UCPC 2024

전국 대학생 프로그래밍 대회 동아리 연합
여름 대회 2024

Finals

Official Solutions

본선 해설

전국 대학생 프로그래밍 대회 동아리 연합 · UCPC 2024 출제진

박승원 (veydpz) 김동현 (kdh9949) 정재현 (Gravekper) pjshwa 임지환 (raararaara) kclee2172 이종서 (leejseo) 이상헌 (evenharder) 김준겸 (ryute)

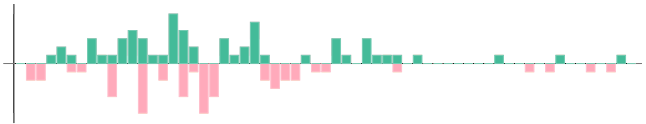
문제		의도한 난이도	출제자
A	관광 코스	**Easy**	golazcc83
B	Rolling Rick	**Hard**	functionx
C	배고픈 무토를 위한 피자 만들기	**Medium**	golazcc83
D	SoleMap	**Easy**	kipa00
E	돌 놓기 게임	**Medium**	kclee2172
F	4색 정리	**Medium**	qwerasdfzxcl
G	$\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리	**Hard**	azberjibiou
H	감옥	**Medium**	gs22123
I	러시안 회전초밥	**Hard**	queued_q
J	Distance Sum Maximization	**Medium**	cki86201
K	스레드	**Hard**	jh05013
L	밤양갱	**Medium**	sorohue
M	지루함 줄이기	**Medium**	exqt

A. 관광 코스

`constructive`, `ad-hoc`

출제진 의도 – **Easy**

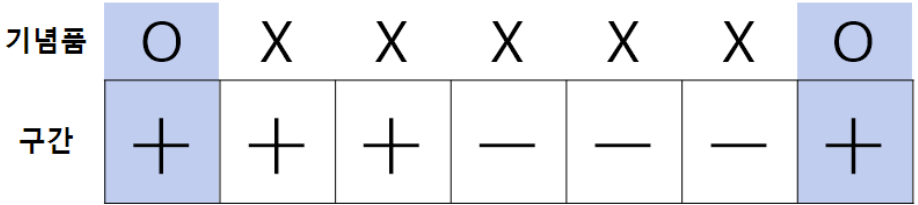
- 제출 151번, 정답 60팀 (정답률 41.060%)
- 처음 푼 팀: **jjmik12321** (jjmjjm, 1234321, ikik), 17분
- 출제자: `golazcc83`
- 가장 짧은 정해: 712B^{C++}, 669B^{Python}



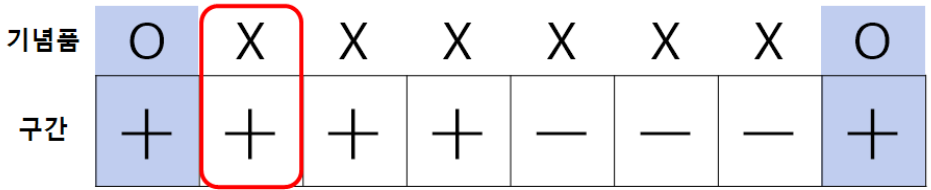
A. 관광 코스

- 편의상 기념품을 구매한 관광객의 시작 구간을 O구간, 구매하지 않았다면 X구간으로 표현하겠습니다.
- 모든 구간이 X구간이라면 모든 구간이 사막인 관광 코스가 조건을 만족합니다.
- 특정 구간이 O구간이라면, 그 구간은 설원입니다.
  - 사막이라면 첫 구간에서 호감도가 0이 되어 왕국을 떠나기 때문입니다.
- 특정 구간이 O구간이라면, 그 구간부터 호감도의 변화가 0 미만이 되는 경우는 없습니다.
  - 관광객이 O구간을 지났다면 그 이후의 호감도는 계산하지 않아도 기념품을 구매한다는 사실을 알 수 있습니다.
- 따라서 어떤 관광객이 기념품을 구매하지 않기 위해서는 O구간을 지나기 전에 호감도가 0이 되어야 합니다.

