Coder’s high 2016 Round 1 해법 설명 프레젠테이션

2016년 5월 30일

A. Mini Fantasy War

- 문제: https://www.acmicpc.net/problem/12790
- 맞은 사람 수 : 443
- 제출 횟수 : 1021
- 정답률: 43.976%
- 처음 맞은 팀: **“Andromeda Express (Feat. kriii)” (Astein, ainu7, kriii)** , 1분
- 출제자 : beingryu (류원하)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 단순 구현

A. Mini Fantasy War

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No		1		6	3	8	8	5	14	8	16	12	6	10	9	13	6	2	14	10	6	10	16	10	11	15	4	9	10	7
	Yes		1	6	13	22	20	27	23	28	30	16	19	12	10	9	16	12	11	8	4	10	2	7	3	8	2	4	2	3	1
14:30	No	11	11	8	7	6	2	4	6	10	5	6	6	4	7	3	5	3	4	6	5	3	7	2	2	7	6	4	5	5	
	Yes	7	3	1	1	3	5	1	3	2	1	3	4	3	2	4	3	1	1		1		4	1	4	3	2	2	2	4	
15:00	No	5	7		2		3	1	7	5	7	4	1	5		1	4	2	2	2	1	1	1	2	2	4	3	2	1	3	1
	Yes		2			1	1				1		1	1	1		2		2	2	2	1					1		2	1	1
15:30	No	4	1	3		3	2	3	2	2	2		2	2	3		1		4		1	1	3			1	1	3			1
	Yes	1			1				1	1	1	3		1	1		1	1			2			1	2				1	1	
16:00	No	2			3			1	1	1	2	1	1							1		1	1	1			1			1	
	Yes		1							1							1			1	1			1							
16:30	No				1	1					2			1	1			3	3	1		2		1		1				2	2
	Yes						1																			1					

A. Mini Fantasy War

시키는 대로 하면 됩니다.

기본 HP, MP, 공격력, 방어력 받고

```
int h, m, a, d; scanf("%d%d%d%d", &h, &m, &a, &d);
```

증감시킨 다음

```
int dh, dm, da, dd; scanf("%d%d%d%d", &dh, &dm, &da, &dd);
h += dh; m += dm; a += da; d += dd;
```

HP, MP는 1보다 작으면 1로 간주, 공격력은 0보다 작으면 0으로 간주

```
h = max(h, 1); m = max(m, 1); a = max(a, 0);
```

출력

```
printf("%d\n", h + 5 * m + 2 * a + 2 * d);
```

A. Mini Fantasy War

3문제 이상 푸신 팀들 중 한 번이라도 A를 틀리신 분들은 자리를 빙빙 돌면서 "나는 빡빡이다"를 20번 외치세요.

\- 류원하 (beingryu)

```
gets;$<.map{|l|z=l.split;p z[0..3].zip(z[4..7],[1,1,0,-9e9],[1,5,2,2]).map{|p,q,r,s|[p.to_i+q.to_i,r].max*s}.inject(:+)}
```

B. Starman

- 문제: https://www.acmicpc.net/problem/12791
- 맞은 사람 수 : 386
- 제출 횟수 : 917
- 정답률: 42.203%
- 처음 맞은 팀: **“홍석주와그의열렬한팬들로구성된엄청난팀” (suckzoo, jihoon, HYEA)** , 4분
- 출제자 : koosaga (구재현)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 단순 구현

B. Starman

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No								2		1		1	3	1	1	1	2	1	1	2	6	3	4	5	2	10	5	7	3	4
	Yes					2	3	1	3	3	3	3	3	9	5	13	9	6	13	10	9	5	2	8	5	10	7	7	6	11	8
14:30	No	2	3	5	4	5	4	6	4	4	4	5	2	8	2	4	4	6	4	7	7	3	3	7	4	4	7	3	2	2	6
	Yes	8	4	6	8	6	9	2	7	8	2	3	5	4	2	2	1	2	3	2	4	7	4	4	1	5	4	1	1		1
15:00	No	4	5	4	3	4	2	5	6	6	1	6	3	5	5	5	4	2	3	5	3		4	3	2	2	3	3		4	7
	Yes	3	2		3	1	2	5	3			2	4	2	1	1	1	3	3	2	2	3	1		1		3		1		3
15:30	No	1	4	5	4	3	1	4	2	3	2	2	3	7	1	2	2	1	2	2	2	5	2	5	3	6	2	2	2	4	4
	Yes	2		1	3	1		1	1			1	1	1			1	1	2			2	1	1	1	3	3		1	1	
16:00	No	2	2	3	2	2	1	2	3	6	2	3	1	3	2	3	1	6	2			1	2	3	3	3	5	5	5	5	6
	Yes		3	1	1	1			1	1						1		1		1	2		1	1				1		1	
16:30	No	3	2		1	2	4		4		1	2	2	1	1	3	3	1		3	2	1	3	2	2			3	1	1	5
	Yes		1			1			1		1									1	1		1		1		1				

B. Starman

수많은 풀이법이 있지만 출제자의 의도는 이렇다고 합니다.

예제 출력 2 복사

```
25
1967 DavidBowie
1969 SpaceOddity
1970 TheManWhoSoldTheWorld
1971 HunkyDory
1972 TheRiseAndFallOfZiggyStardustAndTheSpidersFromMars
1973 AladdinSane
1973 PinUps
1974 DiamondDogs
1975 YoungAmericans
1976 StationToStation
```

⋮

① 모든 앨범 목록이 나와 있는 예제 출력 2를 복사한다.

B. Starman

수많은 풀이법이 있지만 출제자의 의도는 이렇다고 합니다.

```
int q;

int main(){
    cin >> q;
    printf("pair<int, string> bowie[%d] = {", q);
    for(int i=0; i<q; i++){
        int x;
        string y;
        cin >> x >> y;
        printf("{%d, \"%s\"}", x, y.c_str());
        if(i != q-1) printf(",");
    }
    printf("};");
}
```

② 코드를 작성해서
주어진 자료를 배열 형태로 만든다.
(꼭 배열일 필요는 없고,
코드에 첨부 가능한 형태이면 됨)

B. Starman

수많은 풀이법이 있지만 출제자의 의도는 이렇다고 합니다.

```
for(int j=0; j<25; j++){
  if(bowie[j].first >= s
     && bowie[j].first <= e){
   v.push_back(bowie[j]);
  }
}
```

③ 배열이 나왔으니 시키는 대로 한다.
S, E를 입력받은 후, S <= year <= E인 모든 앨범의 목록을 출력하면 된다.

B. Starman

출제자가 강조하고 싶었던 부분은 코드를 출력하는 코드를 짤 수 있다는 것입니다.

이 방법 외에도

- 직접 따옴표를 찍거나,
- 정규표현식을 써서 따옴표를 찍거나,
- Python 같이 편리한 언어를 사용하거나(..)

하는 식으로도 문제를 해결할 수 있습니다.

B. Starman

B번은 multimap<int, string> 쓰라고 전해라~~

- 류원하 (beingryu)

C. 주작 주 주작

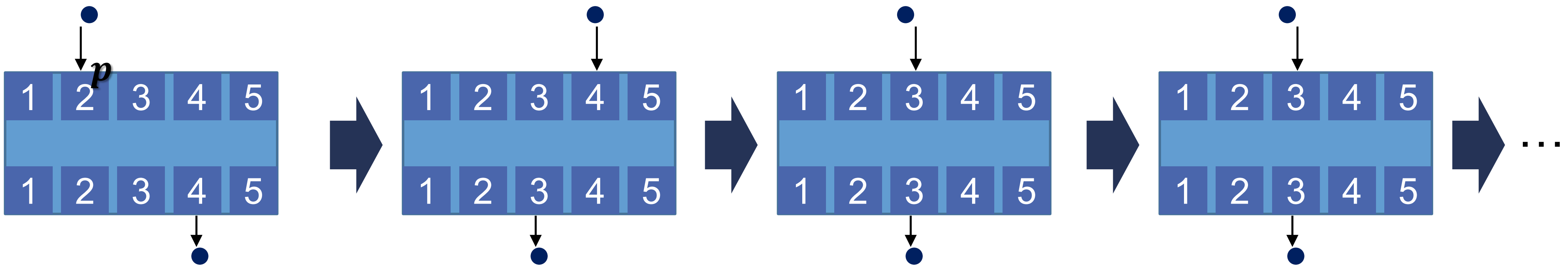
- 문제: https://www.acmicpc.net/problem/12792
- 맞은 사람 수 : 94
- 제출 횟수 : 2215
- 정답률: 4.244%
- 처음 맞은 팀: **“Anti-Q” (tonyjjw, dotorya, qo)** , 5분
- 출제자 : koosaga (구사과)
- 해설 작성자 : koosaga (구사과)
- 분류 : 관찰? 수학?

C. 주작 주 주작

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No								1	2	2	2	2	1		2	2		3	1	1	4	1	1	7	6	2	2	2	5	9
	Yes						1						2		1		1	1	1	1		1		1			2	2	1	1	
14:30	No	6	2	5	6	6	4	6	9	7	8	5	7	7	9	9	5	17	6	14	5	8	5	7	5	10	13	6	12	9	5
	Yes	1		1			1					1		1	1	1		1						1							1
15:00	No	10	4	6	17	10	14	16	17	12	10	7	13	10	9	13	14	10	19	7	17	13	14	10	13	14	18	14	16	11	13
	Yes		1	1		1		1	1	2			2		1		1			3				1		2	1	2	2	1	1
15:30	No	13	11	12	12	13	15	9	13	16	23	15	13	11	11	21	12	16	16	9	11	8	14	11	11	15	15	11	13	16	9
	Yes		1		1				2					1		1							1	1	2						
16:00	No	14	14	26	13	16	10	23	17	12	17	7	19	12	11	15	16	14	11	19	11	12	14	19	12	17	21	24	16	16	21
	Yes	1				1	1	1				1	1				2		2			1			1	1	2	2			
16:30	No	20	21	22	14	22	17	10	18	8	23	17	29	22	14	25	21	23	14	21	23	18	19	16	29	23	27	24	23	22	19
	Yes	2			1	2		2			1		1					1	1		1	1	1	1	1				1		

C. 주작 주 주작

추첨기의 어떠한 자리 p에 구슬을 떨어뜨린 후, 구슬이 나오는 구멍에 해당하는 위치에 다시 구슬을 떨어뜨리는 시행을 계속 반복한다고 상상해 봅시다.



C. 주작 주 주작

가능한 경우에는 두 가지가 있습니다.

1. 공은 다시는 p로 돌아오지 않습니다.
   - 2대 이상의 추첨기를 연결하여서는 절대 구사과의 주작을 방해할 수 없습니다.

