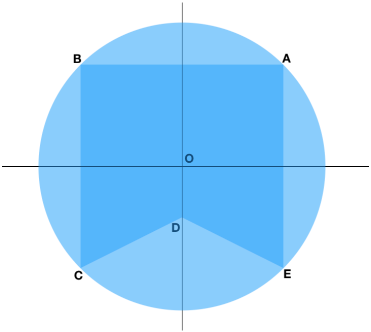
Div2D. 팬이에요

- 자취로 만들어지는 원의 반지름은 얼마일까요?



Div2D. 팬이에요

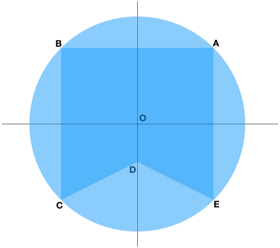
- 자취로 만들어지는 원의 반지름은 얼마일까요?



- 원점으로부터 가장 먼 꼭지점까지 거리입니다.

Div2D. 팬이에요

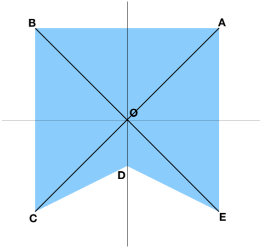
- 완성되는 원의 둘레에 주목해 봅시다.



- 원이 되려면 A가 B 위치까지 가야 하고, B는 C 위치까지, C는 E 위치까지, E는 A 위치까지 가야 합니다.

Div2D. 팬이에요

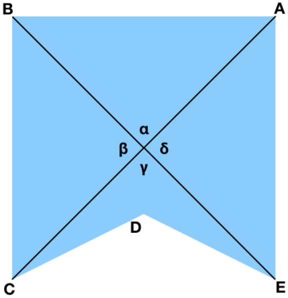
- 문제의 조건에 의해, 원점과 꼭짓점을 잇는 선분 위의 모든 점은 다각형의 내부 혹은 경계에 있습니다.



- 위 조건이 만족될 경우 네 선분 OA, OB, OC, OE의 자취만으로 원이 만들어집니다.

Div2D. 팬이에요

- 즉 아래 4개 중심각 중 가장 큰 각이 답입니다.



Div2D. 팬이에요

- 원점에서 가장 먼 꼭짓점들을 전부 구합니다.

Div2D. 팬이에요

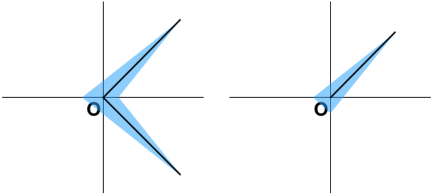
- 원점에서 가장 먼 꼭짓점들을 전부 구합니다.
- 이들은 입력에서 이미 반시계 방향으로 정렬되어 있습니다.

Div2D. 팬이에요

- 원점에서 가장 먼 꼭짓점들을 전부 구합니다.
- 이들은 입력에서 이미 반시계 방향으로 정렬되어 있습니다.
- 이들 안에서 인접한 두 꼭짓점의 중심각 중 가장 큰 각도를 구합니다.

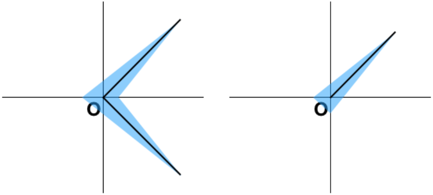
Div2D. 팬이에요

- 답이 정확히 180인 경우, 180보다 큰 경우, 360인 경우에 주의합시다.



Div2D. 팬이에요

- 답이 정확히 180인 경우, 180보다 큰 경우, 360인 경우에 주의합시다.



- 각도를 구하는 방법 중 몇몇은 실수 오차에 주의해야 합니다.
- Ex. `acos(-1.00...02) = nan`

Div2E. 작은 큐브러버

- 제출 횟수 : 13
- 맞은 참가자 수 : 6
- 정답률 : 46.154%
- 처음 맞은 참가자 : 이정민 (1:05)
- 출제자 : doju

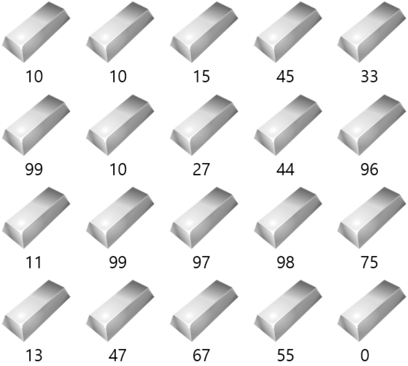
Div2E. 작은 큐브러버

- 큐브의 각 면의 색을 정해 놓고, 분해했을 때 나오는 조각들이 입력으로 주어진 것과 같은지 판별하면 됩니다.
  - $6! \times 8 \times 8 = 46,080$
- 그 외에도 다양한 방법으로 풀 수 있습니다.
- 문제에서 입체 도형이 주어질 땐 직접 종이를 잘라 만들어 보면 좋습니다.

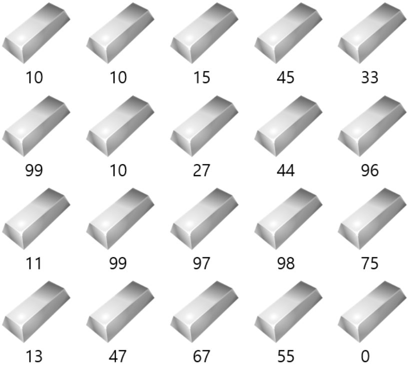
Div2F. 실버런

- 제출 횟수 : 42
- 맞은 참가자 수 : 8
- 정답률 : 19.048%
- 처음 맞은 참가자 : 정유석 (1:19)
- 출제자 : 16silver

Div2F. 실버런

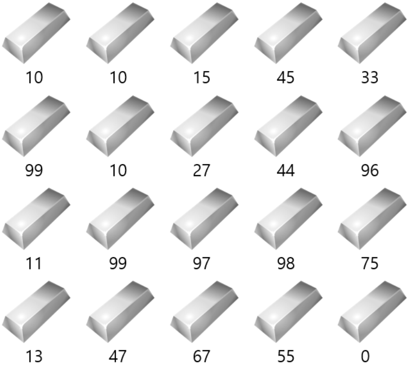


Div2F. 실버런



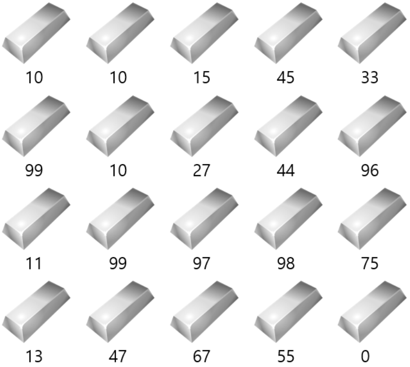
- 실버**런 (Run)**이지만 캐릭터가 달리지는 않습니다.

Div2F. 실버런



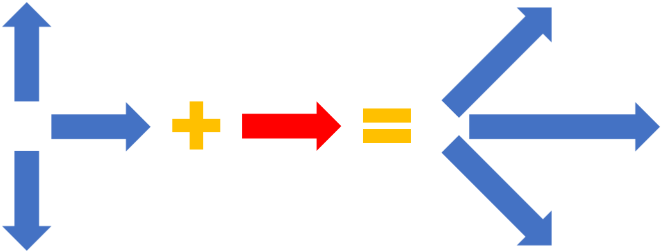
- 실버**런 (Run)**이지만 캐릭터가 달리지는 않습니다.
- 실버와 캐릭터가 모두 움직여서 복잡합니다.

Div2F. 실버런

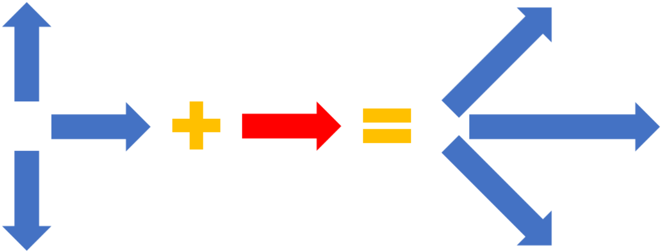


- 실버**런 (Run)**이지만 캐릭터가 달리지는 않습니다.
- 실버와 캐릭터가 모두 움직여서 복잡합니다.
- 그러면 실버에 대한 캐릭터의 상대 속도를 따져 봅시다!

Div2F. 실버런

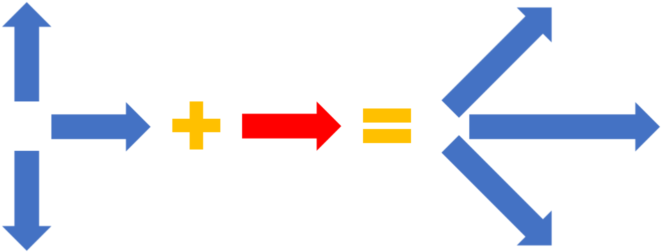


Div2F. 실버런



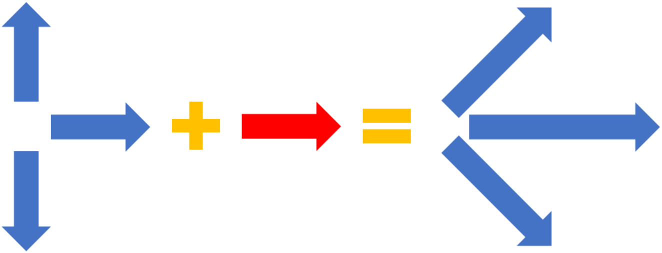
- 맵이 고정되어 있고 캐릭터가 세 방향 중 하나로 가는 문제로 변환!**(주의 : 한 칸 오른쪽으로 갈 수 없습니다!)**

Div2F. 실버런



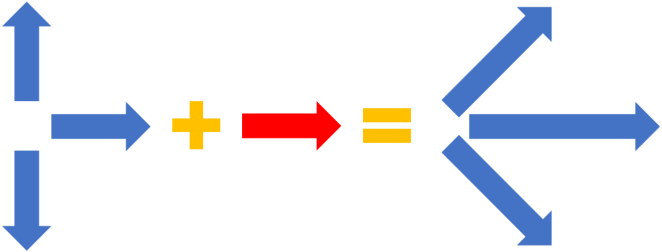
- 맵이 고정되어 있고 캐릭터가 세 방향 중 하나로 가는 문제로 변환!**(주의 : 한 칸 오른쪽으로 갈 수 없습니다!)**
- **다이나믹 프로그래밍 (DP)**으로 이 문제를 해결할 수 있습니다.