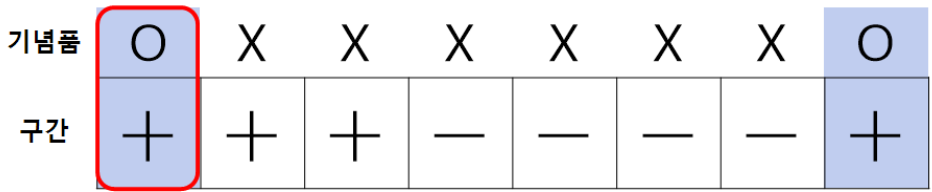
- 이제 X구간이 연속한 경우를 생각해봅시다.
- 연속한 X구간의 길이가 홀수 $(2M+1)$ 라면, M 개의 연속한 설원과 $M+1$ 개의 연속한 사막을 배치할 수 있습니다.



- 연속한 X구간의 길이가 짝수 $(2M)$ 일 때를 생각해봅시다.
- 설원이 M 개 이상이라면, X구간의 첫 번째 설원에서 관광을 시작했을 때 호감도가 0 미만으로 떨어지지 않고 O구간에 도달할 수 있으므로 불가능한 구성입니다.



- 연속한 X구간의 길이가 짝수 ($2M$) 일 때를 생각해봅시다.
- 설원이 M개 미만이라면, X구간 이전 마지막 O구간에서 관광을 시작했을 때 X구간을 지나는 도중 호감도가 0 미만으로 떨어지게 되므로 불가능한 구성입니다.



- 따라서 연속한 X구간의 길이가 짝수인 구간이 존재한다면 관광 코스를 구성할 수 없습니다.

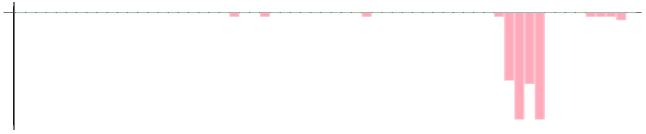
B. Rolling Rick

UCPC 2024

dp, ad-hoc

출제진 의도 – **Hard**

- 제출 112번, 정답 0팀 (정답률 -%)
- 처음 푼 팀: -
- 출제자: `functionx`
- 가장 짧은 정해: 2,278B^{C++}, 8,874B^{Python}

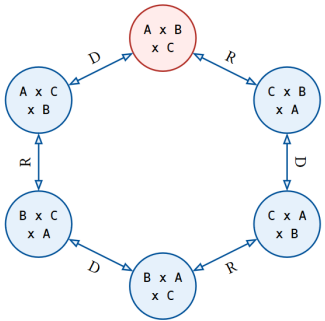


아래와 같이 간단한 DP 식을 세울 수 있습니다.

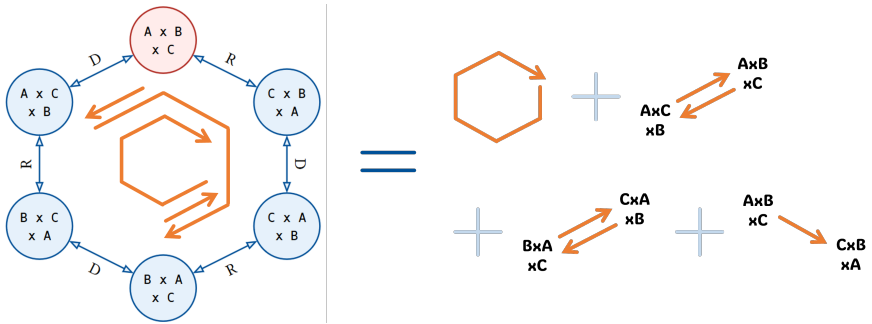
- $D[M][N][a][b][c]$: 크기가 $M \times N$인 종이에서 가로 a, 세로 b, 높이 c인 Rick을 굴릴 때 답
- $D[M][N][a][b][c] = MN + \max(D[M-a][N][c][b][a], D[M][N-b][a][c][b])$

  시간복잡도는 $\mathcal{O}(MN)$으로, 이 방법으로는 문제를 풀 수 없습니다.

앞서 정의한 DP에서, Rick의 상태 (가로, 세로, 높이) 로 가능한 경우는 총 6가지입니다.
상태 간 그래프를 그려보면 아래와 같습니다.



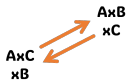
Rick을 굴리는 방법은 그래프에서 보면 경로와 같습니다.
경로에서 방문하는 노드의 순서를 적절히 바꿔서 단순 경로와 단순 싸이클로 분리할 수 있습니다.