2. 최소 T번 만에, 공이 p로 돌아옵니다.
   - 시행의 횟수가 T의 배수라면 구슬은 p에 떨어질 것이고, 그렇지 않다면 구슬은 p에 떨어지지 않을 것입니다.
   - 고로 구사과가 주작을 하려면 <u>연결한 추첨기의 개수는 T의 배수가 아니어야</u> 합니다.
     - 이는 $T = 1$일 때, 즉 어떤 i에 대해 $i = A[i]$일 때에는 답이 없음을 시사합니다.

C. 주작 주 주작

2. 최소 T번 만에, 공이 p로 돌아옵니다. (계속)

- 한편, 여기서 모든 p에 대해서 T는 N 이하임을 증명할 수 있습니다. 어떻게 할까요?
  - 공이 N번 안에 p에 돌아오지 않고, N번이 지난 이후에 p에 돌아왔다고 가정해 봅시다.
  - 이 때, 비둘기집의 원리에 의해 N번의 시행 동안 구슬이 방문한 지점의 번호를 순서대로 나열하면 중복된 수가 있을 것입니다.
  - 그렇다면, 두 중복된 수를 기준으로 수열이 반복될 것임을 알 수 있습니다. 구슬을 떨어뜨린 위치가 같으면 구슬이 나오는 위치도 같기 때문입니다.
  - 따라서 시행을 무한히 반복해도 이미 방문했던 지점들만 계속 방문하게 될 것이므로, 시행을 N번보다 많이 해서 p에 돌아온다는 가정에 모순됩니다.

C. 주작 주 주작

이제,

$T = 1$인 경우를 예외 처리한 후,
1000000 이상의 적당히 큰 소수를 아무거나 출력하면,

항상 답이 됨을 알 수 있습니다.

소수의 정의에 의해, 이 수는 N 이하의 어떠한 수의 배수도 되지 않을 것이기 때문입니다.

여담으로 이 문제는 제가 대학 간 선배한테서 들은 모 술 게임의 필승전략에서 착안하였습니다.

C. 주작 주 주작

변환 함수 f가 순열인지 아닌지 명시하지 않고
최대한 암시로 남기려고 노력했습니다.
까려면 저를 까세요 (..)

- 디스크립션 작성한 류원하 (beingryu)

D. 블록 게임

- 문제: https://www.acmicpc.net/problem/12793
- 맞은 사람 수 : 79
- 제출 횟수 : 545
- 정답률: 16.514%
- 처음 맞은 팀: **“Anti-Q” (tonyjjw, dotorya, qo)** , 27분
- 출제자 : etaehyun4 (이태현)
- 해설 작성자: etaehyun4 (이태현), beingryu (류원하)
- 분류 : 시뮬레이션

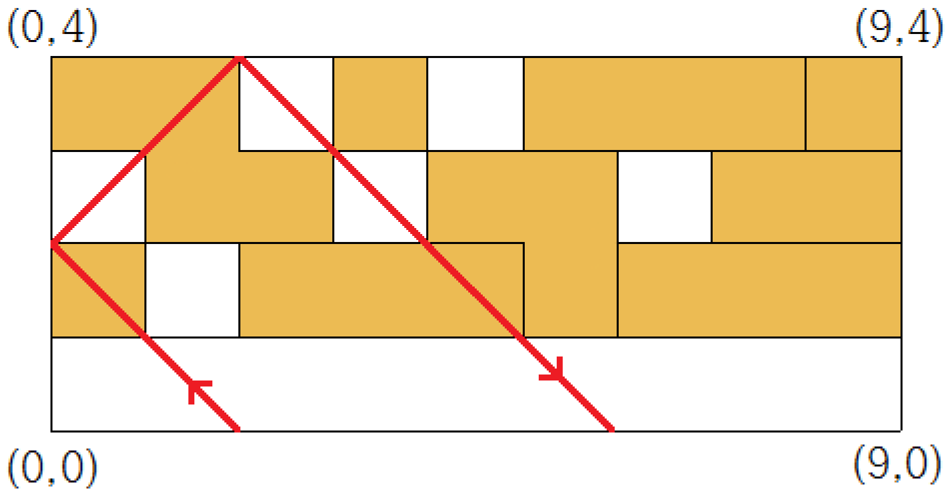
D. 블록 게임

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No																														
	Yes																												1		
14:30	No											1						1	1		2		1	1			2			1	3
	Yes			1		1	1	1	1		2														1	1					
15:00	No	2	2	2	1			1	2	1	3	2	1		4	2		2	4	4		1	1		4	3	3	1	2	1	3
	Yes	1	1			1	1			1	1			1		2	1	1	2	1			1	1	1	1			1		
15:30	No		1	3	4	3	1		2	1	1		4	1	4	2	1	3	5	1	1	2	3	1	2	4	2	3	3	4	8
	Yes	1	1		1	1							1	1			2				1	1	1		1	1	2	1		1	
16:00	No	4	4	4	3	5		4	4	2	6	3	1	2	3	7	5	1	3	3	6	1	6	5	5	1	3	1	4	2	3
	Yes	1			1		2		1	1	1		1	1			1	1	1			1	1	1	1	2					2
16:30	No	7	6	2	4	4	4	5	8	5	7	4	6	6	5	10	5	12	8	7	8	9	9	7	3	5	8	10	10	13	21
	Yes				1			1	1	2			1		1	2	1	1	1	2	3			3	2	1	1				

D. 블록 게임

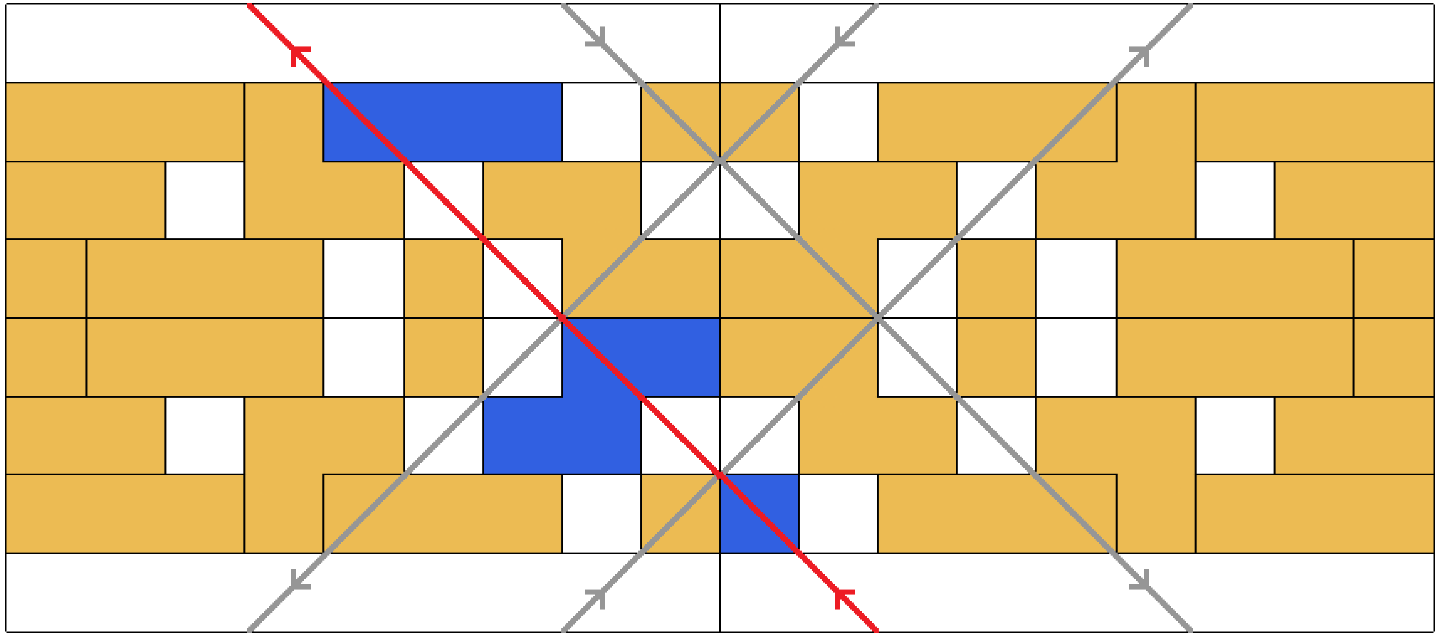
게임판이 크지 않기 때문에 직접 해 보면 됩니다.

~~파싱이 가장 귀찮은 문제~~



D. 블록 게임

다만 조금 더 간단하게 하려면, 반사를 구현하는 대신 게임판을 대칭시키면 됩니다.



E. 위대한 믹싱 가요제

- 문제: https://www.acmicpc.net/problem/12794
- 맞은 사람 수 : 2
- 제출 횟수 : 65
- 정답률: 3.077%
- 처음 맞은 팀: **“탑못쓰” (ainta, gs12117)** , 151분
- 출제자 : xhae (류현종)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 해싱, Suffix Array, Bitmask DP

E. 위대한 믹싱 가요제

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No																												
	Yes																												
14:30	No	1					1	1		1		1								1									
	Yes																												
15:00	No					1							1			2	1	1					1						
	Yes																												
15:30	No		1		1										2	3	2		1				1			1	1		1
	Yes																												
16:00	No			2	1									1					2		1				1				
	Yes																												
16:30	No		1	1			1		2	1	1		1	1	1				1		1	2	2	1	4	2	2	1	2
	Yes		1								1																		

E. 위대한 믹싱 가요제

이 문제의 해결 과정은 크게 두 가지 부분으로 나뉩니다.

1. 한 믹싱에 쓰일 노래들을 골랐을 때, <u>만족도(공통으로 존재하는 가장 긴 멜로디)</u>를 구하기
2. 1의 결과를 이용, 곡을 어떤 조합으로 믹싱할지 결정하여 <u>총 만족도 최대화</u>하기

저의 경우 1과 2를 통합하여 구현하였으나, 이 두 과정을 분리하여 구현하는 편이 코드를 볼 때 조금 더 깔끔한 것 같습니다.

E. 위대한 믹싱 가요제

① 만족도 구하기: Suffix Array version

문제에서 정의된 “만족도”는 몇 개의 문자열에서 공통으로 등장하는 가장 긴 부분문자열(substring)의 길이라고 생각할 수 있습니다. ‘Longest common substring’ 문제라고 하는 것 같습니다.

이는 문자열들이 정해져 있을 경우 suffix array를 이용하여, 대략 $n + \sum|S_i|$ 길이의 문자열의 suffix array를 만드는 데 드는 시간만 들여 공통 부분 문자열을 구할 수 있습니다. 검색해 보면 이런 문서가 있으니 참고하시기 바랍니다(~~설명하기 귀찮아서가 아닙니다~~)

물론 해싱과 같은 다른 방법도 있긴 합니다.

E. 위대한 믹싱 가요제

① 만족도 구하기: Suffix Array version

그런데 이 문제에서는 상황이 약간 다릅니다. 모든 가능한 믹싱에 대해서 만족도를 구할 수 있어야 하기 때문입니다. 하지만 이는 그리 큰 문제가 되지 않습니다.

