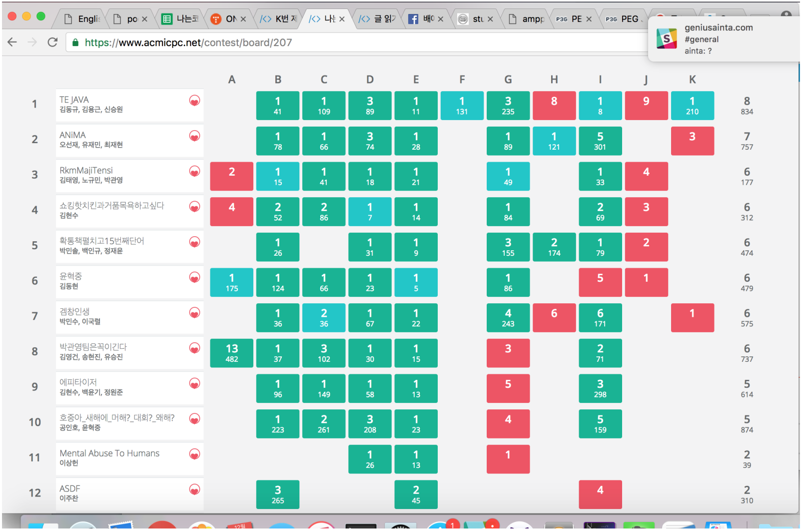
나는코더다 2016 송년 대회: Solution sketch

14004 구재현 (koosaga)
경기과학고등학교

도와주신 분들

- GSHS TeX Society의 자료들이 모든 조판을 LaTeX로 진행하는 데 큰 도움이 되었습니다. 운영해주신 학우 분들께 감사드립니다.
- 대회 진행에 아낌없는 지원을 약속해주신 경기과학고 선생님들에게 감사드립니다.
- 마지막으로, 이 모든 대회 시스템 준비는 스타트링크 사의 무료 지원이 아니었으면 불가능했습니다. 다시 한번 정말, 정말 감사드립니다.

Results : Main Contest



		A	B	C	D	E	F	G	H	I	J	K	
1	TE JAVA 김동규, 김용근, 신승원		1 41	1 109	3 89	1 11	1 131	3 235	8	1 8	9	1 210	8 834
2	ANiMA 오선재, 유재민, 최재현		1 78	1 66	3 74	1 28		1 89	1 121	5 301		3	7 757
3	RkmMajiTensi 김태영, 노규민, 박관영	2	1 15	1 41	1 18	1 21		1 49		1 33	4		6 177
4	쇼킹핫치킨과거품목욕하고싶다 김현수	4	2 52	2 86	1 7	1 14		1 84		2 69	3		6 312
5	확통책펼치고15번째단어 박민솔, 백인규, 정재윤		1 26		1 31	1 9		3 155	2 174	1 79	2		6 474
6	윤혁중 김동현	1 175	1 124	1 66	1 23	1 5		1 86		5	1		6 479
7	겜창인생 박민수, 이국렬		1 36	2 36	1 67	1 22		4 243	6	6 171		1	6 575
8	박관영팀은꼭이긴다 김영건, 송현진, 유승진	13 482	1 37	3 102	1 30	1 15		3		2 71			6 737
9	에피타이저 김현수, 백윤기, 정원준		1 96	1 149	1 58	1 13		5		3 298			5 614
10	호중아_새해에_머해?_대회?_왜해? 공인호, 윤혁중		1 223	2 261	3 208	1 23		4		5 159			5 874
11	Mental Abuse To Humans 이상헌				1 26	1 13		1					2 39
12	ASDF 이주찬		3 265			2 45				4			2 310

Results : Open Contest 1

(토요일날 참가해서 사진에 빠진 두명이 있습니다. ID tlwpdus님은 6/350 (6등), junis3님은 3/191 (10등).)