UCPC 2024

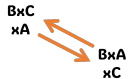
그래프에서 만들 수 있는 단순 싸이클은 총 8가지, 단순 경로는 총 11가지 (빈 경로 포함) 입니다.

좌표 변화 : +(A+B+C, A+B+C)
넓이 변화 : +2(AB+BC+CA)



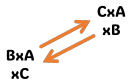
좌표 변화 : +(0, B+C)
넓이 변화 : +(AB+AC)

좌표 변화 : +(0, A+B)
넓이 변화 : +(CA+CB)

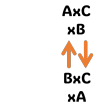


좌표 변화 : +(0, C+A)
넓이 변화 : +(BC+BA)

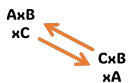
좌표 변화 : +(A+B+C, A+B+C)
넓이 변화 : +2(AB+BC+CA)



좌표 변화 : +(B+C, 0)
넓이 변화 : +(AB+AC)



좌표 변화 : +(A+B, 0)
넓이 변화 : +(CA+CB)



좌표 변화 : +(C+A, 0)
넓이 변화 : +(BC+BA)

- 단순 경로를 하나 정한 다음, 단순 싸이클을 적절히 추가하여 목표 좌표에 맞추면 됩니다.
- 커다란 싸이클 (x-좌표와 y-좌표가 동시에 증가하는 싸이클) 은 최대 1개 사용해도 무방합니다.
  - 커다란 싸이클을 2개 추가하는 것은 각 간선을 왕복하는 싸이클 6개를 추가한 것과 동일한 좌표 변화량 및 넓이 변화량을 가지기 때문입니다.
- 커다란 싸이클 사용 여부를 결정했으면, x-좌표와 y-좌표를 분리해서 DP를 계산할 수 있습니다.
- 이에 따라 시간복잡도를 $\mathcal{O}(N+M)$까지 줄일 수 있습니다.

단순 경로와 추가한 단순 싸이클이 서로 이어지지 않는 경우를 처리하지 않으면 틀렸습니다를 받을 수 있습니다.
아래와 같이 예외처리 가능합니다.

- 경로에서 사용할 간선 목록을 지정합니다. (총 $\binom{6}{2}$ 가지)
- 간선 목록에 들어간 단순 싸이클만 이용해서 DP를 계산합니다.

예외처리를 하더라도 시간복잡도는 여전히 선형입니다.

C. 배고픈 무토를 위한 피자 만들기

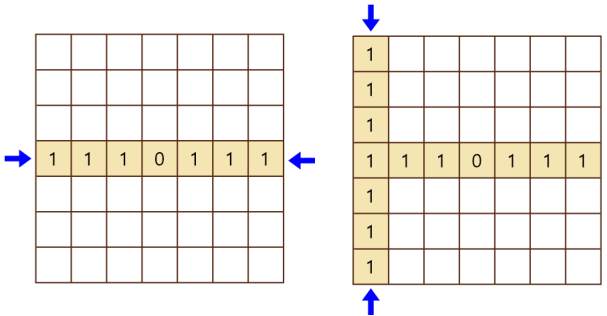
constructive

출제진 의도 – **Medium**

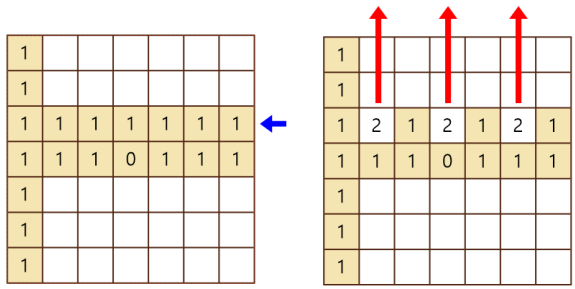
- 제출 115번, 정답 58팀 (정답률 50.435%)
- 처음 푼 팀: **대회에늦게신청할수록강한팀이다** (289를소수로착각한사람, 논문리젝당한사람, 소개원실조교를혼낸사람), 31분
- 출제자: golazcc83
- 가장 짧은 정해: 885B^{C++}, 1,004B^{Python}

– 원하는 배치를 만드는 방법은 여러 가지가 있지만 구현 난이도가 낮은 방법 중 하나를 소개합니다.