가장 빨리 나온 곡 정하는 경우의 수

이후 m년간 나온 곡 중 $c-1$개의 곡을 정하는 경우의 수

서로 다른 가능한 믹싱의 수는 대략 $n \times \binom{m}{c-1} \leq 20n$가지입니다. 한 믹싱에는 최대 m+1개의 곡이 들어가고 $|S_i| \leq 500$이므로, 한 믹싱에서 고려할 총 문자열의 길이는 길어봐야 $500(m+1)$입니다.

그래서 모든 가능한 믹싱을 다 고려해보면서 앞의 방법으로 만족도를 구하면, 많아야 $20n \times 500(m+1) \leq 20 \cdot 500 \cdot 500 \cdot 7 = 35000000$회의 반복이면 충분합니다.

E. 위대한 믹싱 가요제

① 만족도 구하기: Hashing version

엄밀한 정의는 아니지만, 해싱이란 임의의 값을 우리가 좀 더 쉽게 접근할 수 있는 다른 형태로 대표하려는 시도를 말합니다. Ex)

```
long long hashed = 0;
const long long MOD = 1000000007;
for(char ch: melody) hashed = (hashed * 7 + (ch – ‘a’)) % MOD;
```

와 같은 형식으로 ‘abcdefg’로 이뤄져 있는 스트링 하나의 long long 값으로 나타낼 수 있죠.

E. 위대한 믹싱 가요제

① 만족도 구하기: Hashing version

다만 이와 같은 방법으로 스트링을 해싱하게 되면 서로 다른 두 개의 스트링이 같은 long long 값으로 표현되는 문제가 생길 수 있습니다. 이를 해시 충돌이라고 부르며, 정석적인 방법으로는 long long 값에 대하여 실제 스트링을 체인 형태로 보관하는 방법이 있으나 본 문제를 해결할 때에는 서로 다른 2개의 소수 MOD로 동시에 해싱을 진행였습니다.

두 해시가 동시에 충돌이 날 확률은 한 해시에 비해 기하급수적으로 작아지기 때문에 문제풀이를 위해서라면 이와 같은 테크닉으로도 많은 경우 정답으로 인정받을 수 있습니다.

이도 실패한다면 MOD값을 바꿔보는 방법과 체인을 구현하여 정석적으로 해결하는 방법이 있겠네요!

E. 위대한 믹싱 가요제

① 만족도 구하기: Hashing version

경우의 수 자체는 Suffix Array 버전과 크게 다르지 않습니다.

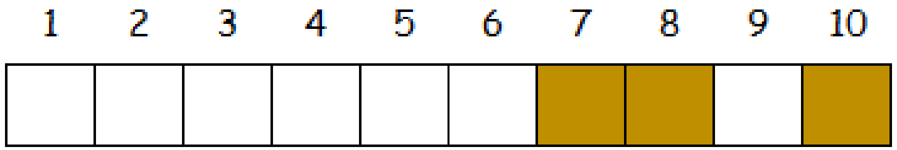
$20n$ 가지의 경우의 수에 대하여 조사를 하는데, Suffix Array와 다르게 해싱 자체는 LCS를 구하는 것에 최적화 되어 있지는 않기 때문에, LCS의 길이는 바이너리 서치를 통해서 확정지을 수 있다는 특성을 사용하여 Suffix Array 버전보다 $O(lgLEN)$이 더 붙어있는 형태에서 각종 최적화를 통하여 답을 구할 수 있습니다.

E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

동적 계획법을 이용합니다. 기본적으로 앞에서부터 차례대로 어떤 곡들을 믹싱할지를 결정해 나가는 구조입니다.

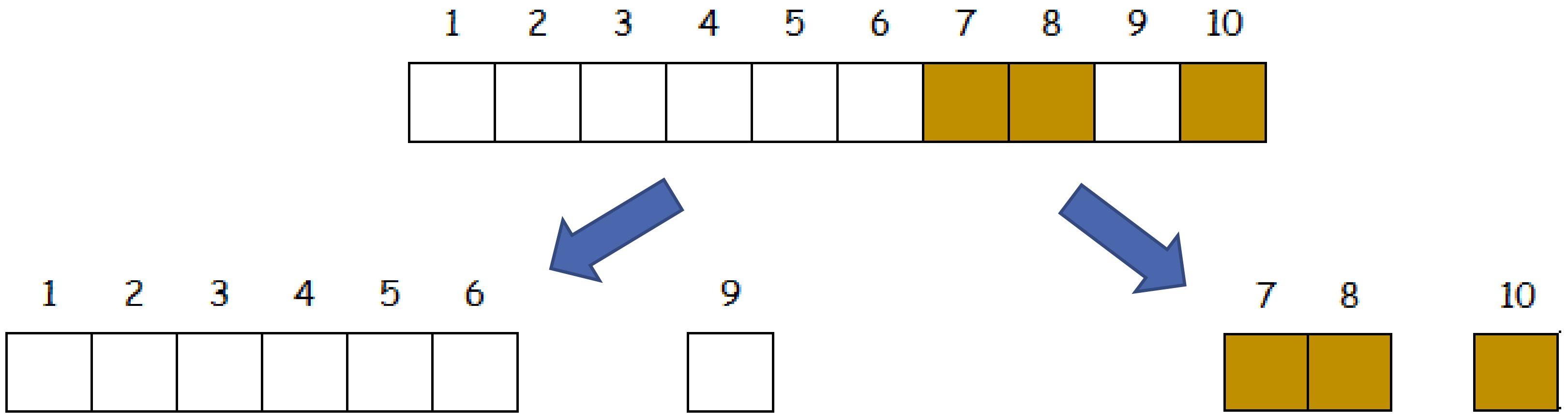
1, 2, 3, ⋯, i번 곡만 있는 상황에서, i를 포함하는 어떤 곡들을 믹싱하기로 결정했다고 합시다. 예를 들어 아래 그림은 1, 2, .., 10번 곡만 있는 상황에서 7, 8, 10번 곡들을 믹싱하기로 결정한 겁니다.



E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

7, 8, 10번 곡을 믹싱하기로 결정했다면, 한 곡은 최대 한 믹싱에만 쓰일 수 있으므로 그냥 저 곡들을 세상에서 지워버립니다. 지워버린 상태에서 만들 수 있는 가장 큰 만족도와 7, 8, 10번 곡의 만족도를 더하면, “1~10번 곡으로 만들 수 있는 가장 큰 만족도”의 후보 중 하나가 됩니다.



E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

이걸 기본 발상으로 하여, 아래와 같이 부분문제를 정의합니다.

$\boldsymbol{D(i,S)}$: $\mathbf{1, 2, \cdots,}\ \boldsymbol{i}$**번 곡**들 가운데
집합 $\boldsymbol{S}$에 해당하는 곡들만 <u>사용이 가능</u>할 때,
이 곡들을 믹싱하여 얻을 수 있는 최적의 만족도

관계식을 찾아봅니다. 일단 당연히 $i \notin S$라면,

$$D(i,S) = D(i-1,S)$$

입니다. i번 곡이 추가되었다 해서 달라지는 게 전혀 없기 때문입니다.

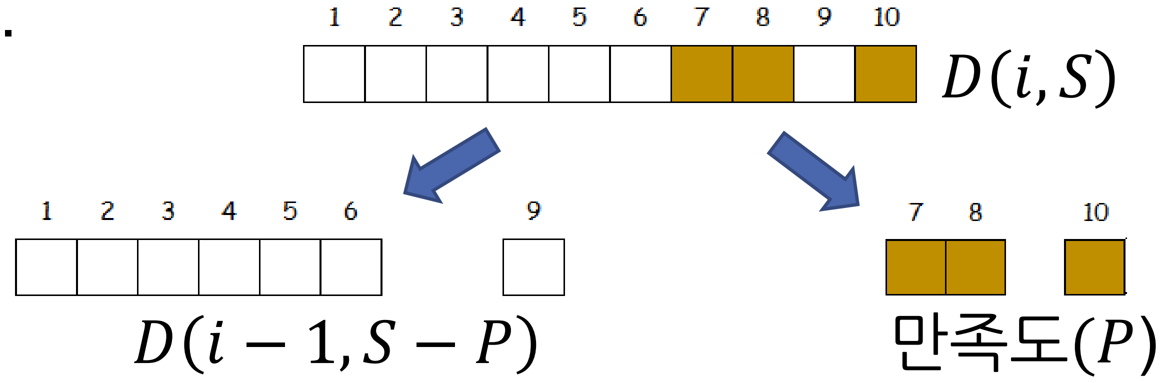
E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

이제 $i \in S$인 경우를 봅시다. i번 곡과 함께 믹싱될 곡들의 집합을 P ($P \subset S$)로 둡시다. 예를 들어 앞의 예시에서 $P = \{7,8,10\}$이 됩니다.

$$D(i,S) = D(i-1, S-P) + (P\text{의 만족도})$$

의 관계를 얻습니다. 이를 모든 가능한 P에 대해 다 계산해 보고 그 중 최댓값을 취하면 됩니다.



E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

또한, 사용 가능한 곡 수가 많아진다 해서 불리할 게 없으므로, S의 모든 부분집합 S'에 대해

$$D(i,S) \geq D(i,S')$$

의 관계를 반드시 만족해야 합니다. D를 해결하는 방법에 따라 이 관계를 만족하지 않는 경우가 있으니 반드시 유의해주셔야 합니다.

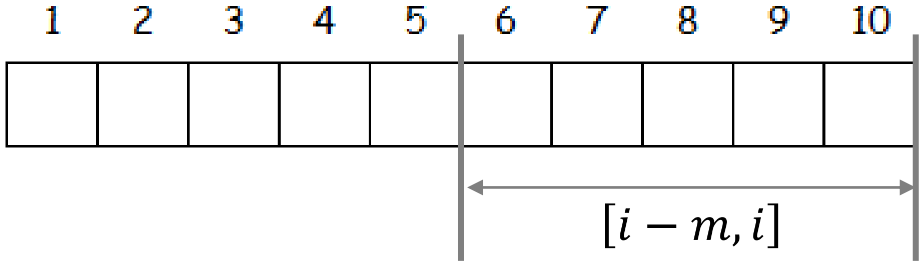
E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

부분문제의 정의에 의해, 우리가 구해야 할 답은

$$D(n, \{1,2,3,\cdots,n\})$$

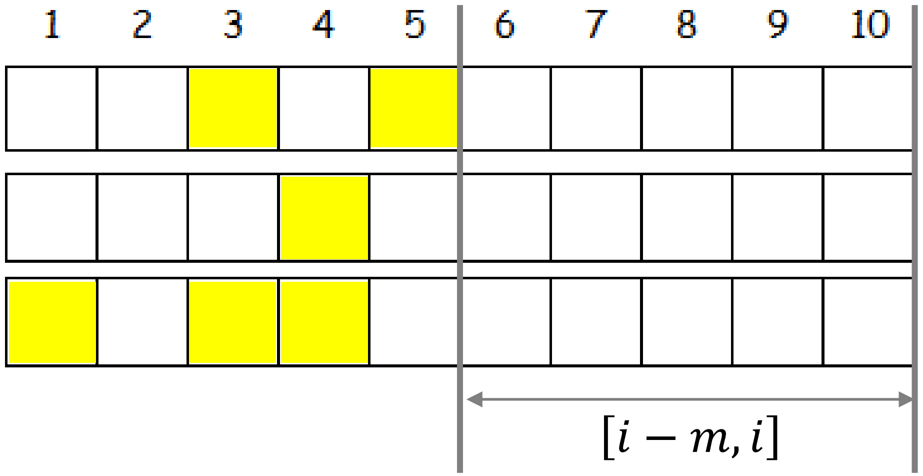
임을 쉽게 알 수 있습니다. 하지만 고려할 상태의 수가 $n \times 2^n$이고, 각 상태마다 $\binom{m}{c-1}$번의 참조가 필요하므로 너무 느립니다. 상태의 수를 좀 많이 줄여야 할 것 같습니다.