17	stack -				2 100	1 47							2 147
18	heehcs -				1 139	1 58		2					2 197
19	aod0326 -				1 107	2 141		2					2 248
20	nona1314 -				1 146	1 113							2 259
21	jjwdi0 -			1		1 137		1 133					2 270
22	jason9319 -				2 281	1 236							2 517
23	dongjae625 -					1 12				2			1 12
24	protalk -				2	2 56							1 56
25	gkswn121 -					2 75							1 75
26	jerichoi224 -				3	1 88		2					1 88
27	sunghwan2789 -				4	2 100							1 100
28	gnurails -					3 138							1 138
29	songuno1 -					1 208							1 208
30	hj4623 -				1	1 218							1 218
31	computerace13 -					1							
32	viodle238 -					1							
33	erik5022 -	10											
34	jaydoubluel -											1	

작 시간: 오후 1시 30분 종료 시간: 오후 6시 0분 Baekjoon Online Judge

Results : Open Contest 2

		A	B	C	D	E	F	G	H	I	J	K	
1	ainta -	1 35	7 332	3 108	1 10	1 15	2 219	1 41	4 173	1 19	16 561		10 1513
2	dotorya -	2 78	1 20	1 62	1 6	1 3	1 73	1 40	2 256	2 54	4		9 592
3	gs13105 -	3 271	1 68	1 150	2 26	1 9	1 131	1 112		5 130	1		8 897
4	august14 -	4 216	1 110	1 97	1 183	1 8	6	1 114	2 111	4 240	3		8 1079
5	dlwodnsdl -		1 26	3 161	1 42	1 31	1 162	2 100		2 114			7 636
6	doju -		1 30	1 207	1 15	1 4		1 52		1 43			6 351
7	bupjae -	1	1 84		1 45	1 33		1 63					4 225
8	exqt -		7	1 114	1 244	1 7		5 285		1			4 650
9	choiking10 -		2 260		1 115	1 105		2 172					4 652
10	citizen -	1	2 125		2 73	1 60							3 258
11	suprodigy -			2	3 148	1 20	1 209						3 377
12	cki86201 -				1 16	1 18							2 34
13	onjo0127 -			1 39		1 17		2					2 56
14	jang718 -	5			1 60	1 13			4				2 73
15	hhj1221 -		2 51			1 40		3		5			2 91
16	operaghost -	2			2 85	1 32							2 117
17	stack -				2 100	1 47							2 147

Problem Statistics

	본선 정답자수 (12팀 참가)	오픈 정답자수 (36명 참가)
A	2 (175min)	4 (35min)
B	11 (15min)	12 (20min)
C	9 (16min)	9 (39min)
D	11 (7min)	21 (6min)
E	12 (5min)	32 (3min)
F	1 (131min)	5 (73min)
G	7 (49min)	11 (40min)
H	2 (121min)	3 (111min)
I	9 (8min)	7 (19min)
J	0	1 (261min)
K	1 (210min)	0
1등	TE JAVA (8 / 834)	ainta (10 / 1513)

A. 버스 여행

$O((nm)^2)$ 알고리즘

- 관광 명소 각각의 격자상 위치를 (x_i, y_i) 라 하고, 방문 시의 이득을 c_i, 건설 시기를 a_i 라고 합시다.
- a_i 순으로 정렬 후, 동적 계획법을 사용할 수 있습니다. dp_i 를 i번 관광 명소를 방문했을 때의 이득이라 정의하면,
- $dp_i = max_{a_j < a_i}(dp_j + |x_j - x_i| + |y_j - y_i|) + c_i$

A. 버스 여행

$O((nm)^2)$ 알고리즘

- 관광 명소 각각의 격자상 위치를 (x_i, y_i) 라 하고, 방문 시의 이득을 c_i, 건설 시기를 a_i 라고 합시다.
- a_i 순으로 정렬 후, 동적 계획법을 사용할 수 있습니다. dp_i 를 i번 관광 명소를 방문했을 때의 이득이라 정의하면,
- $dp_i = max_{a_j < a_i}(dp_j + |x_j - x_i| + |y_j - y_i|) + c_i$

... 너무 느립니다. 어떻게 고쳐야 할까요?