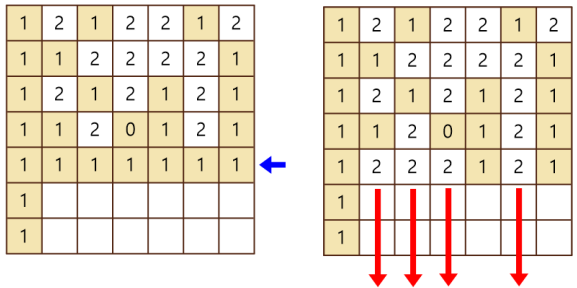
1. 처음 놓인 미트볼의 위치를 기준으로 모든 행을 채우고, 1 열을 채웁니다.



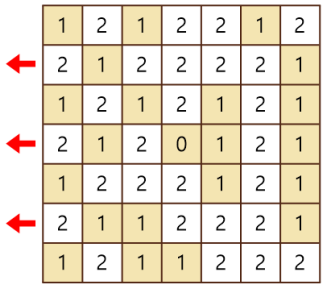
2. 처음 놓인 미트볼의 행 -1 부터 1 행까지에 대해 아래의 작업을 수행합니다.
   - 2 열부터 N 열까지 오른쪽 위치에서 미트볼을 모두 채웁니다.
   - 제거해야 하는 미트볼을 위쪽에서 꺼냅니다.



3. 처음 놓인 미트볼의 행부터 N행까지에 대해 아래의 작업을 수행합니다.
   - 2열부터 N열까지 오른쪽 위치에서 미트볼을 모두 채웁니다.
   - 제거해야 하는 미트볼을 아래쪽에서 꺼냅니다.



4. 1 열에서 제거해야 하는 미트볼을 왼쪽에서 꺼냅니다.



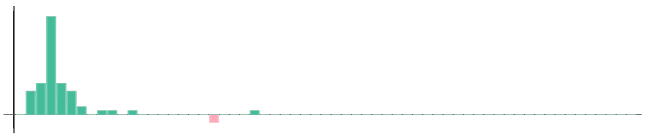
– 각 칸마다 최대 2 번의 염동력을 사용하여 모든 형태의 레시피대로 미트볼을 배치할 수 있습니다. 따라서 필요한 염동력 사용 횟수는 $2N^2$ 번 이하입니다.

D. SoleMap

`math, greedy`

출제진 의도 – **Easy**

- 제출 74번, 정답 59팀 (정답률 79.730%)
- 처음 푼 팀: **사상 첫 포핏** (얘네는 그냥, PS를, 잘함), 6분
- 출제자: `kipa00`
- 가장 짧은 정해: 449B^{C++}, 409B^{Python}



- u에서 v까지 매일 운행하는 차량의 대수가 주어지고 각 도로가 w차로를 갖고 있을 때, 도로 부담을 출력하는 문제입니다.
- 도로 부담은 각 도로를 운행하는 차량의 대수 c에 대해, c대를 w차로에 분배했을 때 각 차로의 운행 차량 수의 제곱의 합의 최솟값입니다.

- 먼저 각 도로에 다니는 차량의 대수를 구해 봅시다.
- 주어지는 u, v, x마다, $u \le \forall i < v$에 대해 $a_i \leftarrow a_i + x$를 시행해야 합니다.
- 시간이 너무 오래 걸리는 것 같습니다...

- 대신 **차분**(differential) $d_i := a_i - a_{i-1}$ 를 생각해 봅시다.
  - 편의상 $a_0 := 0$ 라 둡시다.
- 연산이 $d_u \leftarrow d_u + x$, $d_v \leftarrow d_v - x$ 의 상수 번으로 바뀌었습니다!
- $a_i = a_{i-1} + d_i$ 이므로, 이후 모든 a_i 를 계산하는 과정도 선형에 할 수 있습니다.
- 이와 같은 방법을 **imos법**(いもす法) 이라고 합니다.

- 이제 각 도로에 대해 c대의 차량이 다닌다는 것을 알았습니다.
- 이 도로가 w-차로일 때,

$$\min_{a_1+\cdots+a_w=c}\left(\sum_{i=1}^{w} a_i^2\right)$$

를 구해야 합니다.