E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

i번 곡과의 연도 차이가 m보다 큰 곡들은 믹싱에 쓰였든 안 쓰였든 i번 곡의 믹싱에는 영향을 미치지 않습니다. 그러니 그냥 $1,2,\cdots,i-m-1$번 곡은 모두 사용 가능하다 (즉 $\{1,2,\cdots,i-m-1\} \subset S$)고 생각해도 됩니다.



E. 위대한 믹싱 가요제

② 총 만족도 최대화하기

이제 각 i마다 가능한 S의 수가 2^{m+1}로 대폭 줄어들어, 부분문제 D를 해결하기 위해서 대략 $n \times 2^{m+1} \times \binom{m}{c-1}$회의 반복만 필요하게 됩니다. 대신 D의 관계식을 이용하기 위해 약간 자잘한 처리가 필요할 것입니다.

이 발상은 많은 비트마스크 DP 문제에서 사용되는 것이므로 기억해 두는 것이 좋다고 생각합니다.

F. 반평면 땅따먹기

- 문제: https://www.acmicpc.net/problem/12795
- 맞은 사람 수 : 4
- 제출 횟수 : 540
- 정답률: 0.741%
- 처음 맞은 팀: **“탑못쓰” (ainta, gs12117)** , 80분
- 출제자 : koosaga (구재현)
- 해설 작성자 : koosaga (구재현)
- 분류 : 자료구조

F. 반평면 땅따먹기

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14
14:00	No						1								1	
	Yes															
14:30	No	2	2		3	1	2			2	3	2	1	3	1	1
	Yes															
15:00	No	3	2	2	6	4	2	2	4	3	6	4	3	2	4	1
	Yes															
15:30	No	2	3	4	1	3	1	4	2	3	1	2	3	3	5	1
	Yes															
16:00	No	1	2	2	5	4	3	4	3	4	3	3	4	6	7	2
	Yes			1												
16:30	No	4	4	10	7	5	4	6	4	6	7	4	4	6	2	3
	Yes															

		+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No										1	1	3	1	1	1
	Yes															
14:30	No	1		3	1	4		6	1	4		3	3	1	5	5
	Yes															
15:00	No	1	1	1	1	3	3	4	4	1		1	2	4	2	5
	Yes						1									
15:30	No	7	5	3	5	4	4	2	3	1	3	1	5	1	4	7
	Yes											1				
16:00	No	4	4	4	7	4	4	1	4	4	1	3	4	8	3	3
	Yes														1	
16:30	No	3	6	2	3	1	10	5	8	5	6	8	15	12	8	13
	Yes															

F. 반평면 땅따먹기

삽입 질의가 없이 직선 집합이 고정되어 있을 때, 이 문제를 푸는 방법에 대해서 고민해 봅시다.

이 때는 주어진 직선을 기울기 순으로 정렬한 후,
컨벡스 헐 트릭(Convex Hull Trick) 이라는 자료구조를 선형 시간에 만듦으로써 문제를 풀 수 있습니다.

최댓값 쿼리는, 이진 탐색으로 $O(\lg Q)$에 구현 가능합니다.

http://wcipeg.com/wiki/Convex_hull_trick

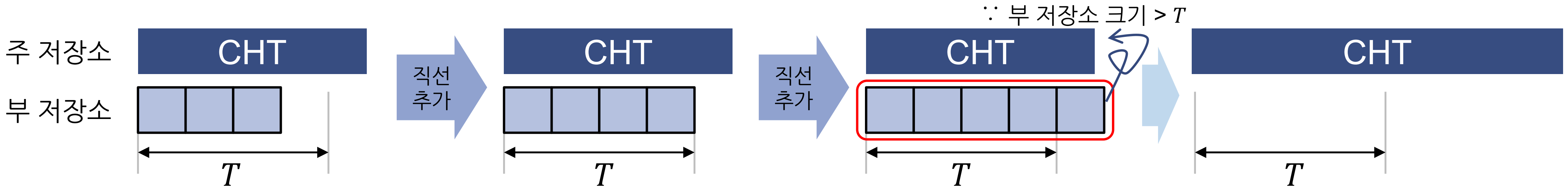
http://codedoc.tistory.com/11

F. 반평면 땅따먹기

삽입 질의가 들어와도, 약간의 아이디어를 동반하면 크게 달라지는 것은 없습니다.

컨벡스 헐 트릭을 사용하는 “주 저장소” 와, 선형 탐색으로 돌리는 “부 저장소” 를 만듭니다. 두 저장소 모두 기울기 순으로 정렬되어 있습니다.

부 저장소의 크기 한계 T를 정해놓읍시다. 만약에 부 저장소의 크기가 T를 넘어가면 그때그때 주 저장소로 선분을 삽입한 후 다시 컨벡스 헐 트릭을 만듭니다. 이는 Merge Sort와 비슷하게 $O(Q)$에 가능합니다.



F. 반평면 땅따먹기

- 직선 삽입은 질의당 $O(T)$
  - 삽입 정렬과 유사한 방법으로 직선을 추가하여, 기울기가 정렬된 상태를 유지하기 때문입니다.
- 최댓값 계산은 질의당 $O(\lg Q + T)$,
  - 주 저장소에서 이분탐색, 부 저장소에서 선형 탐색
- 질의와는 별개로 컨벡스 헐 트릭 재생성에 $O(Q^2/T)$ 만큼의 시간이 사용됩니다.
  - $O(Q/T)$번 마다 주 저장소를 다시 만들고,
  - 한 번 재생성할 때마다 $O(Q)$의 시간을 사용하기 때문입니다.

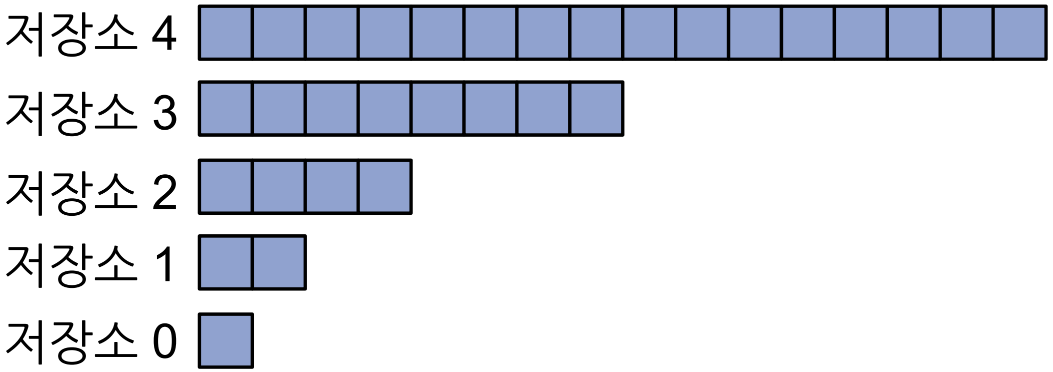
따라서, 총 시간 복잡도는 $O(QT + Q^2/T)$입니다.

$T = O(\sqrt{Q})$로 놓으면, $O(Q^{1.5})$에 문제를 해결하고 정답 처리를 받을 수 있습니다.
의도한 풀이는 이것입니다.

F. 반평면 땅따먹기

모범 풀이는 $\mathrm{O}(Q \lg^2 Q)$에 작동하며, “부 저장소” 아이디어의 일반화입니다.

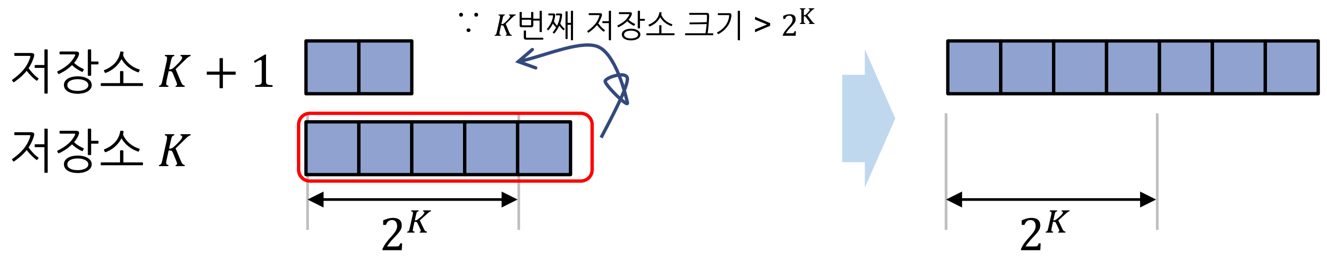
이번에는 $\lg Q$개의 “저장소” 를 만들어서, 각각을 컨벡스 헐 트릭으로 관리합니다. 각각의 크기 한계는, $2^0, 2^1, \cdots, 2^{17}, 2^{18}$ 과 같이 2의 거듭제곱 꼴로 정해집니다.



F. 반평면 땅따먹기

선분 하나가 들어왔을 경우,

- 먼저 2^0의 크기 한계를 가진 0번째 저장소에 선분을 넣습니다.
- 0번째 저장소는, 크기가 2^0을 초과할 경우, 모든 원소를 1번째 저장소에 합치고, 자신의 저장 원소를 삭제합니다.
- 마찬가지로, K번째 저장소는, 크기가 2^K를 초과할 경우, 모든 원소를 $(K+1)$번째 저장소에 보내고, 자신의 저장 원소를 모두 삭제합니다.
- 두 저장소를 합치는 방법은 역시 Merge Sort의 요령으로 해결 가능합니다.



F. 반평면 땅따먹기

시간 복잡도를 분석해 봅시다.