A. 버스 여행

분석 1

- 일단 절댓값 식을 풀어봅시다. 4가지 케이스가 나옵니다.
- $x_i \le x_j$, $y_i \le y_j$ 를 만족할 경우,
  $dp_i = max_{a_j < a_i}(dp_j + x_j - x_i + y_j - y_i) + c_i$ 라는 식이 나옵니다.
- 풀어 쓰면 $dp_i = max_{a_j < a_i}(dp_j + x_j + y_j) - x_i - y_i + c_i$ 으로 정리됩니다.
- 나머지 케이스들도 거의 비슷한 식으로 전개 가능합니다.

A. 버스 여행

분석 2

- 고로 $max_{a_j<a_i, x_i \le x_j, y_i \le y_j}(dp_j + x_j + y_j)$를 빠르게 구하면, 거기다가 $-x_i - y_i + c_i$ 만 더해주면 됩니다.
- 나머지 3개의 케이스도 똑같이 처리해주면 되니까, 저걸 빠르게 구하는 데 주목해 봅시다.
- $a_j < a_i$ 는 같은 a_i 를 현명하게 계산하면 쉽게 처리할 수 있습니다. 설명은 패스.
- $x_i \le x_j, y_i \le y_j$ 가 어려워 보이네요. 그런데....

A. 버스 여행

분석 3

- 저 조건이 없이 순수히 $max_{j<i, a_j<a_i}(dp_j + x_j + y_j) - x_i - y_i + c_i$ 만 계산한다면, 어떤 문제가 생기나요?
- $x_i \leq x_j, y_i \leq y_j$ 일 때는 잘 구해지겠죠. 아닐 때는 계산한 식의 절댓값이 음수가 되어서 틀린 답이 될 겁니다.

A. 버스 여행

분석 3

- 저 조건이 없이 순수히 $max_{j<i, a_j<a_i}(dp_j + x_j + y_j) - x_i - y_i + c_i$ 만 계산한다면, 어떤 문제가 생기나요?
- $x_i \le x_j, y_i \le y_j$ 일 때는 잘 구해지겠죠. 아닐 때는 계산한 식의 절댓값이 음수가 되어서 틀린 답이 될 겁니다.

Key Observation

- **그래도 상관 없습니다.** 절댓값이 음수가 되었다는 것은 원래 답보다 작은 답이 계산된다는 것입니다. 나머지 3개의 케이스를 처리하면서, 원래 답이 결국 반영됩니다.

A. 버스 여행

$O(nmlg(nm))$으로의 개선

- $max_{j<i, a_j<a_i}(dp_j + x_j + y_j)$ 는 변수 하나로 저장할 수 있습니다.
- 나머지 케이스도 각각 변수 하나로 저장할 수 있습니다.
- dp_i 를 계산할 때, 4개의 케이스를 $O(1)$ 만에 처리할 수 있으며, 변수 역시 $O(1)$에 갱신할 수 있습니다. 고로 dp 배열은 $O(nm)$에 채울 수 있습니다.
- 알고리즘의 최종 수행 시간은 a_i 순으로 정렬하는 데 드는 $O(nmlg(nm))$ 입니다. AC.

A. 버스 여행

$O(nmlg(nm))$으로의 개선

- $max_{j<i, a_j<a_i}(dp_j + x_j + y_j)$ 는 변수 하나로 저장할 수 있습니다.
- 나머지 케이스도 각각 변수 하나로 저장할 수 있습니다.
- dp_i 를 계산할 때, 4개의 케이스를 $O(1)$ 만에 처리할 수 있으며, 변수 역시 $O(1)$에 갱신할 수 있습니다. 고로 dp 배열은 $O(nm)$에 채울 수 있습니다.
- 알고리즘의 최종 수행 시간은 a_i 순으로 정렬하는 데 드는 $O(nmlg(nm))$ 입니다. AC.
- counting sort를 통해 $O(nm + max(a_i))$도 가능합니다.