- 잠시 고등학교 시절로 돌아가, "분산은 **제**곱의 **평**균 빼기 **평**균의 **제**곱"을 생각해 봅시다.
- 이를 $a_1, a_2, \cdots, a_w$에 적용하면 다음과 같습니다:

$$\begin{aligned}\mathrm{Var}(a_1, \cdots, a_w) &= \frac{1}{w} \cdot \sum_{i=1}^{w} a_i^2 - \left(\frac{1}{w} \cdot \sum_{i=1}^{w} a_i \right)^2 \\ &= \frac{1}{w} \cdot \sum_{i=1}^{w} a_i^2 - \left(\frac{c}{w} \right)^2\end{aligned}$$

- 우변에서는 빨간색을 최소화해야 하고 나머지는 전부 상수이므로, 좌변의 **분산**을 최소화해야 합니다.
- 분산을 최소화하려면 a_i들이 최대한 비슷한 값을 가져야 할 것 같습니다!

- 이 관찰을 바탕으로, 만일 $a_j - a_i \geq 2$ 라면 구하는 값이 최소가 될 수 없음을 보입시다.

$$\begin{aligned}(a_j - 1)^2 + (a_i + 1)^2 &= a_i^2 + a_j^2 - 2(a_j - a_i + 1) \\ &< a_i^2 + a_j^2 \quad (\because \text{파란색이 양수})\end{aligned}$$

- 따라서 $a_i \leftarrow a_i + 1$, $a_j \leftarrow a_j - 1$ 로 대체하는 것이 최소입니다.
- 즉 수열의 어떤 두 값을 가져오더라도 값이 1 보다 많이 차이가 날 수 없습니다.

- 이제 c를 w로 나누어 $c = wq + r$ $(0 \le r < w)$로 씁시다.
- 어떤 $a_i < q$라면 모든 $a_j \le q$이고, 따라서

$$\sum_{j=1}^{w} a_j \le (w-1)q + a_i < wq \le c$$

  라서 문제 조건에 모순입니다.
- 마찬가지로 어떤 $a_i > q+1$인 경우에도 모든 $a_j \ge q+1$임을 통해 모순임을 보일 수 있습니다.

- 모든 a_i에 대해 $q \le a_i \le q+1$입니다.
- 그런데 각 수열에 일단 q개를 나누어 주고 나면 남은 r개는 각 차로에 최대 하나씩밖에 나눌 수 없습니다.

  $\therefore$ 수열에는 값이 q인 항이 $(w-r)$개, $(q+1)$인 항이 r개 있습니다!
- 구하고자 하는 값은 $q^2(w-r)+(q+1)^2 r$입니다.
- 전체 시간복잡도는 $\mathcal{O}(N+M)$입니다.

E. 돌놓기게임

`game_theory, greedy`

출제진 의도 – **Medium**

- 제출 77번, 정답 58팀 (정답률 79.730%)
- 처음 푼 팀: **Coding Dol and Dool** (나는ps뭐하지, 나는ps돌덩입니다, 나는ps짐승입니다), 25분
- 출제자: `kclee2172`
- 가장 짧은 정해: 908B^{C++}, 854B^{Python}

- 만약 돌이 놓여있는 칸이 없을 경우 서로가 0점을 얻게 됩니다.
- 이미 돌이 놓여있는 칸은 점수가 정해져 있기 때문에 무시할 수 있습니다.
- 원형의 게임판을 빈 칸들로 이루어진 연속된 구간으로 나누어서 생각할 수 있습니다.
- 구간의 양 끝이 같은 색깔의 돌과 연결되어 있다면 해당 구간은 그 색깔의 돌을 가진 사람만이 돌을 놓을 수 있습니다.

E. 돌 놓기 게임

- 구간의 양 끝이 다른 색깔의 돌과 연결되어 있는 경우에 대해 생각합니다.
  - 구간의 길이가 짝수일 경우 상대가 해당 구간에 놓을 때만 자신도 놓는 방식으로 서로가 자신 쪽의 구간 절반을 가져갈 수 있습니다.
  - 구간의 길이가 홀수일 경우 가운데의 한 칸을 제외하고 자신 쪽의 구간 절반을 가져갈 수 있습니다. 그리고 홀수인 구간의 가운데 한 칸씩이 남게 됩니다.
- 길이가 홀수인 구간의 가운데 한 칸씩을 모아서 크기 순으로 정렬한 뒤, 홀수번째 칸을 철수한테, 짝수번째 칸을 영희한테 주는 것이 각자의 최선이 됩니다.
- 해당 사실의 증명은 총 빈 칸의 개수에 대한 수학적 귀납법으로 할 수 있습니다.