Div2F. 실버런



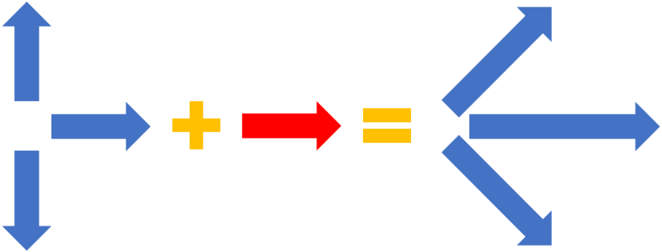
- $(i, j) \to (i-1, j+1)$: $(i-1, j+1)$에 있는 실버를 얻습니다.

Div2F. 실버런



- $(i, j) \to (i-1, j+1)$: $(i-1, j+1)$에 있는 실버를 얻습니다.
- $(i, j) \to (i+1, j+1)$: $(i+1, j+1)$에 있는 실버를 얻습니다.

Div2F. 실버런



- $(i, j) \to (i-1, j+1)$: $(i-1, j+1)$에 있는 실버를 얻습니다.
- $(i, j) \to (i+1, j+1)$: $(i+1, j+1)$에 있는 실버를 얻습니다.
- $(i, j) \to (i, j+2)$: $(i, j+1), (i, j+2)$에 있는 실버를 얻습니다.

Div2F. 실버런

- $\mathrm{dp}(i, j)$ 를, (i, j) 에 도착하는 경로들에서 얻는 실버의 수량 중 최댓값으로 정의합니다.

Div2F. 실버런

- $\mathrm{dp}(i,j)$ 를, (i,j) 에 도착하는 경로들에서 얻는 실버의 수량 중 최댓값으로 정의합니다.
- $\mathrm{dp}(i,j) = \max\{\mathrm{dp}(i-1,j-1) + a_{ij}, \mathrm{dp}(i+1,j-1) + a_{ij}, \mathrm{dp}(i,j-2) + a_{i(j-1)} + a_{ij}\}$

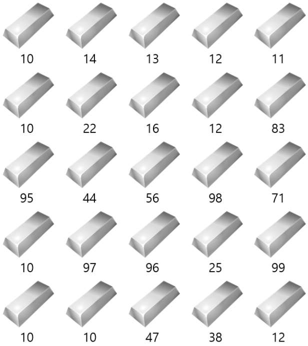
Div2F. 실버런

- $\mathrm{dp}(i,j)$ 를, (i,j) 에 도착하는 경로들에서 얻는 실버의 수량 중 최댓값으로 정의합니다.
- $\mathrm{dp}(i,j) = \max\{\mathrm{dp}(i-1,j-1) + a_{ij}, \mathrm{dp}(i+1,j-1) + a_{ij}, \mathrm{dp}(i,j-2) + a_{i(j-1)} + a_{ij}\}$
- 맵의 어느 위치에서든 시작할 수 있으므로, 초기값은 맵 왼쪽 한 칸에서부터 시작합니다.
- 마지막에 오른쪽으로 가서 맵 오른쪽 한 칸에서 끝나는 경우를 생각해주어야 합니다.
- 답은 $\max\{\mathrm{dp}(i, M+1)\}$ 입니다.

Div2F. 실버런

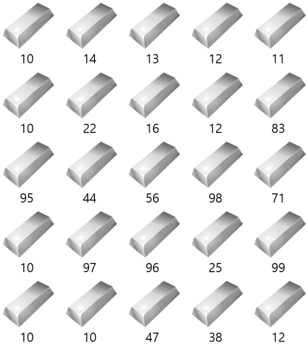
- $\mathrm{dp}(i,j)$ 를, (i,j) 에 도착하는 경로들에서 얻는 실버의 수량 중 최댓값으로 정의합니다.
- $\mathrm{dp}(i,j) = \max\{\mathrm{dp}(i-1,j-1) + a_{ij}, \mathrm{dp}(i+1,j-1) + a_{ij}, \mathrm{dp}(i,j-2) + a_{i(j-1)} + a_{ij}\}$
- 맵의 어느 위치에서든 시작할 수 있으므로, 초기값은 맵 왼쪽 한 칸에서부터 시작합니다.
- 마지막에 오른쪽으로 가서 맵 오른쪽 한 칸에서 끝나는 경우를 생각해주어야 합니다.
- 답은 $\max\{\mathrm{dp}(i, M+1)\}$ 입니다.
- 이 모든 것을 고려하여 코드를 제출하면... 틀렸습니다

Div2F. 실버런



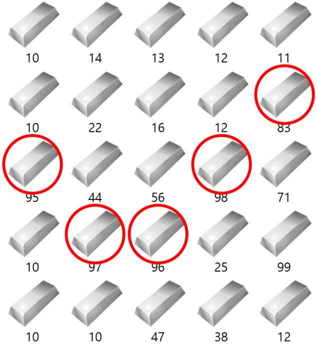
- **예제를 꼭 돌려 봅시다!**
- 위의 풀이에 의하면 답은 432입니다.(95+44+96+98+99)
- 하지만 예제 출력에 나와 있는 답은 469입니다!

Div2F. 실버런



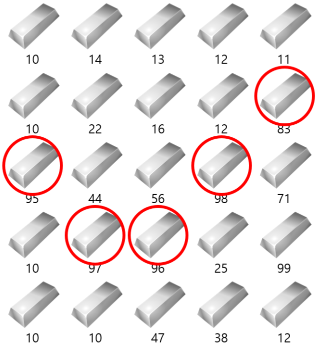
- **예제를 꼭 돌려 봅시다!**
- 위의 풀이에 의하면 답은 432입니다.(95+44+96+98+99)
- 하지만 예제 출력에 나와 있는 답은 469입니다!
- 어떻게 469가 가능한 걸까요?

Div2F. 실버런



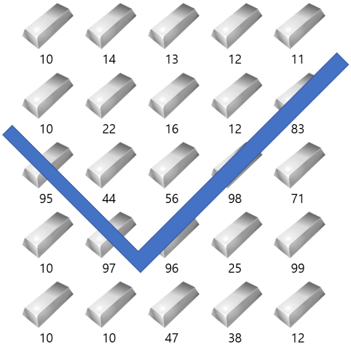
- 관찰을 통해 469가 95+97+96+98+83임을 확인할 수 있습니다.
- 하지만 이 경로는 97에서 96으로 갈 때 멈추므로 불가능한 경로입니다.

Div2F. 실버런

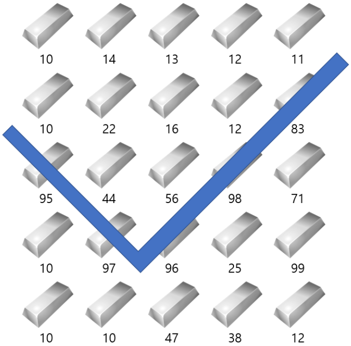


- 관찰을 통해 469가 95+97+96+98+83임을 확인할 수 있습니다.
- 하지만 이 경로는 97에서 96으로 갈 때 멈추므로 불가능한 경로입니다.
- ????????

Div2F. 실버런

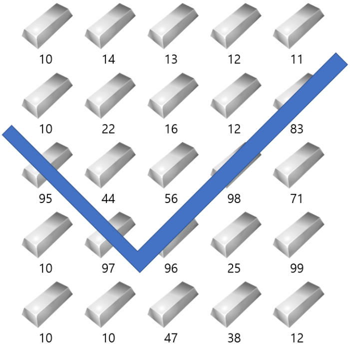

10 14 13 12 11
10 22 16 12 83
95 44 56 98 71
10 97 96 25 99
10 10 47 38 12

Div2F. 실버런



- **정수 좌표가 아니어도 된다!**는 문제 조건이 있습니다.
- 따라서 위 그림과 같은 경로가 가능합니다.

Div2F. 실버런



- **정수 좌표가 아니어도 된다!**는 문제 조건이 있습니다.
- 따라서 위 그림과 같은 경로가 가능합니다.
- 사악한 16silver는 이 경로를 기어코 허용하고야 말았지만, 예제에서 이 경로를 유추할 수 있도록 하였습니다.

Div2F. 실버런

- 최적해는 (정수, 정수) or (정수+0.5, 정수+0.5) 에서 시작하는 경로임이 보장됩니다.
- 실버를 지나가기 위해서는 (정수,r) 이거나 , (정수, 정수) 로부터 대각선으로 갈 수 있는 점이어야 합니다.
- 만약 (정수,r) 인 경우 시작점이 (정수, 정수) 가 되도록 오른쪽으로 평행이동하면 이 경로가 지나는 점을 모두 지납니다.
- 나머지 경우, 시작점이 (정수+0.5, 정수+0.5) 가 되도록 대각선 방향으로 평행이동하면 이 경로가 지나는 점을 모두 지납니다.
- 실버의 수량은 0 이상이므로 더 많은 점을 지난다면 더 좋은 해입니다.
- 그러므로 이 두 경우만 생각해주면 됩니다.