- 각각의 저장소는 $O(Q/2^K)$ 번 초기화 되고, $\mathrm{O}(2^K)$ 의 초기화 시간을 요구합니다. 즉, 총 $\mathbf{O}(\boldsymbol{Q})$의 시간을 소모합니다.
- 저장소가 $\mathrm{O}(\lg Q)$개 있으니, 모든 저장소를 관리하는 데에는 총 $\mathbf{O}(\boldsymbol{Q}\,\mathbf{lg}\,\boldsymbol{Q})$의 시간이 사용됩니다.
- 질의를 할 때에는, 모든 $\lg Q$개의 저장소에 이진 탐색을 돌리니 질의당 $\mathrm{O}(\lg^2 Q)$의 시간, 다 합치면 $\mathbf{O}\left(\boldsymbol{Q}\,\mathbf{lg}^2\,\boldsymbol{Q}\right)$이 사용됩니다.
- 따라서, 총 시간 복잡도는 $\mathbf{O}\left(\boldsymbol{Q}\,\mathbf{lg}^2\,\boldsymbol{Q}\right)$입니다.

이러한 "저장소" 개념은 "Transforming static data structures to dynamic structures", James B. Saxe 논문의 Section 3.1에 자세히 설명되어 있습니다.

F. 반평면 땅따먹기

여담으로, Convex Hull Trick이라는 자료구조를 응용해서, std::set 과 같은 Balanced BST에 선분과 교점을 넣고, 질의를 적절하게 처리하는 알고리즘 역시 존재합니다. 이 때 시간 복잡도는 $\boldsymbol{O(Q \lg Q)}$입니다.

상당히 생각하기 쉬운 방법이지만, 코딩이 꽤 복잡해서 추천하지는 않습니다.

그 외, 선분의 삽입 시간 기준으로 segment tree를 만들어서, 오프라인으로 $\boldsymbol{O\left(Q \lg^2 Q\right)}$에 문제를 푸는 방법 등이 존재합니다.

G. 나의 행렬곱셈 답사기

- 문제: https://www.acmicpc.net/problem/12796
- 맞은 사람 수 : 114
- 제출 횟수 : 204
- 정답률: 55.882%
- 처음 맞은 팀: **“홍석주와그의열렬한팬들로구성된엄청난팀” (suckzoo, jihoon, HYE)**, 19분
- 출제자 : tncks0121 (박수찬)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 국어(?)

G. 나의 행렬곱셈 답사기

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No																		1	1											1
	Yes																				1					2			1		
14:30	No	1						1				1	2		1			1		1				1	1	1		1		1	1
	Yes			2		3			1				2	1	1			1		1	1	2	1		1			1	1	2	1
15:00	No				1					1		1										1	1		1		1	2	1	1	
	Yes		2	1	3	1			3	2			2		2				1				1		2	1		3		1	1
15:30	No			2	1	2	3	1	1		2	1					3				2					1		2	1		
	Yes				3	1		1	2		2	1	3	1	1	2	2	3	1		1		2		1			1			1
16:00	No				1				1	2			2			1	1	1	3	5							1		1		
	Yes			2	1		1					1	1	1			1		2	1	3	1		1	1	1	1			1	1
16:30	No		1			1			1		2	1			3	2					1		3	1		1			1		3
	Yes			1	1						1	3						1				2		1		2					

G. 나의 행렬곱셈 답사기

모든 것은 문제 안에 있습니다.

$\mathbf{A}$가 2×4 행렬이고, $\mathbf{B}$가 4×3 행렬, $\mathbf{C}$가 3×5 행렬이라고 하자. 행렬의 곱 $\mathbf{ABC}$를 계산하기 위해 필요한 정수 곱셈의 수를 분석해 보면 아래와 같다.

- $(\mathbf{AB})\mathbf{C}$와 같이 계산한다면
  - 2×4 행렬 $\mathbf{A}$와 4×3 행렬 $\mathbf{B}$를 곱할 때 $2 \times 4 \times 3 = 24$회의 정수 곱셈이 필요하며, 그 결과 2×3 행렬이 만들어진다.
  - 2×3 행렬 $\mathbf{AB}$와 3×5 행렬 $\mathbf{C}$를 곱할 때 $2 \times 3 \times 5 = 30$회의 정수 곱셈이 필요하며, 그 결과 2×5 행렬이 만들어진다.
  - 따라서 총 $24 + 30 = 54$회의 정수 곱셈이 필요함을 알 수 있다.
- $\mathbf{A}(\mathbf{BC})$와 같이 계산한다면
  - 4×3 행렬 $\mathbf{B}$와 3×5 행렬 $\mathbf{C}$를 곱할 때 $4 \times 3 \times 5 = 60$회의 정수 곱셈이 필요하며, 그 결과 4×5 행렬이 만들어진다.
  - 2×4 행렬 $\mathbf{A}$와 4×5 행렬 $\mathbf{BC}$를 곱할 때 $2 \times 4 \times 5 = 40$회의 정수 곱셈이 필요하며, 그 결과 2×5 행렬이 만들어진다.
  - 따라서 총 $60 + 40 = 100$회의 정수 곱셈이 필요함을 알 수 있다.

G. 나의 행렬곱셈 답사기

모든 것은 문제 안에 있습니다.

$\mathbf{A}$가 $\mathbf{1 \times 1}$ 행렬이고, $\mathbf{B}$가 $\mathbf{1 \times 1}$ 행렬, $\mathbf{C}$가 $\mathbf{1 \times p}$ 행렬이라고 하자. 행렬의 곱 $\mathbf{ABC}$를 계산하기 위해 필요한 정수 곱셈의 수를 분석해 보면 아래와 같다.

- $(\mathbf{AB})\mathbf{C}$와 같이 계산한다면
  - $\mathbf{1 \times 1}$ 행렬 $\mathbf{A}$와 $\mathbf{1 \times 1}$ 행렬 $\mathbf{B}$를 곱할 때 $\mathbf{1 \times 1 \times 1 = 1}$ 회의 정수 곱셈이 필요하며, 그 결과 $\mathbf{1 \times 1}$ 행렬이 만들어진다.
  - $\mathbf{1 \times 1}$ 행렬 $\mathbf{AB}$와 $\mathbf{1 \times p}$ 행렬 $\mathbf{C}$를 곱할 때 $\mathbf{1 \times 1 \times p = p}$ 회의 정수 곱셈이 필요하며, 그 결과 $\mathbf{1 \times p}$ 행렬이 만들어진다.
  - 따라서 총 $\mathbf{p + 1}$ 회의 정수 곱셈이 필요함을 알 수 있다.
- $\mathbf{A}(\mathbf{BC})$와 같이 계산한다면
  - $\mathbf{1 \times 1}$ 행렬 $\mathbf{B}$와 $\mathbf{1 \times p}$ 행렬 $\mathbf{C}$를 곱할 때 $\mathbf{1 \times 1 \times p = p}$ 회의 정수 곱셈이 필요하며, 그 결과 $\mathbf{1 \times p}$ 행렬이 만들어진다.
  - $\mathbf{1 \times 1}$ 행렬 $\mathbf{A}$와 $\mathbf{1 \times p}$ 행렬 $\mathbf{BC}$를 곱할 때 $\mathbf{1 \times 1 \times p = p}$ 회의 정수 곱셈이 필요하며, 그 결과 $\mathbf{1 \times p}$ 행렬이 만들어진다.
  - 따라서 총 $\mathbf{p + p}$ 회의 정수 곱셈이 필요함을 알 수 있다.

G. 나의 행렬곱셈 답사기

모든 것은 문제 안에 있습니다.

$\mathbf{A}$가 1×1 행렬이고, $\mathbf{B}$가 1×1 행렬, $\mathbf{C}$가 $1 \times p$ 행렬이라고 하자. 행렬의 곱 $\mathbf{ABC}$를 계산하기 위해 필요한 정수 곱셈의 수를 분석해 보면 아래와 같다.

- $(\mathbf{AB})\mathbf{C}$와 같이 계산한다면
  - 1×1 행렬 $\mathbf{A}$와 1×1 행렬 $\mathbf{B}$를 곱할 때 $1 \times 1 \times 1 = 1$ 회의 정수 곱셈이 필요하며, 그 결과 1×1 행렬이 만들어진다.
  - 1×1 행렬 $\mathbf{AB}$와 $1 \times p$ 행렬 $\mathbf{C}$를 곱할 때 $1 \times 1 \times p = p$ 회의 정수 곱셈이 필요하며, 그 결과 $1 \times p$ 행렬이 만들어진다.
  - 따라서 총 $p + 1$ 회의 정수 곱셈이 필요함을 알 수 있다.
- $\mathbf{A}(\mathbf{BC})$와 같이 계산한다면
  - 1×1 행렬 $\mathbf{B}$와 $1 \times p$ 행렬 $\mathbf{C}$를 곱할 때 $1 \times 1 \times p = p$ 회의 정수 곱셈이 필요하며, 그 결과 $1 \times p$ 행렬이 만들어진다.
  - 1×1 행렬 $\mathbf{A}$와 $1 \times p$ 행렬 $\mathbf{BC}$를 곱할 때 $1 \times 1 \times p = p$ 회의 정수 곱셈이 필요하며, 그 결과 $1 \times p$ 행렬이 만들어진다.
  - 따라서 총 $p + p$ 회의 정수 곱셈이 필요함을 알 수 있다.

$(p+p)-(p+1)=p-1$회의 차이가 있네요. $p=K+1$로 잡으면 원하는 결과가 나옵니다.

G. 나의 행렬곱셈 답사기

따라서..

$$3$$
$$1 \;\; 1 \;\; 1 \;\; K+1$$

를 출력하면 됩니다. 다른 다양한 풀이도 많습니다.