E. 돌 놓기 게임

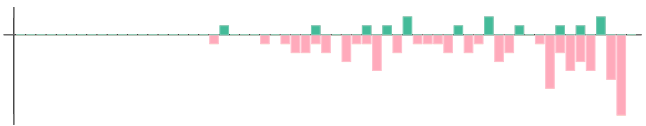
- 앞의 알고리즘의 증명은 다음과 같습니다.
- 먼저 후수가 앞의 알고리즘의 결과값 이상의 점수를 얻을 수 있음을 다음과 같이 증명할 수 있습니다.
- 길이가 짝수인 구간을 각각 한개의 그룹으로 놓고 길이가 홀수인 구간을 가운데 칸에 적힌 수가 큰 순서대로 두개씩 짝 지어서 그룹을 만듭시다. 점수가 0인 칸이 한개 존재하는 구간이 새로 생기어도 답이 변하지 않기 때문에 만약 길이가 홀수인 구간이 홀수개일 경우 이 구간을 추가하여 생각할 수 있습니다.
- 그 이후 후수는 선수가 돌을 놓은 구간에 돌을 놓고 만약 그 구간에 돌을 놓지 못 할 경우 같은 그룹에 있는 구간에 돌을 놓습니다.
- 이렇게 할 경우 후수는 길이가 짝수인 구간의 자신쪽 절반, 그리고 길이가 홀수인 구간의 자신쪽 절반과 가운데 칸중 각 그룹에서 1개씩 가져갈 수 있으며 이는 앞의 알고리즘의 답보다 크거나 같습니다.

- 선수의 경우 첫번째 수를 길이가 홀수인 구간 중 가운데 칸이 가장 큰 구간에 돌을 놓은 뒤에 자신이 후수인 것 처럼 전략을 세우면 앞의 알고리즘의 답 이상의 점수를 얻을 수 있습니다. (만약 길이가 홀수인 구간이 없다면 아무칸에 돌을 놓고 진행할 수 있습니다.)
- 두 명 모두 앞의 알고리즘으로 구해진 점수 이상을 보장할 수 있고 점수의 합이 고정되어 있기 때문에 앞의 알고리즘으로 구해진 점수가 정답이 됩니다.

F. 4색 정리

graph
출제진 의도 – **Medium**

- 제출 94번, 정답 14팀 (정답률 14.894%)
- 처음 푼 팀: **사상 첫 포핏** (얘네는 그냥, PS를, 잘함), 102분
- 출제자: qwerasdfzxcl
- 가장 짧은 정해: 1,725B^{C++}, 1,855B^{Python}



- 입력으로 주어지는 그래프를 G, 색을 정점으로 놓고 인접할 수 있는 서로 다른 두 색을 간선으로 이어서 만든 그래프를 H라고 합시다.
- 즉, H는 입력으로 들어온 K개의 순서쌍의 여집합에 대한 그래프라고 할 수 있습니다.

- Case 1: H에 간선이 존재하지 않을 경우
- 어떤 두 색도 인접할 수 없으므로 항상 불가능합니다.

- Case 2: H에 삼각형이 존재할 경우
- 일반성을 잃지 않고 $1, 2, 3$번 색이 삼각형을 이룬다고 합시다.
- G를 $1, 2, 3$번 색으로 3-coloring할 수 있으면 충분합니다.
- 이는 G의 조건에 의해 항상 가능합니다.

- G는 정n각형의 triangulation 형태의 그래프의 부분그래프입니다.
  - G에 간선을 추가해 적당한 triangluation으로 만듭니다.
  - 이 그래프에서 문제를 풀어서 원래의 G에 같은 색으로 칠하면 됩니다.
- 정n각형 내부에 있는 면으로 만든 dual graph는 트리입니다.
- 트리의 리프에 해당하는 면(삼각형)을 생각하면, 차수가 2인 정점 v가 그 위에 존재합니다.
- $G-v$를 3-coloring했다면, v는 차수가 2이므로 남은 색으로 칠해주면 됩니다.
- $G-v$도 같은 형태의 그래프이므로 이를 재귀적으로 하면 됩니다.