Div2F. 실버런

- 방법 1: 새로운 DP table
- $\mathrm{dp2}(i, j, \mathrm{up}) = \max\{\mathrm{dp2}(i+1, j-1, \mathrm{up}), \mathrm{dp2}(i, j-1, \mathrm{down})\} + a_{ij}$
- $\mathrm{dp2}(i, j, \mathrm{down}) = \max\{\mathrm{dp2}(i-1, j-1, \mathrm{down}), \mathrm{dp2}(i, j-1, \mathrm{up})\} + a_{ij}$

Div2F. 실버런

- 방법 1: 새로운 DP table
- $\mathrm{dp2}(i, j, \mathrm{up}) = \max\{\mathrm{dp2}(i+1, j-1, \mathrm{up}), \mathrm{dp2}(i, j-1, \mathrm{down})\} + a_{ij}$
- $\mathrm{dp2}(i, j, \mathrm{down}) = \max\{\mathrm{dp2}(i-1, j-1, \mathrm{down}), \mathrm{dp2}(i, j-1, \mathrm{up})\} + a_{ij}$
- 방법 2: 전체 scale을 2배로 늘리기
- 출발점이 정수점이 되도록 scale을 2배로 늘리면 이전에 정의한 dp 점화식만으로 문제를 해결할 수 있습니다.
- 주의 : 이동 길이 역시 2배로 늘려줘야 합니다!

Div1C/Div2G. PPAP

- 제출 횟수 : 74
- 맞은 참가자 수 : 14
- 정답률 : 20.270%
- 처음 맞은 참가자 : 이정민 (0:21)
- 출제자 : 16silver

Div1C/Div2G. PPAP

- 주어진 문자열이 PPAP 문자열인지 아닌지 판단하는 문제

Div1C/Div2G. PPAP

- 주어진 문자열이 PPAP 문자열인지 아닌지 판단하는 문제
- 관찰을 통해 PPAP 문자열의 규칙을 찾아내는 것이 중요!

Div1C/Div2G. PPAP

- 주어진 문자열이 PPAP 문자열인지 아닌지 판단하는 문제
- 관찰을 통해 PPAP 문자열의 규칙을 찾아내는 것이 중요!
- **P를 PPAP로 바꾸는 동안 유지되는 성질들 (필요조건)**

Div1C/Div2G. PPAP

- 주어진 문자열이 PPAP 문자열인지 아닌지 판단하는 문제
- 관찰을 통해 PPAP 문자열의 규칙을 찾아내는 것이 중요!
- **P를 PPAP로 바꾸는 동안 유지되는 성질들 (필요조건)**
  1. 문자열의 시작과 끝은 P이다.
  2. A는 연속해서 나오지 않는다.
  3. P의 개수를 n_P, A의 개수를 n_A라 할 때, $n_P = 2 \times n_A + 1$

Div1C/Div2G. PPAP

- 주어진 문자열이 PPAP 문자열인지 아닌지 판단하는 문제
- 관찰을 통해 PPAP 문자열의 규칙을 찾아내는 것이 중요!
- **P를 PPAP로 바꾸는 동안 유지되는 성질들 (필요조건)**
  1. 문자열의 시작과 끝은 P이다.
  2. A는 연속해서 나오지 않는다.
  3. P의 개수를 n_P, A의 개수를 n_A라 할 때, $n_P = 2 \times n_A + 1$
- 하지만 아직 충분조건은 아닙니다!

Div1C/Div2G. PPAP

- 생각 : PPAP 문자열에서 역으로 “PPAP” 를 “P” 로 바꾸면 “P” 로 만들 수 있다.

Div1C/Div2G. PPAP

- 생각 : PPAP 문자열에서 역으로 “PPAP” 를 “P” 로 바꾸면 “P” 로 만들 수 있다.
- 질문 : 어떤 “PPAP” 를 바꿀 것인가?

Div1C/Div2G. PPAP

- 생각: PPAP 문자열에서 역으로 “PPAP”를 “P”로 바꾸면 “P”로 만들 수 있다.
- 질문: 어떤 “PPAP”를 바꿀 것인가?
- 추측: 아무거나? 그래도 될까요?

Div1C/Div2G. PPAP

- 생각 : PPAP 문자열에서 역으로 “PPAP” 를 “P” 로 바꾸면 “P” 로 만들 수 있다.
- 질문 : 어떤 “PPAP” 를 바꿀 것인가?
- 추측 : 아무거나? 그래도 될까요?
- 결론 : 됩니다!

Div1C/Div2G. PPAP

- 생각 : PPAP 문자열에서 역으로 “PPAP”를 “P”로 바꾸면 “P”로 만들 수 있다.
- 질문 : 어떤 “PPAP”를 바꿀 것인가?
- 추측 : 아무거나? 그래도 될까요?
- 결론 : 됩니다!
- 이제 증명을 해 보도록 합시다.

Div1C/Div2G. PPAP

- 생각 : PPAP 문자열에서 역으로 "PPAP"를 "P"로 바꾸면 "P"로 만들 수 있다.
- 질문 : 어떤 "PPAP"를 바꿀 것인가?
- 추측 : 아무거나? 그래도 될까요?
- 결론 : 됩니다!
- 이제 증명을 해 보도록 합시다.
- 증명에 앞서, 올바른 괄호 문자열에 대해 살펴봅시다.

Div1C/Div2G. PPAP

- ‘(’ 와 ‘)’ 로만 이루어진 문자열이 올바른 괄호 문자열이라는 것은 다음과 같다.

Div1C/Div2G. PPAP

- '(' 와 ')' 로만 이루어진 문자열이 올바른 괄호 문자열이라는 것은 다음과 같다.
  1. 빈 문자열은 올바른 괄호 문자열이다.
  2. X가 올바른 괄호 문자열이면 (X) 도 올바른 괄호 문자열이다.
  3. X, Y가 올바른 괄호 문자열이면 XY도 올바른 괄호 문자열이다.

Div1C/Div2G. PPAP

- '(' 와 ')' 로만 이루어진 문자열이 올바른 괄호 문자열이라는 것은 다음과 같다.
  1. 빈 문자열은 올바른 괄호 문자열이다.
  2. X가 올바른 괄호 문자열이면 (X) 도 올바른 괄호 문자열이다.
  3. X, Y가 올바른 괄호 문자열이면 XY도 올바른 괄호 문자열이다.
- 올바른 괄호 문자열일 필요충분조건: 빈 문자열에서 시작, 문자열의 임의의 위치에 "()" 를 넣는 것을 반복하여 만들 수 있다.

Div1C/Div2G. PPAP

- '(' 와 ')' 로만 이루어진 문자열이 올바른 괄호 문자열이라는 것은 다음과 같다.
  1. 빈 문자열은 올바른 괄호 문자열이다.
  2. X가 올바른 괄호 문자열이면 (X)도 올바른 괄호 문자열이다.
  3. X, Y가 올바른 괄호 문자열이면 XY도 올바른 괄호 문자열이다.
- 올바른 괄호 문자열일 필요충분조건: 빈 문자열에서 시작, 문자열의 임의의 위치에 "()"를 넣는 것을 반복하여 만들 수 있다.
- 증명: $(\Rightarrow)$ 문자열 길이에 대해 수학적 귀납법
- $(\Leftarrow)$ "()"를 넣은 횟수에 대해 수학적 귀납법

Div1C/Div2G. PPAP

- 또 하나의 필요충분조건: x=0에서 시작, 문자열의 문자들을 하나씩 보면서 '(' 이면 x++, ')' 이면 x--할 때, x가 항상 0 이상이고 마지막에 x가 0이다.

Div1C/Div2G. PPAP

- 또 하나의 필요충분조건: x=0에서 시작, 문자열의 문자들을 하나씩 보면서 '(' 이면 x++, ')' 이면 x--할 때, x가 항상 0 이상이고 마지막에 x가 0이다.
- 증명: $(\Rightarrow)$ "()"를 넣은 횟수에 대해 수학적 귀납법
- $(\Leftarrow)$ 문자열 길이에 대해 수학적 귀납법, 처음 이후로 최초로 x가 0이 되는 지점에 주목

Div1C/Div2G. PPAP

- 또 하나의 필요충분조건: x=0에서 시작, 문자열의 문자들을 하나씩 보면서 '(' 이면 x++, ')' 이면 x--할 때, x가 항상 0 이상이고 마지막에 x가 0이다.
- 증명: ($\Rightarrow$) "()"를 넣은 횟수에 대해 수학적 귀납법
- ($\Leftarrow$) 문자열 길이에 대해 수학적 귀납법, 처음 이후로 최초로 x가 0이 되는 지점에 주목
- 따름정리: 올바른 괄호 문자열에서 "()"를 빼도 올바른 괄호 문자열이다.

Div1C/Div2G. PPAP

Claim

'P' 와 'A' 만으로 이루어진 문자열 S에 대해 다음 세 가지 명제는 동치이다.

1. S가 PPAP 문자열이다.
2. S가 조건 1, 2를 만족하며, S에 있는 모든 "AP" 를 ')' 로 바꾼 뒤, 맨 앞의 "P" 를 제외한 모든 "P" 를 '(' 로 바꾸었을 때 첫 'P' 뒤에 나오는 문자열 (이 문자열을 "S로부터 생성된 괄호 문자열" 이라 한다) 이 올바른 괄호 문자열이다.
3. S에서 임의의 위치에 있는 "PPAP" 를 "P" 로 바꾸는 과정을 가능한 만큼 최대한 반복했을 때, 남는 문자열이 "P" 이다.