H. 연금술

- 문제: https://www.acmicpc.net/problem/12797
- 맞은 사람 수 : 5
- 제출 횟수 : 261
- 정답률: 4.981%
- 처음 맞은 팀: **“Andromeda Express (Feat. kriii)” (Astein, ainu7, kriii)** , 83분
- 출제자 : cubelover (윤지학)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 행렬(?), 수학

H. 연금술

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No																										1		1		
	Yes																														
14:30	No							1								1		1			1	1			1	1		1			
	Yes																														
15:00	No	1	2	1		2		1		1		1	1	1	1	1	3	1	1		1		2	2	2		1		1		1
	Yes																								1					1	
15:30	No		1	1	2	3	1			2	2			1	2	1	2	1	2	1	3	2	2	1	1				1		2
	Yes		4	2					1																						
16:00	No			1	2	1	1		2	1	1	2	2	2	4	3	1	4		1	1	3	2	3	3	3	3	4		4	3
	Yes																									1					
16:30	No	4	5		6			5	2	2	2	2	1	3	2	6	3	7	6	3	6	6	10	8	5	5	5	2	6	5	2
	Yes					1				1																				1	

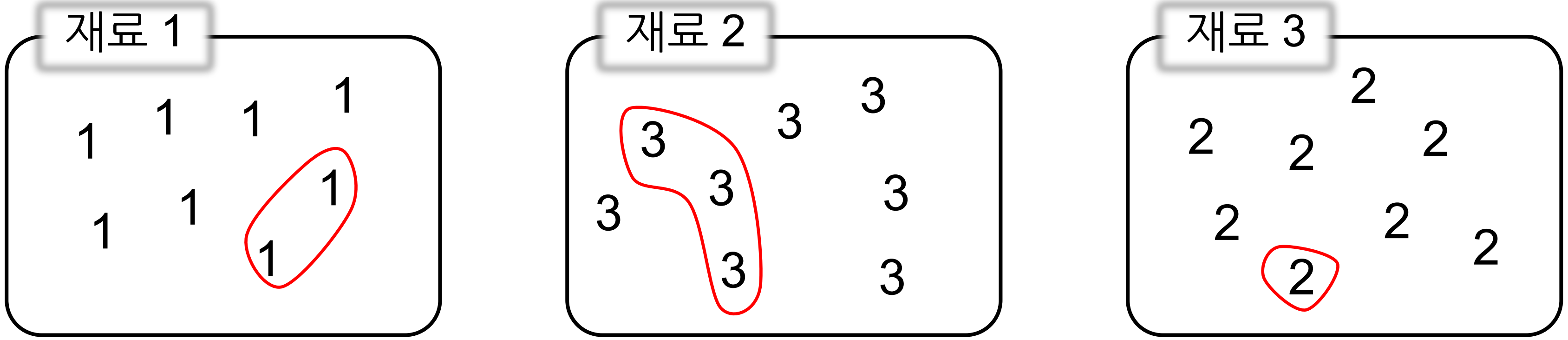
H. 연금술

이 풀이는 출제자의 풀이는 아닙니다. 출제자는 행렬의 대각화를 사용했다고 합니다만 이 풀이를 작성하는 사람은 그게 뭔지 몰라 설명을 못 합니다ㅠㅠ.. 그러니 제 풀이를 설명하겠습니다.

주어진 문제는 아래와 같이 품질이 적당한 자연수인 재료들이 무한히 있을 때, 이 중 n개를 택하는 것입니다.

H. 연금술

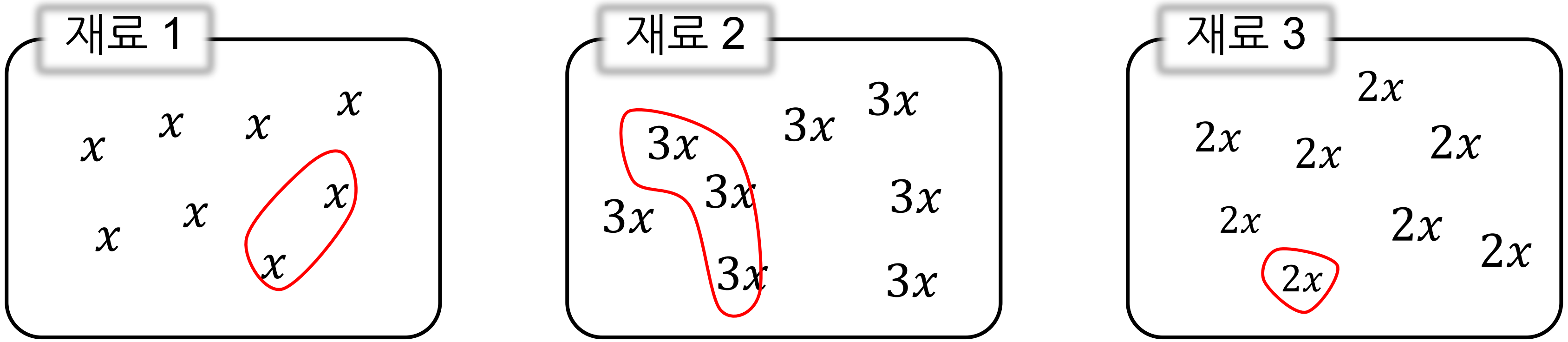
이렇게 말이죠..



$$\textbf{품질} = 1 \times 1 \times 3 \times 3 \times 3 \times 2 = 54$$

H. 연금술

재료를 몇 개 선택했는지 편하게 알기 위해 재료의 품질에다가 x를 덧붙여 봅니다. 그럼 완성품의 품질에는 x^n이 덧붙여질 것입니다.



$$\textbf{품질} = \boldsymbol{x \times x \times 3x \times 3x \times 3x \times 2x = 54x^6}$$

H. 연금술

품질이 ax인 재료를 k개 선택하면 완성품의 품질에 $(ax)^k$만큼 기여하고, 같은 재료끼리는 구별이 안 되므로, 각 상자를 다항식으로 바꿔서 “어떤 재료를 k개 택한다”는 것을 “항 $(ax)^k$를 택한다”는 것으로 생각해 봅시다.

재료 1

$$1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + \cdots$$

재료 2

$$1 + 3x + (3x)^2 + (3x)^3 + (3x)^4 + (3x)^5 + \cdots$$

재료 3

$$1 + 2x + (2x)^2 + (2x)^3 + (2x)^4 + (2x)^5 + \cdots$$

$$\textbf{품질} = x^2 \times (3x)^3 \times 2x = 54x^6$$

H. 연금술

재료를 정확히 n개 선택했다면 x는 정확히 n번 곱해졌을 겁니다. 가능한 모든 경우에 대한 품질의 합을 구하고자 하므로, 결국 구해야 할 것은 다항식

$$
\begin{gathered}
(1 + a_1x + (a_1x)^2 + (a_1x)^3 + \cdots) \\
\times (1 + a_2x + (a_2x)^2 + (a_2x)^3 + \cdots) \times \cdots \\
\times (1 + a_mx + (a_mx)^2 + (a_mx)^3 + \cdots)
\end{gathered}
$$

에서 x^n의 계수임을 알 수 있습니다.

H. 연금술

그런데

$$1 + r + r^2 + r^3 + \cdots = \sum_{i=0}^{\infty} r^i = \frac{1}{1-r}$$

임을 이용하여, 앞의 식을

$$\frac{1}{1-a_1x} \times \frac{1}{1-a_2x} \times \cdots \times \frac{1}{1-a_mx}$$

로 바꿀 수 있습니다. 보기 좋게 바꾸긴 했지만, 그래도 계수를 찾는 건 어려워 보입니다.

H. 연금술

그런데(또?)

$$\frac{1}{1-a_1x} \times \frac{1}{1-a_2x} \times \cdots \times \frac{1}{1-a_mx}$$

는 적당한 수 $q_1, q_2, \cdots, q_m$에 대해

$$\frac{q_1}{1-a_1x} + \frac{q_2}{1-a_2x} + \cdots + \frac{q_m}{1-a_mx}$$

와 같이 나타낼 수 있습니다. 부분분수 분해를 한 것입니다. 그럼 $q_1, q_2, \cdots, q_m$은 어떻게 구할까요? 수능계(?)에서 *헤비사이드 부분분수 분해법*이라고 불리는 것을 사용합니다. 어떻게 하는 거냐면..

H. 연금술

예를 들어 q_1을 구하고 싶다고 합시다. 원래 식

$$\frac{1}{1-a_1x}\times\frac{1}{1-a_2x}\times\cdots\times\frac{1}{1-a_mx}=\frac{q_1}{1-a_1x}+\frac{q_2}{1-a_2x}+\cdots+\frac{q_m}{1-a_mx}$$

의 양변에 $1-a_1x$을 곱하여,

$$\frac{1}{1-a_2x}\times\cdots\times\frac{1}{1-a_mx}=q_1+(1-a_1x)\left(\frac{q_2}{1-a_2x}+\cdots+\frac{q_m}{1-a_mx}\right)$$

로 나타냅시다. 그래놓고 $x=1/a_1$을 대입(?!)하면

$$q_1=\frac{1}{1-a_2/a_1}\times\frac{1}{1-a_3/a_1}\times\cdots\times\frac{1}{1-a_m/a_1}$$

을 얻게 됩니다.

H. 연금술

양변을 0으로 나눠 놓고 0을 대입하는 듯한 이 이상한 방법이 성립한다는 것은 함수의 극한을 이용해서 증명할 수 있습니다.

모듈러 상에서 역원을 구하는 데에 $O(\log MOD)$가 든다고 치면, q_i를 구하는 데에는 $O(m + \log MOD)$의 시간이 필요합니다. 그 이유를 대강 설명하자면,

$$\frac{1}{1 - a_2/a_1} \times \frac{1}{1 - a_3/a_1} \times \cdots \times \frac{1}{1 - a_m/a_1}$$

의 값을 구하기 전 $1/a_1$을 미리 계산해 놓은 후, 이를 이용해 분모의 곱을 구해놓고, 맨 나중에 역원을 취하면 되기 때문입니다.

이런 방식으로 $O(m^2)$의 시간으로 $q_1, q_2, \cdots, q_m$을 구할 수 있습니다.

H. 연금술

그럼 이제 어떻게 할까요? 이제 원래대로 돌아가서,

$$\frac{q_1}{1-a_1x}+\frac{q_2}{1-a_2x}+\cdots+\frac{q_m}{1-a_mx}$$

는 $\frac{1}{1-r}=1+r+r^2+r^3+\cdots=\sum_{i=0}^{\infty}r^i$ 에 의해(..)

$$\begin{aligned}&q_1(1+a_1x+(a_1x)^2+(a_1x)^3+\cdots)\\&+q_2(1+a_2x+(a_2x)^2+(a_2x)^3+\cdots)+\cdots\\&+q_m(1+a_mx+(a_mx)^2+(a_mx)^3+\cdots)\end{aligned}$$

가 됩니다. (..) 따라서 x^n의 계수는

$$q_1\cdot(a_1)^n+q_2\cdot(a_2)^n+\cdots+q_m\cdot(a_m)^n$$

입니다..