A. 버스 여행

$O(nmlg(n)lg(m))$

- Key Observation을 안 해도 $O(nmlg(n)lg(m))$ 에 2D Segment Tree를 통해 구현할 수 있습니다. 자세한 설명은 생략합니다.
- Fenwick Tree스럽게 구현한 후 FastIO 를 쓰면 860ms 걸립니다. FastIO 안쓴 정해가 430ms라 그냥 막는 건 포기..

B. 큐브러버

풀이 1

- $N = 4$ 일 때, x_1, x_2, x_3, x_4 를 알고 있으면 유일한 a, b, c, d를 항상 찾을 수 있습니다.
- 고로, $N \leq 4$ 일 때는 항상 YES입니다. $N > 4$ 일 때는 찾은 a, b, c, d를 x_5 부터 대입해서 식이 맞는지 검사하면 됩니다.
- 그래서 a, b, c, d 를 어떻게 찾을까요?

B. 큐브러버

풀이 1

- $N = 4$ 일 때, x_1, x_2, x_3, x_4 를 알고 있으면 유일한 a, b, c, d를 항상 찾을 수 있습니다.
- 고로, $N \leq 4$ 일 때는 항상 YES입니다. $N > 4$ 일 때는 찾은 a, b, c, d를 x_5 부터 대입해서 식이 맞는지 검사하면 됩니다.
- 그래서 a, b, c, d 를 어떻게 찾을까요? $x_1..x_4$ 에 대해 쓰면 4 개의 일차연립방정식이 나옵니다. 손으로 풀기는 번거로우니 울프람알파로 역행렬을 구해서 찾으면 됩니다.
- 분수를 동반하지만 어쨌든 모든 계산은 정수 범위에서 가능합니다.

B. 큐브러버

풀이 2

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,

B. 큐브러버

풀이 2

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,
- 모든 $2 \le i \le N$ 에 대해 $y_i = x_i - x_{i-1} = f_2(i)$인 2차 이하의 다항식 f_2가 존재한다는 것과 필요충분 관계이고,

B. 큐브러버

풀이 2

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,
- 모든 $2 \le i \le N$ 에 대해 $y_i = x_i - x_{i-1} = f_2(i)$인 2차 이하의 다항식 f_2가 존재한다는 것과 필요충분 관계이고,
- 모든 $3 \le i \le N$ 에 대해 $z_i = y_i - y_{i-1} = f_3(i)$인 1차 이하의 다항식 f_3가 존재한다는 것과 필요충분 관계이고,

B. 큐브러버

풀이 2

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,
- 모든 $2 \le i \le N$ 에 대해 $y_i = x_i - x_{i-1} = f_2(i)$인 2차 이하의 다항식 f_2가 존재한다는 것과 필요충분 관계이고,
- 모든 $3 \le i \le N$ 에 대해 $z_i = y_i - y_{i-1} = f_3(i)$인 1차 이하의 다항식 f_3가 존재한다는 것과 필요충분 관계이고,
- 모든 $4 \le i \le N$ 에 대해 $w_i = z_i - z_{i-1} = f_4(i)$인 상수 함수..

B. 큐브러버

풀이 2

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,
- 모든 $2 \le i \le N$ 에 대해 $y_i = x_i - x_{i-1} = f_2(i)$인 2차 이하의 다항식 f_2가 존재한다는 것과 필요충분 관계이고,
- 모든 $3 \le i \le N$ 에 대해 $z_i = y_i - y_{i-1} = f_3(i)$인 1차 이하의 다항식 f_3가 존재한다는 것과 필요충분 관계이고,
- 모든 $4 \le i \le N$ 에 대해 $w_i = z_i - z_{i-1} = f_4(i)$인 상수 함수..
- ????