Div1C/Div2G. PPAP

- $(1) \to (2)$
- 수학적 귀납법: 길이 $3n+1$ 인 PPAP 문자열이 (2) 를 만족함을 보입니다.
- $n = 0$: “P” 는 (2) 조건 만족 (빈 문자열은 올바른 괄호 문자열)
- $n = k$일 때 성립 가정, $n = k+1$ 일 때, PPAP 문자열을 만들 때 마지막 바꾸기 과정 직전의 문자열을 생각합니다.
- 이 문자열은 (2) 를 만족합니다.(귀납 가정)
- “P” 를 “PPAP” 로 바꾸는 것은 결국 ‘P’ 뒤에 “PAP” 를 넣는 것이므로, 여전히 조건 1, 2를 만족하며, 최종 문자열로부터 생성된 괄호 문자열은 직전의 괄호 문자열의 특정 위치에 “()” 를 넣은 것입니다.
- 그러므로 역시 올바른 괄호 문자열입니다. Q. E. D.

Div1C/Div2G. PPAP

- $(2) \to (3)$
- 수학적 귀납법: 생성된 괄호 문자열의 길이가 $2n$일 때 (3)을 만족함을 보입니다.
- $n = 0$: “P”는 (3) 조건 만족
- $n = k$일 때 성립 가정, $n = k + 1$일 때, “PPAP”를 “P”로 바꾸는 것은 결국 ‘P’ 뒤에 있는 “PAP”를 없애는 것이므로, 한 번 바꾼 다음의 문자열로부터 생성된 문자열은 이전 문자열에서 “()”를 하나 뺀 것이므로 올바른 괄호 문자열입니다.
- 그러므로 귀납 가정에 의해 “PPAP”를 “P”로 한 번 바꾼 문자열은 (3)을 만족합니다. 그러므로 원래 문자열도 (3)을 만족합니다. Q. E. D.

Div1C/Div2G. PPAP

- $(3) \to (1)$
- 바꾸는 과정의 정확히 역순으로 “P” 를 “PPAP” 로 바꾸면 원래 문자열이 만들어집니다. 그러므로 PPAP 문자열입니다. Q. E. D.

Div1C/Div2G. PPAP

- $(3) \rightarrow (1)$
- 바꾸는 과정의 정확히 역순으로 “P” 를 “PPAP” 로 바꾸면 원래 문자열이 만들어집니다. 그러므로 PPAP 문자열입니다. Q. E. D.
- 그러므로 세 조건은 동치입니다.

Div1C/Div2G. PPAP

- 그러므로 다음과 같은 풀이들이 모두 맞습니다.

Div1C/Div2G. PPAP

- 그러므로 다음과 같은 풀이들이 모두 맞습니다.
- 앞에서부터 문자를 하나씩 스택에 push하며, 스택의 마지막 네 개의 문자가 “PPAP” 일 때마다 “PAP” 를 pop했을 때 남은 문자열이 “P” 이면 PPAP 문자열, 아니면 NP

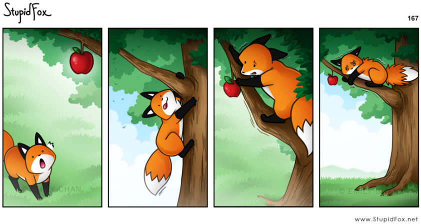
Div1C/Div2G. PPAP

- 그러므로 다음과 같은 풀이들이 모두 맞습니다.
- 앞에서부터 문자를 하나씩 스택에 push하며, 스택의 마지막 네 개의 문자가 “PPAP”일 때마다 “PAP”를 pop했을 때 남은 문자열이 “P”이면 PPAP 문자열, 아니면 NP
- 조건 1, 2를 만족함을 확인한 뒤, x=0에서 시작하여, 문자를 하나씩 읽으며, ‘P’를 보면 앞 문자가 ‘P’인 경우 x++, ‘A’인 경우 x--를 했을 때 x가 항상 0 이상이고 최종 x가 0이면 PPAP 문자열, 아니면 NP

Div1A/Div2H. 달빛 여우

- 제출 횟수 : 18
- 맞은 참가자 수 : 2
- 정답률 : 11.111%
- 처음 맞은 참가자 : 윤민혁 (2:07)
- 출제자 : doju

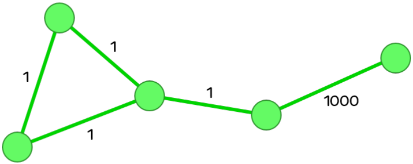
Div1A/Div2H. 달빛 여우



- 여우는 사랑입니다.
- 지문을 통해 여러분의 순수함을 끌어내고자 노력했습니다.

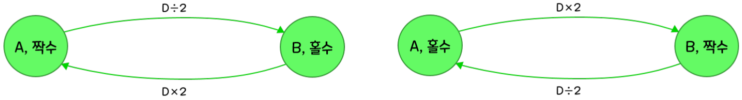
Div1A/Div2H. 달빛 여우

- 달빛 여우가 각 그루터기까지 이동하는 데 걸리는 시간은 간단한 최단 거리 문제입니다.
- 그러나 달빛 늑대는 필요하다면 같은 그루터기를 두 번 지날 수도 있습니다.



Div1A/Div2H. 달빛 여우

- 달빛 늑대가 짝수 개의 오솔길을 지나서 같은 그루터기로 돌아오는 것은 손해입니다.
- 각 그루터기에 홀수 개의 오솔길을 지나 도착하는 경우와 짝수 개의 오솔길을 지나 도착하는 경우로 나눠서 그래프를 구성하면 됩니다.



Div1L/Div2I. 뚜루루 뚜루

- 제출 횟수 : 6
- 맞은 참가자 수 : 2
- 정답률 : 33.333%
- 처음 맞은 참가자 : 이정민 (2:52)
- 출제자 : doju

Div1L/Div2I. 뚜루루 뚜루

- 각각의 **뚜**를 시작점으로 잡고 깊이 우선 탐색 등의 방법으로 뚜루루 뚜루 경로를 세면 문제를 풀 수 있습니다.
- 하지만 종이가 최대 약 1.5억 개의 칸을 가질 수 있고, 정직하게 세면 시간 초과가 납니다.

Div1L/Div2I. 뚜루루 뚜루

뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜
루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루
뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루
루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜
루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루
뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜
루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루
뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루
루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜
루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루
뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜
루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루	루
뚜	루	뚜	루	루	뚜	루	뚜	루	루	뚜	루	뚜	루

- 문자열의 길이가 5글자이기 때문에 5×5 크기의 패턴이 반복됩니다.

Div1L/Div2I. 뚜루루 뚜루

4×4	4×5		4×4
5×4	5×5		5×4
4×4	4×5		4×4

- 따라서 실제로 시작점으로 따져 봐야 하는 범위는 그다지 넓지 않습니다.
- 최대 13×13개 칸을 시작점으로 잡아 보면 전체의 답을 계산할 수 있습니다.

Div1L/Div2I. 뚜루루 뚜루

- 시작점이 아니라 가운데의 **루**를 기준으로 잡으면 좀 더 간단하게 풀 수 있습니다.

뚜 ← 루 ← **루** → 뚜 → 루

- 깊이 우선 탐색이나 각 칸의 방문 여부 체크도 필요하지 않고, 9×9개 칸만 미리 검사하면 문제를 풀 수 있습니다.

Div1B/Div2J. Cherrypick

- 제출 횟수: 14
- 맞은 참가자 수: 0
- 정답률: 0.000%
- 처음 맞은 참가자: -
- 출제자: kazel

Div1B/Div2J. Cherrypick

- 기본 아이디어

Div1B/Div2J. Cherrypick

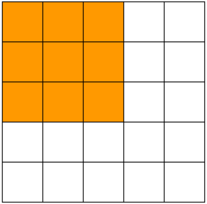
- 기본 아이디어
- 어떤 점을 중심으로 $2K-1$ by $2K-1$ 정사각형 영역 안에서의 최댓값을 구하면?

Div1B/Div2J. Cherrypick

- 기본 아이디어
- 어떤 점을 중심으로 $2K-1$ by $2K-1$ 정사각형 영역 안에서의 최댓값을 구하면?
- K by K 정사각형으로 중심이 된 점과 최댓값을 가지고 있는 점을 모두 포함시킬 수 있다

Div1B/Div2J. Cherrypick

- 기본 아이디어
- 어떤 점을 중심으로 $2K-1$ by $2K-1$ 정사각형 영역 안에서의 최댓값을 구하면?
- K by K 정사각형으로 중심이 된 점과 최댓값을 가지고 있는 점을 모두 포함시킬 수 있다



Div1B/Div2J. Cherrypick

- 예전의 풀이

Div1B/Div2J. Cherrypick

- 예전의 풀이
- 2차원 sparse table을 만들자

Div1B/Div2J. Cherrypick

- 예전의 풀이
- 2차원 sparse table을 만들자
- 원하는 정사각형 영역 안의 최댓값을 $O(1)$에 찾을 수 있다

Div1B/Div2J. Cherrypick

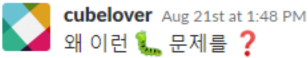
- 예전의 풀이
- 2차원 sparse table을 만들자
- 원하는 정사각형 영역 안의 최댓값을 $O(1)$에 찾을 수 있다
- 각 점에 대한 쿼리를 그 점을 중심으로 하는 모든 정사각형을 보면서 처리하면 $O(n)$만큼 걸린다

Div1B/Div2J. Cherrypick

- 예전의 풀이
- 2차원 sparse table을 만들자
- 원하는 정사각형 영역 안의 최댓값을 $O(1)$에 찾을 수 있다
- 각 점에 대한 쿼리를 그 점을 중심으로 하는 모든 정사각형을 보면서 처리하면 $O(n)$만큼 걸린다
- 현실 : 정사각형 영역 안의 최댓값 찾는데 최소 8개의 쿼리 필요 $\rightarrow$ 킹짱느림