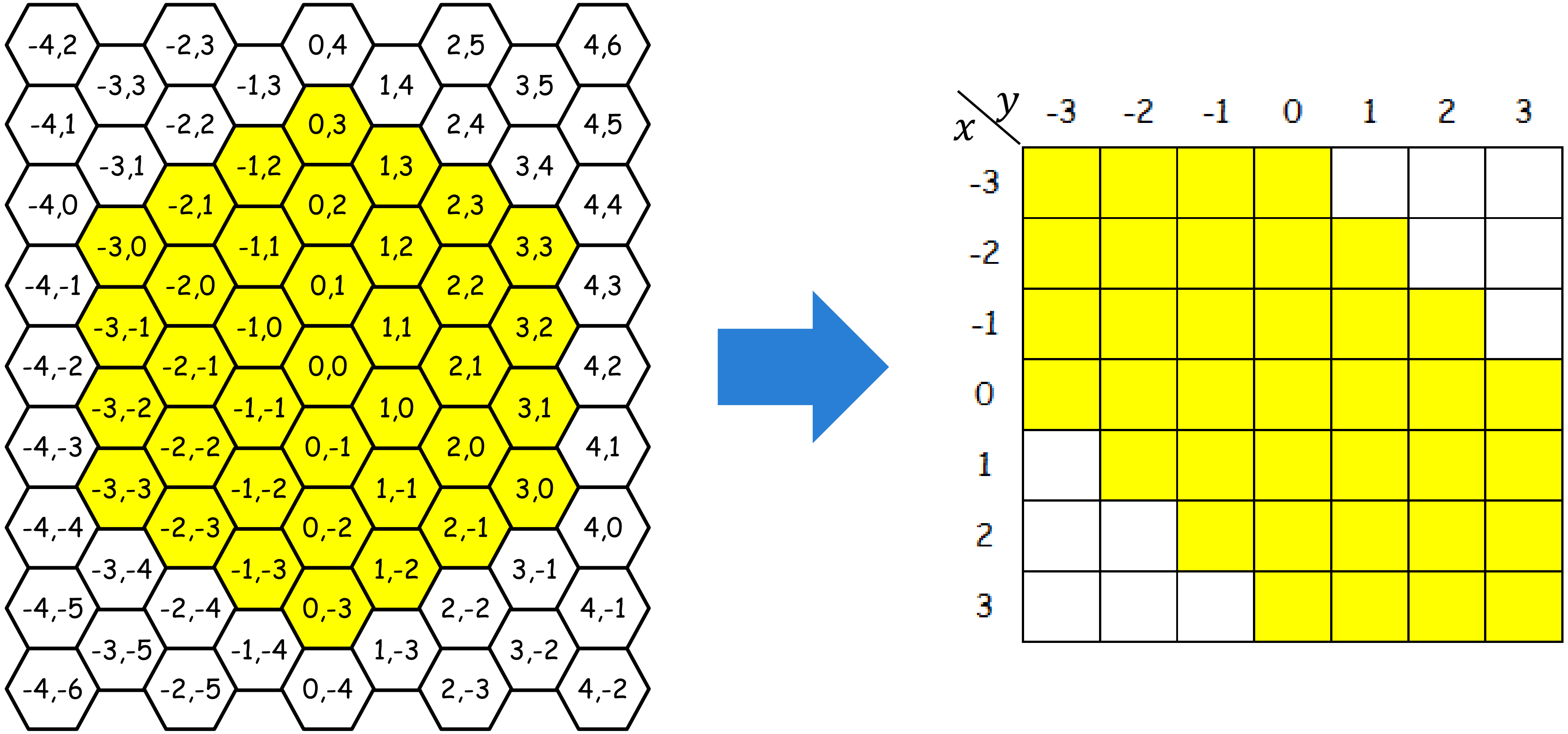
I. 게나디는 머리가 좋습니다

- 문제: https://www.acmicpc.net/problem/12798
- 맞은 사람 수 : 0
- 제출 횟수 : 274
- 정답률: 0%
- 처음 맞은 팀: N/A
- 출제자 : tncks0121 (박수찬)
- 해설 작성자 : tncks0121 (박수찬)
- 분류 : 자료구조

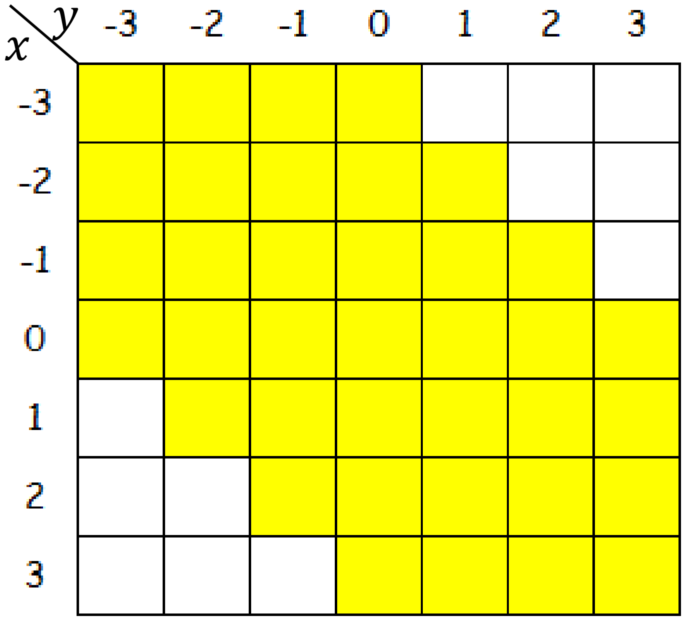
I. 게나디는 머리가 좋습니다

		+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 07	+ 08	+ 09	+ 10	+ 11	+ 12	+ 13	+ 14	+ 15	+ 16	+ 17	+ 18	+ 19	+ 20	+ 21	+ 22	+ 23	+ 24	+ 25	+ 26	+ 27	+ 28	+ 29
14:00	No																														
	Yes																														
14:30	No							1	1						1	1	2			1	1	3	1	1		1		2	1		
	Yes																														
15:00	No	2	2		1	1		3	1	1			1	1		1	1	1		1	1	2	2	2	2	4	2	3	1		
	Yes																														
15:30	No		1		2	2		3	1	1	1	2	2	3	1	1	2		5	5	3	5		2	1		4	1		2	1
	Yes																														
16:00	No		1		2	4	2	2			1	1	4	1	2	5	4	1	4	2	6	5	6	5	6	4	1		2	2	4
	Yes																														
16:30	No	5	2	3	4			3	4	3	2	4	1	4	4	1	3	1	3	3	5	2	2	3	3	4	4	4	2	4	10
	Yes																														

I. 게나디는 머리가 좋습니다



I. 게나디는 머리가 좋습니다



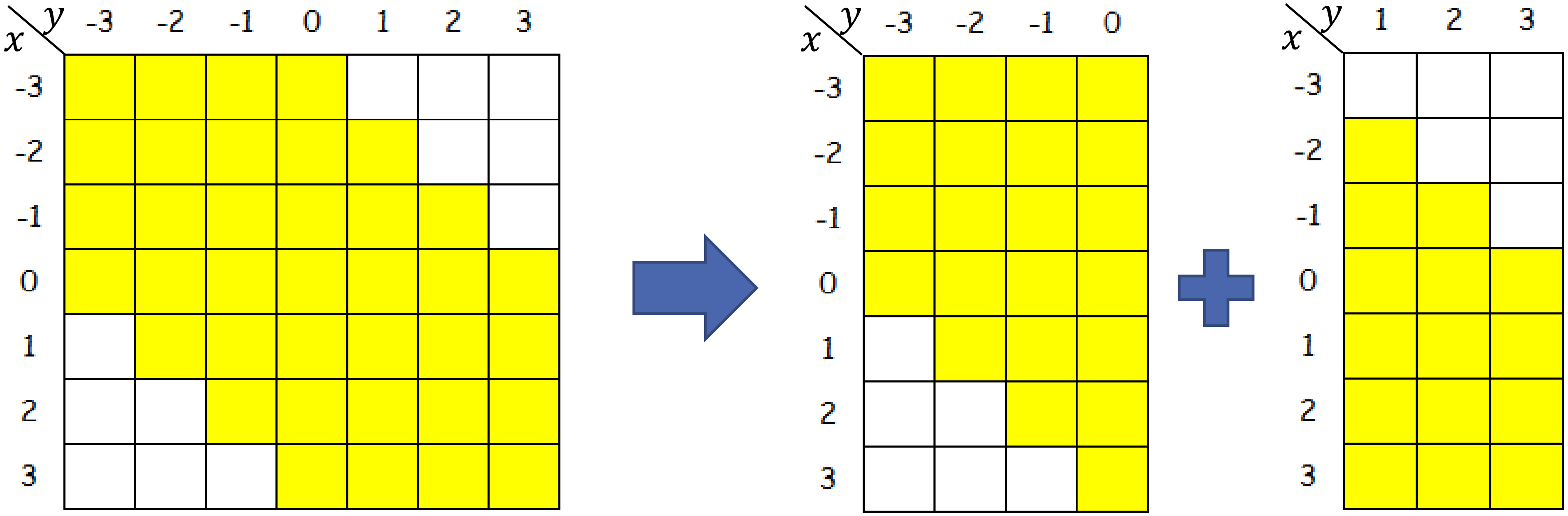
이렇게 생긴 요상한 도형에다가 1을 더해야 합니다. 막막하네요..

이왕 나누는 김에 직사각형으로 도형을 쪼개보려 하지만 잘 되지 않는 것 같습니다. 뭔가 다른 아이디어가 필요한 것 같습니다.

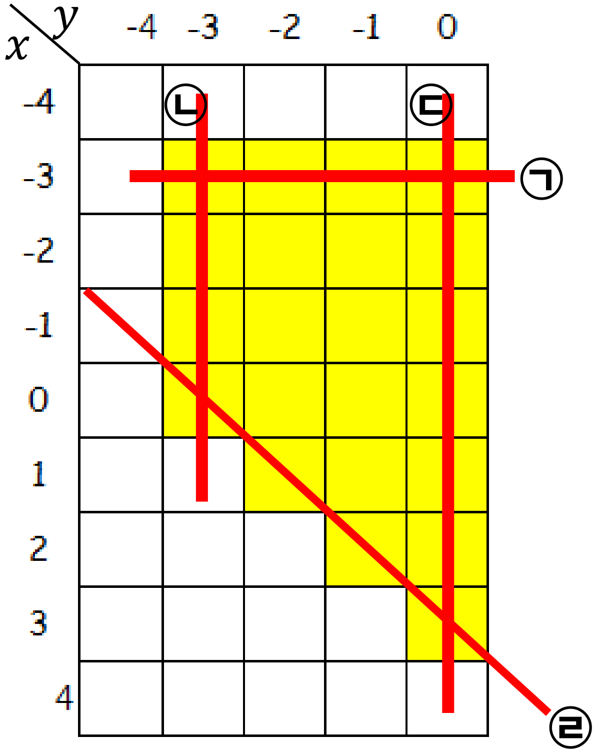
그래서..

I. 게나디는 머리가 좋습니다

도형을 반으로 쪼개서 비슷하게 생긴 두 개의 도형을 만듭니다. 둘 중 하나만 관찰해 봅시다.