B. 큐브러버

풀이 2 (미분 아이디어)

- 모든 $1 \le i \le N$ 에 대해 $x_i = f_1(i)$인 3차 이하의 다항식 f_1가 존재한다는 것은,
- 모든 $2 \le i \le N$ 에 대해 $y_i = x_i - x_{i-1} = f_2(i)$인 2차 이하의 다항식 f_2가 존재한다는 것과 필요충분 관계이고,
- 모든 $3 \le i \le N$ 에 대해 $z_i = y_i - y_{i-1} = f_3(i)$인 1차 이하의 다항식 f_3가 존재한다는 것과 필요충분 관계이고,
- 모든 $4 \le i \le N$ 에 대해 $w_i = z_i - z_{i-1} = f_4(i)$인 상수 함수..
- ????
- PROFIT!

C. 크리스 마틴

Greedy Solution

- A, C, G, T 중 가장 등장 수가 작은 알파벳을 찾아서, 그걸 n번 찍으면 풀 수 있습니다.
- 왜 될까요?

C. 크리스 마틴

Greedy Solution

- A, C, G, T 중 가장 등장 수가 작은 알파벳을 찾아서, 그걸 n번 찍으면 풀 수 있습니다.
- 왜 될까요?
- 가장 등장 수가 작은 알파벳의 등장 수를 F라고 하고, 더 작은 답을 내는 문자열이 존재한다고 칩니다.
- 이 문자열의 A, C, G, T의 개수는 모두 $F - 1$ 이하여야 합니다. (아니면 그 알파벳만 찍는 공통 부분 수열이 존재)
- 다 합치면 길이가 $4F - 4$ 이하입니다. $F \leq n/4$ 라서 이러한 길이 n의 문자열은 존재하지 않습니다.

D. 내일 할거야

$O(nlgn)$ Greedy Solution

- 보장이 되어 있다고는 하지만, 만약에 보장이 되어 있는지 없는지를 판별하는 문제를 푼다고 합니다. 어떻게 하나요?
- 현실에서 여러분들은 어떻게 하시나요? 저는 데드라인이 가장 빠른 것부터 하는데...

D. 내일 할거야

$O(nlgn)$ Greedy Solution

- 보장이 되어 있다고는 하지만, 만약에 보장이 되어 있는지 없는지를 판별하는 문제를 푼다고 합니다. 어떻게 하나요?
- 현실에서 여러분들은 어떻게 하시나요? 저는 데드라인이 가장 빠른 것부터 하는데...
- 네, 데드라인이 빠른 순으로 정렬해서, 그 순서대로 하는 것이 최적입니다. proof left as an exercise ㅎㅎ

D. 내일 할거야

$O(nlgn)$ Greedy Solution

- 초기에 노는 시간을 T만큼 확보한다는 것은, 데드라인을 모두 T 만큼 감소시킨다는 것과 동치입니다.
- 데드라인을 모두 일정하게 T만큼 감소시켜도, 데드라인의 대소 관계는 변하지 않습니다. 여전히 과제를 하는 가장 효율적인 방법은 데드라인이 빠른 순으로 진행하는 것입니다.
- 실제 얼마만큼의 노는 시간을 확보할 수 있는지는, 실제 배정 순의 역순으로 시간을 계산해나가면서 가능합니다. 아니면, Parametric Search를 사용해도 됩니다.

E. 풍선 놀이

Do you know C++?

- 풍선이 설치되었는지 아닌지를 배열로 관리합니다.
- 반복문으로 풍선을 설치합니다.
- `for(int j=1; j<=n; j+=i) chk[j] = 1;`
- 최후에 배열을 전부 돌면서 `chk[i] == 0`인 원소를 셉니다.
- 꿀팁 : `cout << count(chk+1, chk+n+1, 0);`

F. 장비를 정지합니다

$O(n+m)$ 알고리즘 1

- DFS를 통해 DP를 구현해 봅시다. $mincost(i) =$ i번 정점의 장비를 정지하는 최소 비용 이라고 정의합시다.
- 강한 충격을 주면 z_i만큼 쓰면 되고, 약한 충격일 때는 $u_i + \sum_{j \in g_i} mincost(j)$의 비용이 듭니다. 정의는 쉽습니다.
- 하지만, 이대로 짜면 예제도 안나옵니다. 무한루프...