Div1B/Div2J. Cherrypick

- 예전의 풀이
- 2차원 sparse table을 만들자
- 원하는 정사각형 영역 안의 최댓값을 $O(1)$에 찾을 수 있다
- 각 점에 대한 쿼리를 그 점을 중심으로 하는 모든 정사각형을 보면서 처리하면 $O(n)$만큼 걸린다
- 현실 : 정사각형 영역 안의 최댓값 찾는데 최소 8개의 쿼리 필요
  $\rightarrow$ 킹짱느림

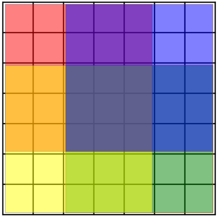


Div1B/Div2J. Cherrypick

- $2K+1$ by $2K+1$ 영역 안의 최댓값을 구하는 다른 방법

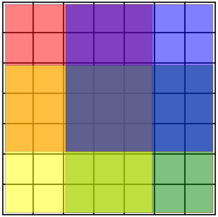
Div1B/Div2J. Cherrypick

- $2K+1$ by $2K+1$ 영역 안의 최댓값을 구하는 다른 방법



Div1B/Div2J. Cherrypick

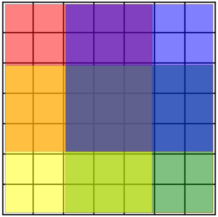
- $2K+1$ by $2K+1$ 영역 안의 최댓값을 구하는 다른 방법



- $2K-1$ by $2K-1$ 영역 4개로 $2K+1$ by $2K+1$ 영역의 최댓값을 계산할 수 있다

Div1B/Div2J. Cherrypick

- $2K+1$ by $2K+1$ 영역 안의 최댓값을 구하는 다른 방법



- $2K-1$ by $2K-1$ 영역 4개로 $2K+1$ by $2K+1$ 영역의 최댓값을 계산할 수 있다
- 3 by 3 영역의 최댓값을 구하고 나면 필요한 모든 정사각형 영역의 최댓값을 계산할 수 있다

Div1B/Div2J. Cherrypick

- 총 시간복잡도 $O(n^3)$: Too slow!

Div1B/Div2J. Cherrypick

- 총 시간복잡도 $O(n^3)$: Too slow!
- 체리의 최대 당도가 1000 이하

Div1B/Div2J. Cherrypick

- 총 시간복잡도 $O(n^3)$: Too slow!
- 체리의 최대 당도가 1000 이하
- $\sqrt{1000} \approx 31.6$ 이상은 볼 필요가 없다

Div1B/Div2J. Cherrypick

- 총 시간복잡도 $O(n^3)$: Too slow!
- 체리의 최대 당도가 1000 이하
- $\sqrt{1000} \approx 31.6$ 이상은 볼 필요가 없다
- 총 시간복잡도 $O(n^2\sqrt{1000})$

Div1A/Div2H. 달빛 여우

- 제출 횟수 : 35
- 맞은 참가자 수 : 23
- 정답률 : 65.714%
- 처음 맞은 참가자 : 조승현 (0:08)
- 출제자 : doju

Div1B/Div2J. Cherrypick

- 제출 횟수 : 52
- 맞은 참가자 수 : 12
- 정답률 : 23.077%
- 처음 맞은 참가자 : 조승현 (0:27)
- 출제자 : kazel

Div1C/Div2G. PPAP

- 제출 횟수 : 45
- 맞은 참가자 수 : 24
- 정답률 : 53.333%
- 처음 맞은 참가자 : 박성관 (0:04)
- 출제자 : 16silver

Div1D. 사무실 이전

- 제출 횟수 : 35
- 맞은 참가자 수 : 18
- 정답률 : 51.429%
- 처음 맞은 참가자 : 최석환 (0:50)
- 출제자 : kazel

Div1D. 사무실 이전

- 루트에 연결된 어떤 한 서브트리에 속하면서 거리 i에 있는 사무실 후보의 개수를 X_i, 집 개수를 Y_i라고 하자

Div1D. 사무실 이전

- 루트에 연결된 어떤 한 서브트리에 속하면서 거리 i에 있는 사무실 후보의 개수를 X_i, 집 개수를 Y_i라고 하자
- 모든 서브트리 쌍에 대해서 $\sum\sum(X_i \cdot Y_j) * (i+j)^2$을 구하면 이 정점을 통과하는 모든 경로의 값을 계산 가능

Div1D. 사무실 이전

- 루트에 연결된 어떤 한 서브트리에 속하면서 거리 i에 있는 사무실 후보의 개수를 X_i, 집 개수를 Y_i라고 하자
- 모든 서브트리 쌍에 대해서 $\sum\sum(X_i \cdot Y_j) * (i+j)^2$을 구하면 이 정점을 통과하는 모든 경로의 값을 계산 가능
- 그렇다면 어떤 서브트리 쌍에서 거리 D만큼 떨어진 {사무실 후보, 집} 쌍의 개수는?

Div1D. 사무실 이전

- 루트에 연결된 어떤 한 서브트리에 속하면서 거리 i에 있는 사무실 후보의 개수를 X_i, 집 개수를 Y_i라고 하자
- 모든 서브트리 쌍에 대해서 $\sum\sum(X_i \cdot Y_j) * (i+j)^2$을 구하면 이 정점을 통과하는 모든 경로의 값을 계산 가능
- 그렇다면 어떤 서브트리 쌍에서 거리 D만큼 떨어진 {사무실 후보, 집} 쌍의 개수는?

$$\sum_{i=0}^{D}(X_i \cdot Y_{D-i})$$

Div1D. 사무실 이전

- 루트에 연결된 어떤 한 서브트리에 속하면서 거리 i에 있는 사무실 후보의 개수를 X_i, 집 개수를 Y_i라고 하자
- 모든 서브트리 쌍에 대해서 $\sum\sum(X_i \cdot Y_j) * (i+j)^2$ 을 구하면 이 정점을 통과하는 모든 경로의 값을 계산 가능
- 그렇다면 어떤 서브트리 쌍에서 거리 D만큼 떨어진 {사무실 후보, 집} 쌍의 개수는?

$$\sum_{i=0}^{D}(X_i \cdot Y_{D-i})$$

- 어디선가 많이 보던 식인데...?

Div1D. 사무실 이전

- 답은 FFT다!

Div1D. 사무실 이전

- 답은 FFT다!

Div1D. 사무실 이전

- 답은 FFT다!

- $O(n \log^2 n)$ 까지 줄일 수 있지만 여전히 너무 느리다

Div1D. 사무실 이전

- 답은 FFT다!

- $O(n \log^2 n)$ 까지 줄일 수 있지만 여전히 너무 느리다
- 최대한 최적화했을 때 4.8초 정도 걸림

Div1D. 사무실 이전

- 식을 조금 바꿔보자

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i번째 집까지의 거리를 H_i라고 하자

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i번째 집까지의 거리를 H_i라고 하자
- 구해야 하는 식은 $\sum \sum (O_i + H_j)^2$

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i번째 집까지의 거리를 H_i라고 하자
- 구해야 하는 식은 $\sum\sum(O_i + H_j)^2$
- $Sqsum(X) = \sum X_i^2$, $Sum(X) = \sum X_i$, $Count(X)$ = X 집합에 속한 값의 개수로 정의하자

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i 번째 집까지의 거리를 H_i라고 하자
- 구해야 하는 식은 $\sum\sum(O_i + H_j)^2$
- $Sqsum(X) = \sum X_i^2$, $Sum(X) = \sum X_i$, $Count(X) =$ X 집합에 속한 값의 개수로 정의하자
- $\sum\sum(O_i + H_j)^2 =$

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i번째 집까지의 거리를 H_i라고 하자
- 구해야 하는 식은 $\sum\sum(O_i + H_j)^2$
- $Sqsum(X) = \sum X_i^2$, $Sum(X) = \sum X_i$, $Count(X) =$ X 집합에 속한 값의 개수로 정의하자
- $\sum\sum(O_i + H_j)^2 =$
$Count(H) \cdot Sqsum(O) + 2 \cdot Sum(O) \cdot Sum(H) +$
$Count(O) \cdot Sqsum(H)$

Div1D. 사무실 이전

- 식을 조금 바꿔보자
- 어떤 한 서브트리에 속한 i번째 사무실 후보까지의 거리를 O_i, i 번째 집까지의 거리를 H_i라고 하자
- 구해야 하는 식은 $\sum\sum(O_i + H_j)^2$
- $Sqsum(X) = \sum X_i^2$, $Sum(X) = \sum X_i$, $Count(X)$ = X 집합에 속한 값의 개수로 정의하자
- $\sum\sum(O_i + H_j)^2 =$

  $Count(H) \cdot Sqsum(O) + 2 \cdot Sum(O) \cdot Sum(H) +$

  $Count(O) \cdot Sqsum(H)$

  $\rightarrow$ 위의 세 함수만으로 계산 가능!

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?
- 원하는 값은 $\sum (X_i + 1)^2$

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?
- 원하는 값은 $\sum(X_i + 1)^2$

  $= Sqsum(X) + 2 * Sum(X) + Count(X)$

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?
- 원하는 값은 $\sum(X_i + 1)^2$

  $= Sqsum(X) + 2 * Sum(X) + Count(X)$
- 역시 갱신 가능

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?
- 원하는 값은 $\sum (X_i + 1)^2$

  $= Sqsum(X) + 2 * Sum(X) + Count(X)$
- 역시 갱신 가능
- Centroid Decomposition을 끼얹으면 원하는 답을 $O(n \log n)$에 찾을 수 있다

Div1D. 사무실 이전

- 따라서 노드당 값 6개만 있으면 원하는 답을 얻을 수 있다
- Sum과 $Count$의 갱신은 trivial. 하지만 $Sqsum$은?
- 원하는 값은 $\sum (X_i + 1)^2$

  $= Sqsum(X) + 2 * Sum(X) + Count(X)$
- 역시 갱신 가능
- Centroid Decomposition을 끼얹으면 원하는 답을 $O(n \log n)$에 찾을 수 있다
- rooted tree에서 위로 올라가는 경로가 포함되는 경우와 아닌 경우를 분리하면 DFS 두 번으로도 가능

  이렇게 풀면 시간복잡도는 $O(n)$

Div1E. Unary

- 제출 횟수 : 41
- 맞은 참가자 수 : 17
- 정답률 : 80.488%
- 처음 맞은 참가자 : 시제연 (0:27)
- 출제자 : doju

Div1E. Unary

- 연산자 두 개를 조합해 봅시다.
  - `~-x = x - 1`
  - `~~x, --x = x`
  - `-~x = x + 1`
- `~-~-~-x` 또는 `-~-~-~x` 꼴을 만들고 `--`나 `~~`를 끼워넣으면 될 것만 같습니다.