I. 게나디는 머리가 좋습니다

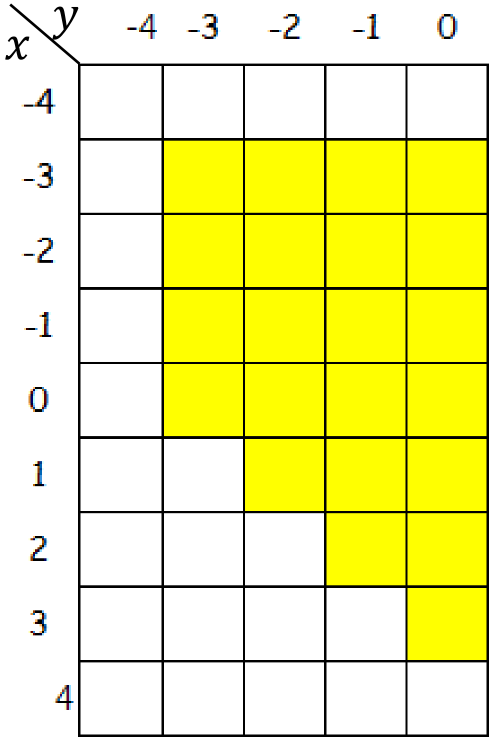


이 도형의 둘레를 살펴봅시다. 다행히도 둘레는 네 개의 선분으로 이루어져 있다고 생각할 수 있고, 각 선분의 방정식을 적어보면

ㄱ: $x = -3$, ㄴ: $y = -3$, ㄷ: $y = 0$, ㄹ: $y = x - 3$

과 같기에, 이 도형은 $\begin{cases} -3 \le y \le 0 \\ y \ge x - 3 \\ x \ge -3 \end{cases}$ 의 영역에 있다고 볼 수 있습니다.

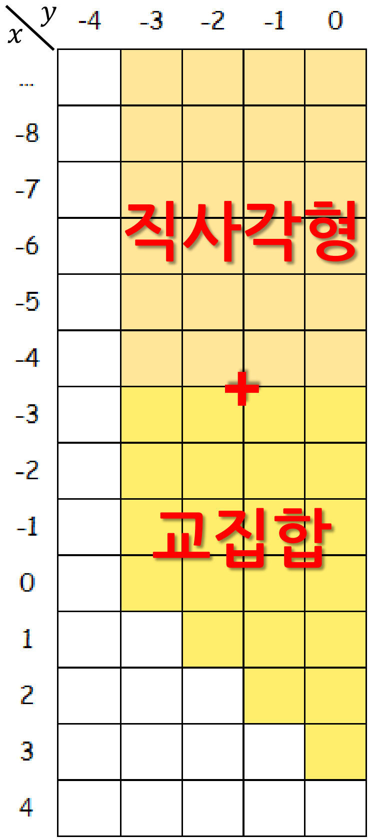
I. 게나디는 머리가 좋습니다



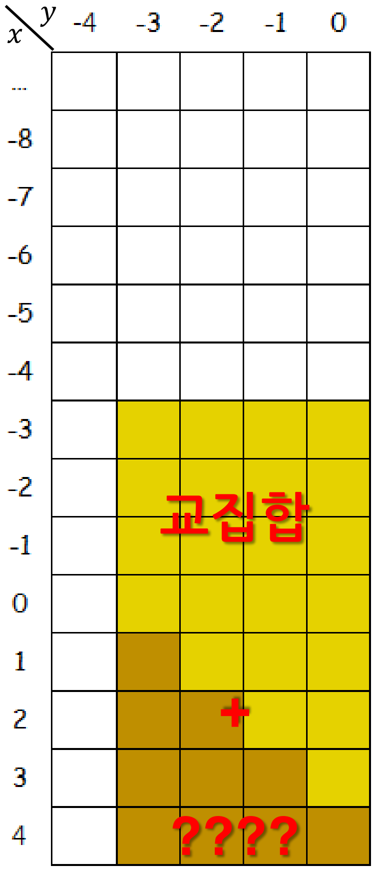
이 관계식을 바로 적용할 자료구조를 찾기는 좀 어려워 보이니, 나누어서 생각해 봅니다.

왼쪽의 영역은 $\begin{cases} -3 \le y \le 0 \\ y \ge x-3 \end{cases}$ 과 $\begin{cases} -3 \le y \le 0 \\ x \ge -3 \end{cases}$ 의 교집합입니다. 그럼 이 두 영역은 어떻게 생겼을까요?

I. 게나디는 머리가 좋습니다

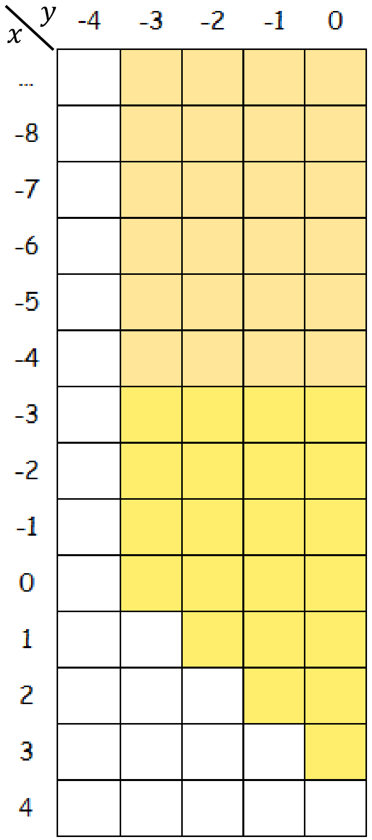


$$\begin{cases} -3 \le y \le 0 \\ y \ge x - 3 \end{cases}$$



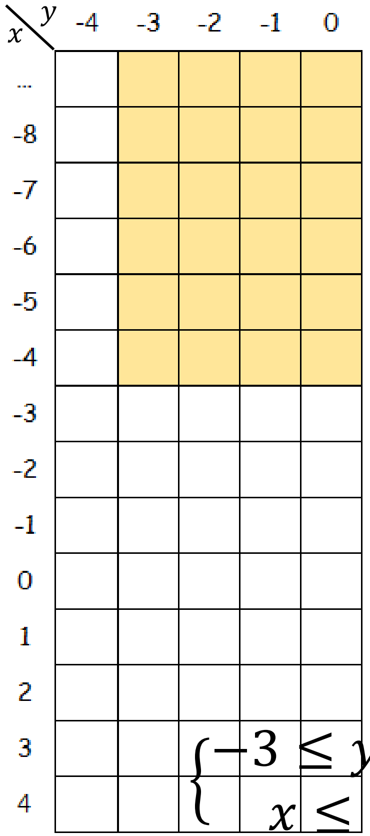
$$\begin{cases} -3 \le y \le 0 \\ x \ge -3 \end{cases}$$

I. 게나디는 머리가 좋습니다



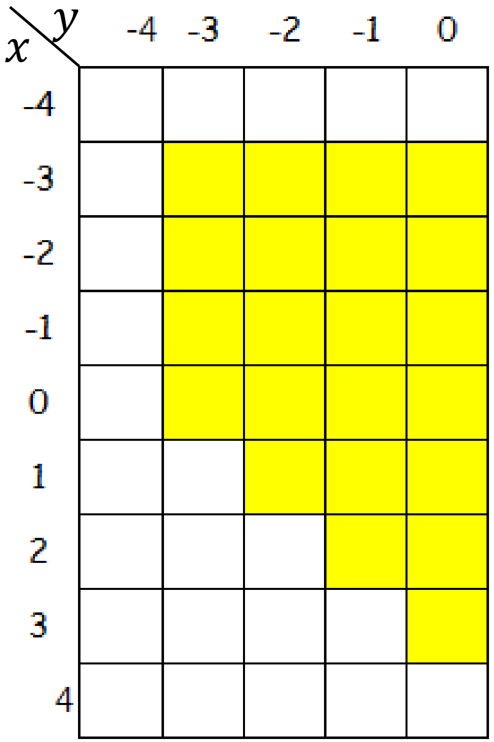
$$\begin{cases} -3 \le y \le 0 \\ y \ge x - 3 \end{cases}$$

−



$$\begin{cases} -3 \le y \le 0 \\ x \le -4 \end{cases}$$

=



I. 게나디는 머리가 좋습니다

$x \backslash y$	-4	-3	-2	-1	0
...		+1-1	+1-1	+1-1	+1-1
-8		+1-1	+1-1	+1-1	+1-1
-7		+1-1	+1-1	+1-1	+1-1
-6		+1-1	+1-1	+1-1	+1-1
-5		+1-1	+1-1	+1-1	+1-1
-4		+1-1	+1-1	+1-1	+1-1
-3		+1	+1	+1	+1
-2		+1	+1	+1	+1
-1		+1	+1	+1	+1
0		+1	+1	+1	+1
1			+1	+1	+1
2				+1	+1
3					+1
4					

따라서 $\begin{cases} -3 \leq y \leq 0 \\ y \geq x - 3 \end{cases}$ 에 속하는 모든 지점에 1을 더하고,

$\begin{cases} -3 \leq y \leq 0 \\ x \leq -4 \end{cases}$ 에 속하는 모든 지점에 1을 빼면

원하는 부분에만 1을 더한 것과 같은 효과를 보입니다.

이를 효율적으로 처리하기 위해 2개의 자료구조를 만듭니다.

I. 게나디는 머리가 좋습니다

1. $\begin{cases} -3 \leq y \leq 0 \\ y \geq x - 3 \end{cases}$ 에 1을 더하는 방법:
   - 이 식을 다시 쓰면 $\begin{cases} -3 \leq y \leq 0 \\ x - y \leq 3 \end{cases}$ 입니다.
   - 그래서 자료구조 $\boldsymbol{D_1}$을 만들어, 이 자료구조에서는 점 (x, y)를 $(y, x - y)$로 옮긴다고 생각하여, $[\mathbf{-3}, \mathbf{0}] \times [-\infty, \mathbf{3}]$의 영역에 1을 더합니다.
2. $\begin{cases} -3 \leq y \leq 0 \\ x \leq -4 \end{cases}$ 에서 1을 빼는 방법:
   - 자료구조 $\boldsymbol{D_2}$을 만들어, $[-\infty, \mathbf{-4}] \times [\mathbf{-3}, \mathbf{0}]$의 영역에서 1을 뺍니다.

직사각형 영역에 1을 더하거나 빼는 것은 2D Fenwick tree 등을 활용하여 할 수 있습니다.
그 방법에 대한 설명은 생략합니다.

I. 게나디는 머리가 좋습니다

이렇게 도형의 왼쪽 반을 어떻게 업데이트하는지에 대한 설명이 끝났습니다. 남은 오른쪽 반 역시 똑같은 방식으로 직사각형 영역에 1을 더하거나 빼도록 할 수 있습니다. 이제 1번 종류의 연산을 모두 해결할 수 있습니다.

우리에게 남은 건 2번 종류의 연산입니다. 특정 격자 (x, y)에 어떤 수가 적혀 있는지 알기 위해서는

- 자료구조 $\boldsymbol{D_1}$의 $(\boldsymbol{y}, \boldsymbol{x}-\boldsymbol{y})$에 적혀 있는 수와
- 자료구조 $\boldsymbol{D_2}$의 $(\boldsymbol{x}, \boldsymbol{y})$에 적혀 있는 수

를 합치면 됩니다.

Special thanks to

- beingryu
- cubelover
- etaehyun4
- kcm1700
- koosaga
- tncks0121
- xhae
- zigui

- NexonGT
- Startlink

Coder's high

감사합니다!

본선 때 봐요~