F. 장비를 정지합니다

$O(n+m)$ 알고리즘 2

- 무한 루프가 걸리는 이유는, $mincost(i)$ 를 계산하기 위해 $mincost(i)$ 를 호출하는 일이 생기기 때문입니다. 생각해보면, 이러한 형태로는 최적화가 되지 않기 때문에 쓸모가 없습니다.
- 고로 이러한 경우를 없애기 위해서, 현재 계산중인 (계산이 이뤄지지 않은, 아주 끝난것도 아닌, 계산 중인) 정점을 호출했을 때는 더 진행하지 않고 ∞ 를 반환합니다.
- 재귀 stack에 있는 원소를 체크하는 식으로 할 수 있습니다.
- 시간 복잡도는 $O(n+m)$ 입니다.

G. 순열 그래프의 연결성 판별

$O(n)$ 풀이

- i 까지의 연결성을 결정했을 때, 이를 토대로 $i+1$까지의 연결성을 결정해 봅시다.
- 각각의 컴포넌트는 순열 상에서 연결된 구간의 형태 $(s, s+1, s+2, ..., e)$ 를 띕니다. 스택에 각각의 컴포넌트를 번호 순으로 저장합니다.
- 현재 정점을 유일한 원소로 갖는 컴포넌트를 만듭니다.
- 현재 들어간 정점이 스택의 top에 있는 컴포넌트에 속한다면, 현재 컴포넌트에 스택의 top에 있는 컴포넌트를 합쳐줍니다. 그렇지 않다면 현재 컴포넌트를 스택에 넣습니다.

G. 순열 그래프의 연결성 판별

$O(n)$ 풀이

- 그럴듯 하지만, 그냥 짜면 왠지 $O(n^2)$ 가 나올 것 같은 풀이입니다.
- 이를 타개하기 위해서 스택에 컴포넌트를 조금 더 현명하게 넣어야 합니다. 각각의 컴포넌트에 4개의 정보를 부여합시다 : 구간의 시작점, 끝점, 구간 a_i 최대, 최소.
- 이 4가지 값을 통해서 위에 제시한 연산을 모두 $O(n)$에 할 수 있습니다.
- n 개의 연결성을 모두 파악했다면 print만 해주면 됩니다.

G. 순열 그래프의 연결성 판별

$O(n)$ 풀이 2

- 순열 그래프의 컴포넌트는 연속한 구간으로 쪼개져 있다고 말했습니다.
- 그렇다면, 어떠한 상황에 구간이 쪼개질까요?
- $[1, i]$ 구간의 a_i 최댓값이 i인 것과 구간이 $[?, i]$ 와 $[i+1, ?]$으로 쪼개지는 것이 동치입니다.
- 고로 쉽게 $O(n)$에 풀 수 있습니다.

H. 두더지 잡기

$O(nk)$ 동적 계획법

- 한번 두더지를 쫓아낸 곳에 또 다시 두더지를 쫓을 이유는 없습니다. (굳이 상황을 만들 수는 있지만 매우 비효율적...)
- 원호에다가 두더지가 도망간 곳을 다 마킹했다고 쳐 봅시다.
- 당연하지만 마킹된 곳에 있는 두더지는 다 도망갑니다.
- 인접한 두 마킹된 위치가 거리 3 이상이라면, 그 사이에 있는 두더지는 도망가지 않습니다.
- 인접한 두 마킹된 위치가 거리 2 이하면, 그 사이에 있는 두더지를 도망가게 할 수 있습니다.
- OVOOVOVVOVOOV $\rightarrow$ XXOOXXXXXXOOX

H. 두더지 잡기

$O(nk)$ 동적 계획법

- 결론적으로, 두 쫓아낸 위치 사이에 있는 두더지들은 모두 도망갑니다.
- 원을 직선으로 폈다면, $O(nk)$ 에 문제를 해결할 수 있습니다. $dp_{i,j} = i$ 번 지점까지 총으로 j번 쫓아냈을 경우라고 정의하면 됩니다.
- 하지만 직선으로 펼 수 없죠... 원호에서는 이를 적당히 처리하셔야 합니다.
- 이 부분은 알고리즘적으로 어렵지는 않고 구현이 힘듭니다.