Div1E. Unary

doju 2:02 PM
Unary 검수해 줘여

kaazel 2:02 PM
폴리곤이 죽어서

아무것도모타는 상황

doju 2:03 PM
세팅하고 나니까 Div 1에서 제일 쉬운 문제 같음

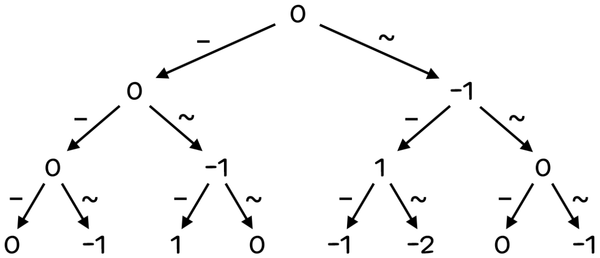
kaazel 2:03 PM
안말리면쉽죠

doju 2:03 PM
다들 말렸으면 좋겠다

😈 1

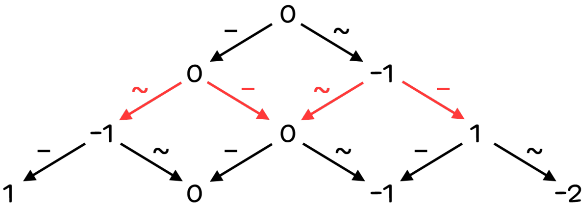
Div1E. Unary

- 연산자를 한 개씩 추가해 봅시다.



Div1E. Unary

- 연산자의 순서를 바꿔 봅시다.



- 답이 깔끔한 이항계수 꼴로 나옴을 알 수 있습니다.

Div1F. 피타고라스 쌍

- 제출 횟수 : 4
- 맞은 참가자 수 : 0
- 정답률 : 0.000%
- 처음 맞은 참가자 : -
- 출제자 : kipa00

Div1F. 피타고라스 쌍

양의 정수 L이 주어지고, $1 \le n < m \le L$인 정수 n, m에 대해,

$$
\begin{aligned}
a &= m^2 - n^2 \\
b &= 2mn \\
c &= m^2 + n^2
\end{aligned}
$$

이라 할 때 $\gcd(a, b, c) = 1$인 것의 개수를 구하는 문제

Div1F. 피타고라스 쌍

Theorem

$\gcd(a, b, c) = 1$일 필요충분조건은 $\gcd(n, m) = 1$이고 n과 m의 홀짝성이 다른 것이다.

따라서

- $n, m \le L$이고 $\gcd(n, m) = 1$인 것의 개수에서
- $n, m \le L$이고 $\gcd(n, m) = 1$이고 n과 m이 모두 홀수인 것의 개수를 뺀 다음

2로 나누면 정답을 구할 수 있습니다

Div1F. 피타고라스 쌍

$n, m \leq L$ 이고 $\gcd(n, m) = 1$ 인 것의 개수:

- 포함-배제 원리를 이용하면 (p_i 소수)

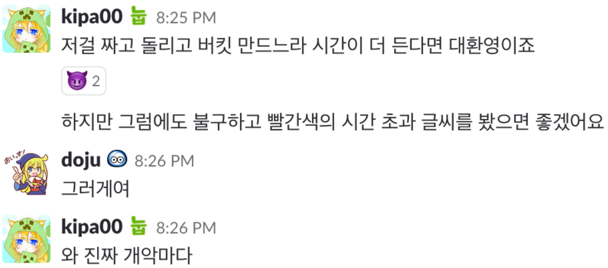
$$L^2 - \sum_{p_1} \left\lfloor \frac{L}{p_1} \right\rfloor^2 + \sum_{p_1, p_2} \left\lfloor \frac{L}{p_1 p_2} \right\rfloor^2 - \sum_{p_1, p_2, p_3} \left\lfloor \frac{L}{p_1 p_2 p_3} \right\rfloor^2 + \cdots$$

- 산술의 기본 정리에 의해 시그마 식의 분모가 항상 유일한 값을 만들어냅니다
- $\mu(i)$ 를 위 식의 분모가 i 일 때의 계수라고 하면 (없으면 0),

$$\sum_{i=1}^{L} \mu(i) \left\lfloor \frac{L}{i} \right\rfloor^2$$

- 시간 복잡도: $\mu(i)$ 의 특성을 잘 생각하면 $O(L \log \log L)$

Div1F. 피타고라스 쌍



- 출제자의 예상: 여기까지 생각하고 버킷을 이용해서 문제를 뚫을 것이다
- $b = 5 \cdot 10^6$ 개의 정수를 소수인지 아닌지 판별하는 데 $O(b \log \log L)$ 이므로 시간 안에 돌릴 수는 있습니다
- 이게 정해였으면 문제 안 냈습니다

Div1F. 피타고라스 쌍

$$\sum_{i=1}^{L} \mu(i) \left\lfloor \frac{L}{i} \right\rfloor^2 = \sum_{i=1}^{a_L - 1} \mu(i) \left\lfloor \frac{L}{i} \right\rfloor^2 + \sum_{j=1}^{\lfloor n/a_L \rfloor} \left(M\left(\frac{n}{j}\right) - M\left(\frac{n}{j+1}\right)\right) j^2$$

- $\left\lfloor \frac{L}{i} \right\rfloor$ 의 값은 i가 커질수록 그렇게 자주 변하지 않습니다
- $M(x)$ 는 x 이하의 모든 μ 값의 합, a_L 은 $\left\lfloor \frac{L}{a_L - 1} \right\rfloor \neq \left\lfloor \frac{L}{a_L} \right\rfloor$ 인 상수
- 필요한 모든 값이 계산되고 나면, a_L 을 $\sqrt{L}$에 가깝게 골라 $O(\sqrt{L})$ 만에 문제를 해결할 수 있습니다

Div1F. 피타고라스 쌍

Theorem

모든 자연수 n에 대해,

$$\sum_{i=1}^{n} M\left(\frac{n}{i}\right) = 1$$

이 성립한다.

- $b_L \geq \sqrt{L}$까지 전처리를 한다고 생각했을 때, 시간 복잡도가 최소가 되는 $b_L \approx L^{2/3}$입니다
- $b_L = L^{2/3}$일 때, $O(L^{2/3} \log\log L)$의 시간 복잡도로 문제를 해결할 수 있습니다

Div1F. 피타고라스 쌍

$n, m \le L$이고 $\gcd(n, m) = 1$이며 n과 m이 모두 홀수인 것의 개수:

- 역시 포함-배제 원리를 이용하면 (p_i는 3 이상의 소수)

$$\left\lfloor \frac{L+1}{2} \right\rfloor^2 - \sum_{p_1} \left\lfloor \frac{\left\lfloor \frac{L}{p_1} \right\rfloor + 1}{2} \right\rfloor^2 + \sum_{p_1, p_2} \left\lfloor \frac{\left\lfloor \frac{L}{p_1 p_2} \right\rfloor + 1}{2} \right\rfloor^2 - \cdots$$

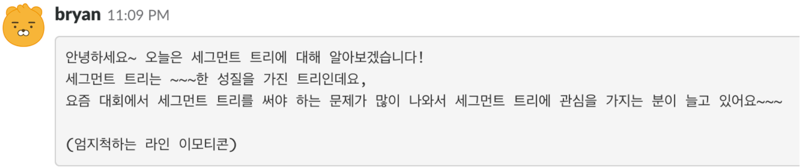
- 똑같이 정리하면, 비슷한 식이 나오고, 대신 x 이하의 홀수들의 μ 값을 모두 더한 M' 이라는 함수가 필요
- 모든 실수 x에 대해 $M'(x) = M(x) + M'(x/2)$가 성립

http://snups.snucse.org/snupc2018/Div1F.pdf
위 링크에서 모든 세부사항을 확인할 수 있습니다.
시간 복잡도 $O(L^{2/3} \log\log L)$

Div1G. 나는 행복합니다

- 제출 횟수 : 30
- 맞은 참가자 수 : 2
- 정답률 : 6.667%
- 처음 맞은 참가자 : 박성관 (0:51)
- 출제자 : bryan

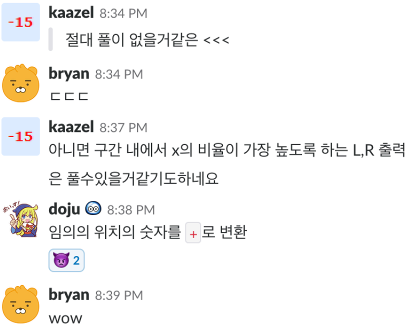
Div1G. 나는 행복합니다



- n자리의 수가 주어지고, 다음 두 가지 쿼리를 처리하는 문제
    - 특정 구간의 모든 숫자 f를 숫자 t로 바꿈
    - 특정 구간의 정수를 $998,244,353$으로 나눈 나머지 출력
- 전형적인 세그먼트 트리 문제
- 각 구간마다, 어떤 숫자를 어떤 숫자로 바꾸는지를 저장하기 위한 크기 10의 배열을 만들고 이를 lazy propagation 때 잘 갱신하면 됩니다

Div1G. 나는 행복합니다

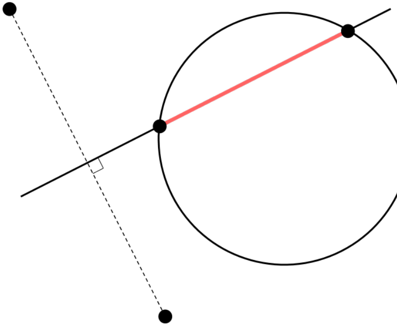
- square root decomposition 으로도 통과할 수 있습니다
- 악마같이 **임의 위치의 숫자를 + 로 변환** 쿼리를 넣으려 했던 출제진들을 bryan이 막았습니다
- 아프리카에 있는 bryan에게 감사합시다



Div1H. 영점사격

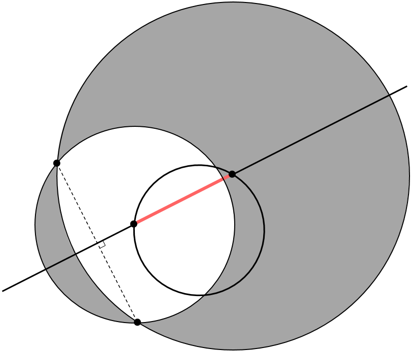
- 제출 횟수 : 34
- 맞은 참가자 수 : 2
- 정답률 : 5.882%
- 처음 맞은 참가자 : 시제연 (3:41)
- 출제자 : doju

Div1H. 영점사격



- 외심에서 두 점까지의 거리는 같으므로 외심은 항상 두 점을 잇는 선분의 수직이등분선 위에 있습니다.

Div1H. 영점사격



- 다음과 같은 영역의 넓이를 구하면 됩니다.
    - 증명은 원주각을 생각하면 됩니다.
- 구하는 과정은 단순 계산이므로 생략합니다.

Div1H. 영점사격

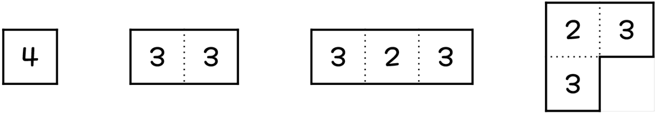
- 처음에는 좌표 범위 10^4, 오차 범위 10^{-6} 이었으나 `double`로 안정적으로 통과할 수 있도록 범위를 크게 줄였습니다.

Div1I. 동아리방 확장

- 제출 횟수 : 30
- 맞은 참가자 수 : 9
- 정답률 : 30.000%
- 처음 맞은 참가자 : 조승현 (1:52)
- 출제자 : doju

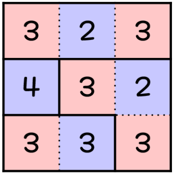
Div1I. 동아리방 확장

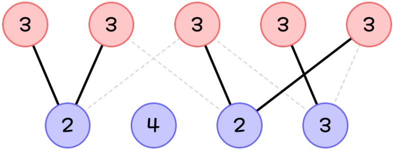
- 방의 크기가 3 이하이므로 다음과 같은 네 종류의 방만 만들 수 있습니다.



- 2가 적힌 방은 반드시 3이 적힌 방 두 개와 연결되어야 한다는 특징이 있습니다.

Div1I. 동아리방 확장





- 홀짝성으로 이분그래프를 구성하고 모든 정점이 주어진 용량만큼 매칭되는지 확인해 보면 됩니다.

Div1J. 미생물 키우기

- 제출 횟수 : 22
- 맞은 참가자 수 : 1
- 정답률 : 13.636%
- 처음 맞은 참가자 : ?
- 출제자 : cki86201

Div1J. 미생물 키우기

- 어떤 미생물에 대해, 그 미생물을 만드는 비용은 이미 있는 미생물의 종류와만 관련이 있습니다.

Div1J. 미생물 키우기

- 어떤 미생물에 대해, 그 미생물을 만드는 비용은 이미 있는 미생물의 종류와만 관련이 있습니다.
- 즉 같은 미생물이 1개든 2개든 앞으로의 비용에 영향을 주지 않습니다.

Div1J. 미생물 키우기

- 어떤 미생물에 대해, 그 미생물을 만드는 비용은 이미 있는 미생물의 종류와만 관련이 있습니다.
- 즉 같은 미생물이 1개든 2개든 앞으로의 비용에 영향을 주지 않습니다.
- 어떤 미생물이 이미 방에 있다면, 그 미생물을 하나 더 만드는 순간은 최대한 늦춰도 상관이 없습니다.

Div1J. 미생물 키우기

- 모든 미생물을 하나씩 만들어 놓은 후 나머지 미생물들은 그리디하게 선택하면 됩니다.
- 모든 미생물을 1개씩 만드는 비용을 구해 봅시다.

Div1J. 미생물 키우기

- 가상의 $N + 1$번 미생물을 만들고, 처음에는 $N + 1$번 미생물 1개만 있다고 합시다.

Div1J. 미생물 키우기

- 가상의 $N+1$번 미생물을 만들고, 처음에는 $N+1$번 미생물 1개만 있다고 합시다.
- y_i를 i번 미생물을 사 오는 비용이 아니라, $N+1$번 미생물이 i번 미생물을 만드는 비용이라고 합시다.

Div1J. 미생물 키우기

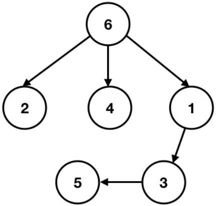
- 가상의 $N+1$번 미생물을 만들고, 처음에는 $N+1$번 미생물 1 개만 있다고 합시다.
- y_i를 i번 미생물을 사 오는 비용이 아니라, $N+1$번 미생물이 i번 미생물을 만드는 비용이라고 합시다.
- 다시 말해, $z_{N+1,i} = y_i$입니다.

Div1J. 미생물 키우기

- i번 미생물이 j번 미생물을 만드는 비용이 $z_{i,j}$입니다.
- 이를 i에서 j로 가는 가중치 $z_{i,j}$인 방향 간선으로 생각해 봅시다.

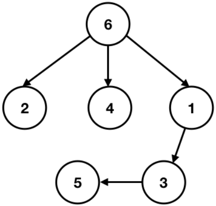
Div1J. 미생물 키우기

- 우리가 사용하는 간선들을 생각해 봅시다.
- $N+1$을 루트로 하고 뻗어나가는 트리 모양이 됩니다.
- 이 형태의 그래프를 Arborescence라고 합니다.



Div1J. 미생물 키우기

- 우리가 사용하는 간선들을 생각해 봅시다.
- $N+1$을 루트로 하고 뻗어나가는 트리 모양이 됩니다.
- 이 형태의 그래프를 Arborescence라고 합니다.



- 만약 모든 i, j에 대해 $z_{i,j} = z_{j,i}$이었다면 MST가 될 것입니다.

Div1J. 미생물 키우기

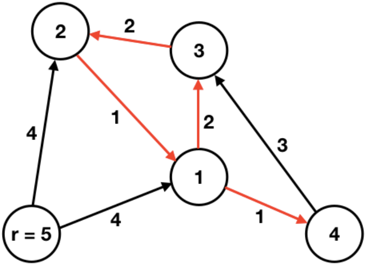
- 그래프가 방향 그래프여도 Edmonds' algorithm을 통해 MST를 구할 수 있습니다.
- 이 알고리즘을 정직하게 구현하면 $O(EV)$ 입니다.

Div1J. 미생물 키우기

- 그래프가 방향 그래프여도 Edmonds' algorithm을 통해 MST 를 구할 수 있습니다.
- 이 알고리즘을 정직하게 구현하면 $O(EV)$ 입니다.
- 참고로 이를 잘 구현하면 $O(E + V \log V)$ 로까지 줄어드는데, 이는 무방향 그래프 MST와 같은 시간복잡도입니다.
- 다행히도 $O(EV)$ 로 안 풀리는 문제는 아직 못 봤습니다.

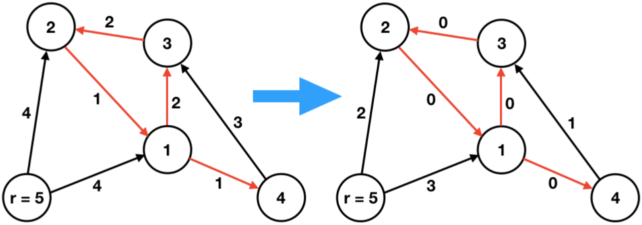
Div1J. 미생물 키우기

- 그래프 V, E와 루트 r이 주어졌을 때 r에서 뻗어나가는 스패닝 트리들 중 간선 가중치 합이 최소인 트리를 구하면 됩니다.
- r을 제외한 임의의 정점 u에 대해, u로 오는 간선 중 가중치의 최솟값을 $\delta(u)$라고 합시다.
- 아래 그림에서 $\delta(1) = 1, \delta(2) = 2, \delta(3) = 2, \delta(4) = 1$입니다.



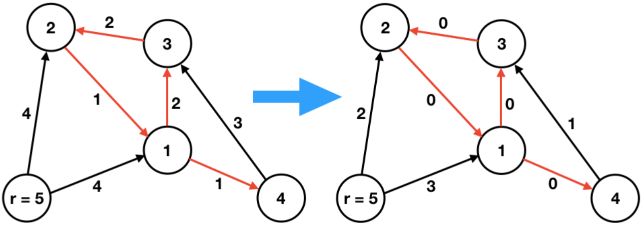
Div1J. 미생물 키우기

- 모든 간선에 대해, u에서 v로 가는 가중치 w인 간선이라면
- 이 가중치를 $w - \delta(v)$로 만든 새로운 그래프를 생각해 봅시다.