H. 두더지 잡기

$O(nk)$ 동적 계획법

- 결론적으로, 두 쫓아낸 위치 사이에 있는 두더지들은 모두 도망갑니다.
- 원을 직선으로 폈다면, $O(nk)$ 에 문제를 해결할 수 있습니다. $dp_{i,j} = i$ 번 지점까지 총으로 j번 쫓아냈을 경우라고 정의하면 됩니다.
- 하지만 직선으로 펼 수 없죠... 원호에서는 이를 적당히 처리하셔야 합니다.
- 이 부분은 알고리즘적으로 어렵지는 않고 구현이 힘듭니다.
- 그래서 Left as an exercise ㅎㅎㅎ

I. 수열

$O(n)$ 동적 계획법

- 홀수냐 짝수냐만이 중요합니다. 고로 홀짝성을 적절히 결정해서 주어진 식을 만족시켜야 합니다.
- 모든 $1 \le i \le n-k$ 에 대해서,
  $a_i + a_{i+1} + \ldots + a_{i+k-1} \equiv a_{i+1} + \ldots + a_{i+k-1} + a_{i+k} \pmod 2$
  입니다.
- 양변을 정리하면 모든 $1 \le i \le n-k$ 에 대해서,
  $a_i \equiv a_{i+k} \pmod 2$ 입니다.
- 이 때문에, $a_1, \ldots, a_k$ 의 홀짝성을 정해주면 그 이후 원소의 홀짝성은 자동으로 결정됩니다.

I. 수열

$O(n)$ 동적 계획법

- $a_1 + a_2 + ... + a_k \equiv 0 \ (mod\ 2)$ 입니다.
- 이걸 만족하면 나머지 식은 다 만족이 됩니다.
- a_i 를 짝수로 만들면, $a_i, a_{i+k}, a_{i+2k}..$ 중 홀수인 애들은 전부 바꿔줘야 합니다. 이런 개수를 다 세줍시다.
- 이제 $dp_{i,j} = i$번 원소까지의 합이 $j \ (mod\ 2)$ 인 경우의 수 라고 정의하고 풀면, $O(n)$ 에 해결 가능합니다.

J. 비무장 지대

관찰 1

- 각각의 지뢰를 터트렸을 때, 연속적인 지뢰 폭발이 일어날 것입니다. 이러한 연속적인 폭발로 파괴되는 지역은 하나의 연속된 구간을 이룹니다.
- 이걸 PL_i, PR_i 와 같이 어떻게 계산했다고 쳐봅시다. 이후에는 정렬 후 간단한 sweeping을 하면 오프라인으로 $O((n+m)lg(n+m))$에 풀 수 있습니다.
- 핵심은 PL_i, PR_i 를 빠르게 계산하는 겁니다.

J. 비무장 지대

관찰 2

- 여기서 아주 다양한 풀이가 있을 것이라고 생각합니다. 제 풀이는 다음과 같습니다.
- 편의상 PL_i 만 구하다고 하겠습니다. PR_i 는 반대로 뒤집어서 똑같이 계산 가능합니다.
- PL_i 는 어떠한 지뢰의 L_i 값 중 하나일 겁니다. 각각의 지뢰를 L_i 순으로 정렬해서, Flood Fill의 요령으로 거꾸로 PL_i 에 칠해봅시다.
- 단순하게 하면 $O(n^2)$니까, $O(nlgn)$ 으로 줄여봅시다.