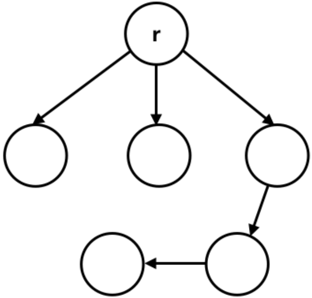
Div1J. 미생물 키우기

- 아래 두 성질을 보일 수 있습니다.
  1. 원래 그래프와 새로운 그래프는 같은 Directed MST를 갖습니다.
  2. 두 Directed MST 값의 차이는 $\sum \delta(u)$ 입니다.



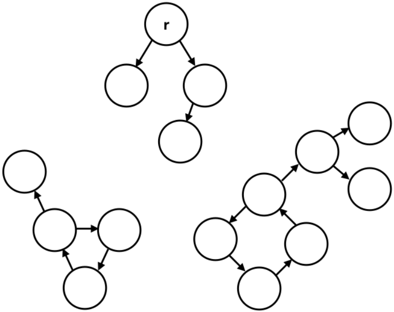
Div1J. 미생물 키우기

- r아닌 각 정점에 대해, 그 정점에서 나가는 간선 중 하나는 가중치가 0입니다.
- 이 간선들만 모아놨을 때 사이클이 없다면, 이것은 0짜리 Directed MST이므로 답이 0입니다.



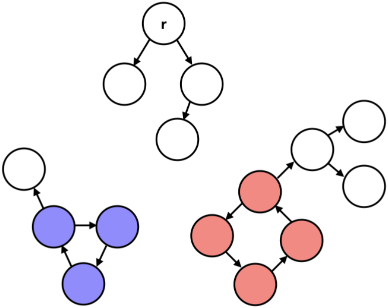
Div1J. 미생물 키우기

- 그 외의 경우는 이 간선들이 아래 그래프처럼 나올 것입니다.



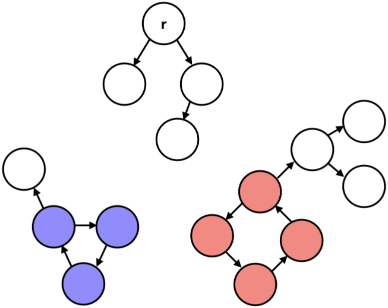
Div1J. 미생물 키우기

- 사이클에 주목해 봅시다.



Div1J. 미생물 키우기

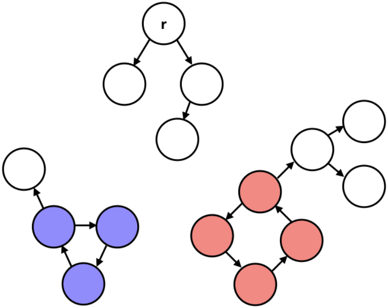
- 사이클에 주목해 봅시다.



- 그림에서 표현된 간선들은 모두 가중치가 0입니다.

Div1J. 미생물 키우기

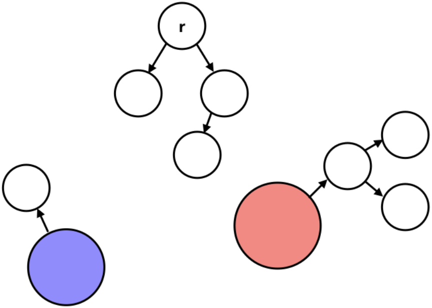
- 사이클에 주목해 봅시다.



- 그림에서 표현된 간선들은 모두 가중치가 0입니다.
- 따라서 이 사이클은 하나의 정점으로 합칠 수 있습니다.

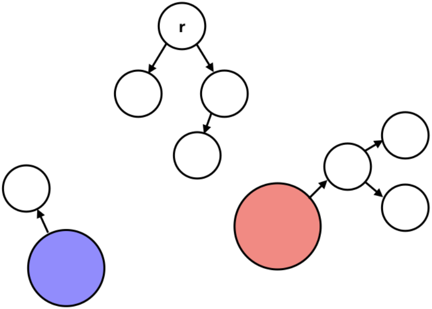
Div1J. 미생물 키우기

- 합친 다음에는 처음으로 돌아가 다시 Directed MST를 구합니다.



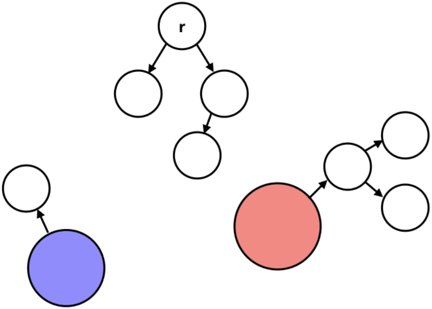
Div1J. 미생물 키우기

- 사이클들을 합칠 때마다 정점 개수가 하나 이상 줄어듭니다.
- 따라서 $V-1$번 이내에 알고리즘이 끝납니다.



Div1J. 미생물 키우기

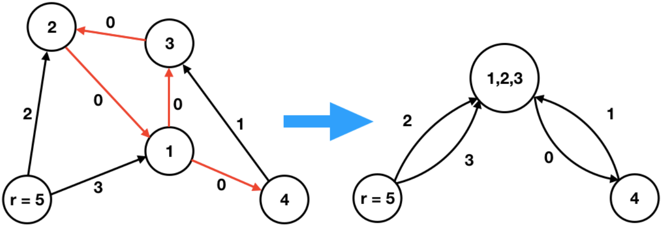
- 사이클들을 합칠 때마다 정점 개수가 하나 이상 줄어듭니다.
- 따라서 $V-1$ 번 이내에 알고리즘이 끝납니다.



- 따라서 시간복잡도는 $O(EV)$ 입니다.

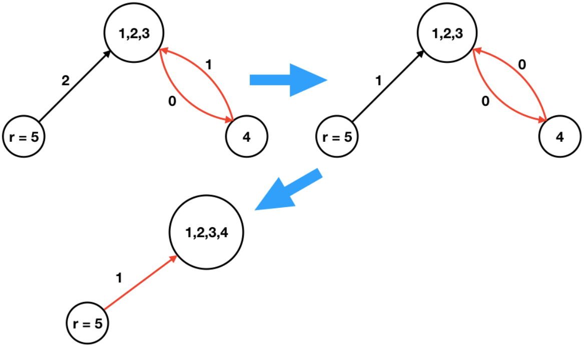
Div1J. 미생물 키우기

- 정점을 합치는 예시입니다.



Div1J. 미생물 키우기

- 정점을 합치는 예시입니다.



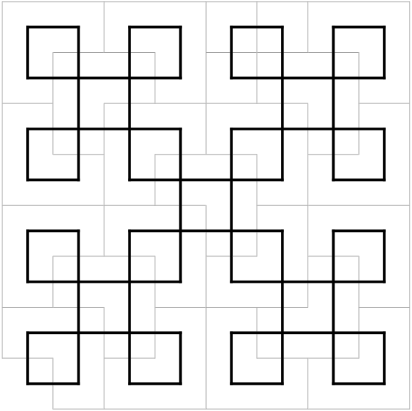
Div1J. 미생물 키우기

- 이 문제에서는 안 해도 되지만, 역추적이 까다롭습니다.
- 이 알고리즘으로 풀리는 문제는 NEERC 13 D. Dictionary 등이 있습니다.

Div1K. 스눕시티

- 제출 횟수 : 20
- 맞은 참가자 수 : 5
- 정답률 : 25.000%
- 처음 맞은 참가자 : 최석환 (3:02)
- 출제자 : doju

Div1K. 스눕시티

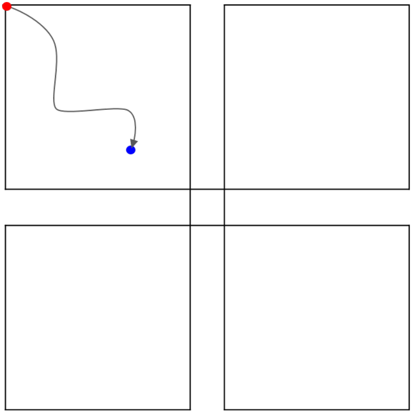


- 건물을 회전시키는 것은 빈 칸이 이동하는 것으로 생각할 수 있습니다.
- 빈 칸이 이동할 수 있는 경로는 프랙탈 형태의 그래프입니다.
- 불가능한 경우가 없음은 자명합니다.

Div1K. 스눕시티

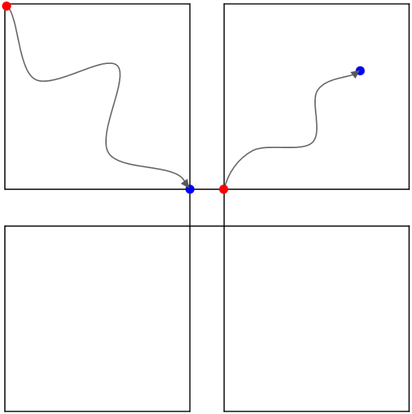
- 두 점이 중심을 기준으로 다른 사분면에 있다면 반드시 중심의 사각형을 지나가야 합니다.
- 중심의 사각형을 기준으로 경로를 분할하면 $2^{N-1} \times 2^{N-1}$ 그래프에서 두 점 사이의 최단거리를 구하는 문제 두 개로 분리됩니다.
- 단, 새로운 문제에서는 출발점이 항상 가장 구석의 점으로 고정됩니다.

Div1K. 스눕시티



- 도착점이 출발점과 같은 사분면이라면 바로 $2^{N-1} \times 2^{N-1}$ 크기의 문제가 됩니다.

Div1K. 스놉시티



- 도착점이 출발점과 다른 사분면이라면 출발점에서 중심의 사각형을 거쳐 도착점으로 가게 됩니다.

Div1L/Div2I. 뚜루루 뚜루

- 제출 횟수 : 40
- 맞은 참가자 수 : 17
- 정답률 : 55.000%
- 처음 맞은 참가자 : 김종범 (0:41)
- 출제자 : doju