J. 비무장 지대

관찰 3

- 거꾸로 하는 거니까, 현재 지뢰를 터트릴 수 있는 지뢰의 조건을 생각해 봅시다. $X_j - L_j <= X_i$ 이며, $X_j + R_j >= X_i$ 인 지뢰 j 는 i에서 도달 가능합니다.
- 자료 구조를 생각해 봅시다. 일단 저러한 지뢰 하나를 방문하기만 하면, 다시 방문할 필요가 없습니다. $O(lgn)$ 시간 안에 저러한 지뢰 하나를 반환하고, 지울 수 있다면...
- **Segment Tree를 사용합시다!** $X_j - L_j$ 를 인덱스로 하고, $X_j + R_j$ 를 값으로 하는 Maximum Segment Tree를 사용합니다.

J. 비무장 지대

$O(nlgn)$ 풀이

- $X_j - L_j$ 이하의 구간에서, $X_j + R_j$ 값을 최대로 하는 지뢰 하나가 만약에 도달 불가능하다면, 그 상황에서 종료합니다.
- 그렇지 않다면, 그 지뢰를 제거한 후 방문합니다.
- 지뢰를 지우는 것은 원소 갱신으로 대체 가능합니다.
- 좌표 범위가 큰 것은 적당한 좌표 압축으로 해결합시다.

J. 비무장 지대

연습 문제

- 이런 식으로 특수한 그래프에서 DFS를 빠르게 하는 유형의 문제들이 몇개 있습니다. 굳이 정해가 아니더라도 굉장히 여러 문제의 풀이에 도움이 돼요.
- Can of Worms: `https://www.acmicpc.net/problem/7149`
- 전선 연결하기: `https://www.acmicpc.net/problem/11915/`

K. 그래프와 쿼리

DP + 이진 탐색 $O(nm + qlgn)$

- 풀이가 두개 있습니다. 이건 제가 풀었던 방법입니다.
- 쿼리를 오프라인으로 처리해 봅시다. 간선을 끊는 대신, 각각의 간선에 가중치 t_i 를 매겨봅시다.
- 간선이 끝까지 끊기지 않는다면 $t_i = \infty$ 이고, 끊긴다면 그 쿼리가 들어온 시점을 t_i 에 저장합니다.
- 어떠한 경로가 쿼리 시점 T에 사용 가능하려면, 그 경로 상에 있는 모든 간선이 $t_i > T$ 를 만족해야 합니다.

K. 그래프와 쿼리

DP + 이진 탐색 $O(nm + qlgn)$

- 이제 DP를 합시다. DP 정의는 벨만 포드와 유사합니다.
- $dp_{i,j} = 1$ - j를 잇는 모든 길이 i 이하의 경로 중, t의 최솟값을 최대화하는 경로의 t 값.
- $O(nm)$ 에 계산 가능합니다.
- 이제, E 쿼리를 처리합니다. $dp_{dist,p} > i$ 를 만족하는 최소의 $dist$ 가 E 쿼리의 답입니다.
- 이는 이진 탐색으로 $O(qlgn)$ 에 찾을 수 있으며, 사실 $O(qn)$ 에 찾아도 엄청 빠르게 나옵니다.

K. 그래프와 쿼리

다른 풀이

- TE JAVA 팀과 박범수 (IOI 2012 금메달리스트) 님이 이렇게 푸셨습니다.
- 쿼리를 거꾸로 처리해서, 간선을 더해주는 형태로 처리합니다. 처음에 BFS로 최단 거리를 다 저장합니다.
- 간선이 추가되면 최단 거리가 줄어들기도 합니다. 이 때 줄어드는 모든 경우에 대해서 BFS로 naive하게 뿌려줍니다.
- naive하게 뿌려줬으니 $O(qm)$ 이라고 생각할 수 있겠지만, 뿌려주는 경우에 대해서는 거리가 항상 감소하거나, 도달 불가능한 정점이 도달 가능해 집니다. 각각의 정점을 방문하는 횟수는 많아야 $O(n)$번이라, 시간 복잡도는 $O(nm+q)$ 입니다.