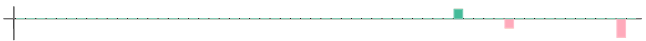
- Case 3: 나머지 경우
- H의 정점은 4개이고 삼각형이 없으므로 H는 이분그래프입니다.
- 해가 존재할 필요충분조건은 G가 이분그래프입니다.
- 증명은 생략합니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

dp, greedy

출제진 의도 – **Hard**

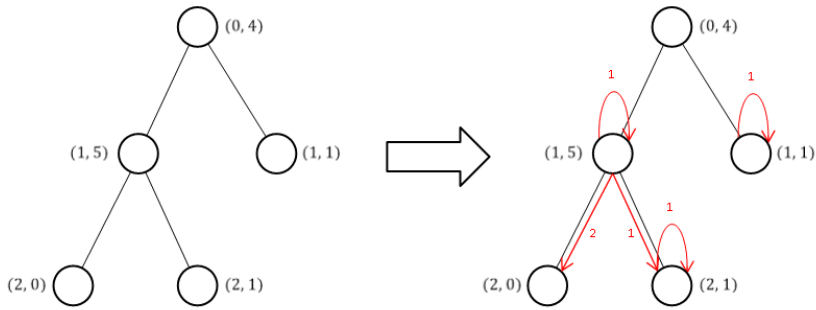
- 제출 6번, 정답 1팀 (정답률 16.667%)
- 처음 푼 팀: **사상 첫 포핏** (얘네는 그냥, PS를, 잘함), 218분
- 출제자: azberjibiou
- 가장 짧은 정해: 1,630B^{C++}



G. $\prod_{i=1}^{N}(R_i - L_i + 1)$개의 트리

- 고정된 c_i에 대해서, $\sum c_i a_i$를 최소화하기 전에 임의의 해를 dfs를 통해 만들어내는 방법을 알아봅시다.
- i번 정점의 i번을 제외한 서브트리 내부 모든 정점 j가 $S_j \geq q_j$를 만족한다고 가정합니다.
- $S_i \geq q_i$를 만족할 때까지 $a_j < p_j$를 만족하는 서브트리 내부 정점 j의 a_j를 1 증가시키는 시행을 반복합시다.
  - 이를 i에서 j를 **선택했다**고 하고, j가 i에게 **선택받았다**고 합시다.
  - 루트를 기준으로, 잎에서 루트 방향으로 $S_i \geq q_i$를 만족시켜 나가는 것입니다.
- $(c_1, c_2, \ldots, c_N)$과 무관하게 i번 정점에서 자신의 자손 j번 정점을 선택하는 횟수는 동일합니다. 이 횟수를 w_i로 정의합시다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리



각 정점에는 (a_i, S_i)가 적혀 있습니다.

i에서 j를 선택하면 $i \to j$로 화살표를 그립니다. 화살표에 쓰여 있는 수는 선택한 횟수입니다.

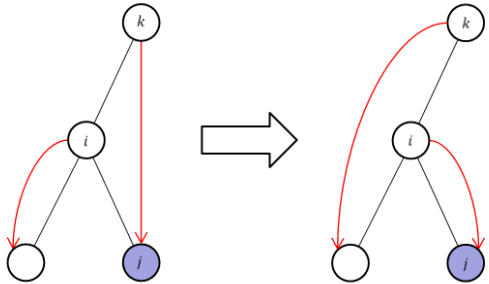
G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- 이제 $\sum c_i a_i$ 를 최소화해 봅시다.
- S_i 를 1 증가시키기 위해 a_j 를 1 늘리는 과정에서, c_j 가 최소인 j 를 선택할 수 있습니다.
- 이 전략은 exchange argument로 증명할 수 있습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- i 번 정점에서 c_j 가 최소인 j 를 선택하지 않았다고 합시다.
- 모든 선택 과정을 끝마친 뒤,
  - $a_j < p_j$ 를 만족한다면, a_j 를 1 증가시키고 i 에서 선택한 j' 번 정점의 $a_{j'}$ 을 1 감소시키면 더 나은 해가 나옵니다.
  - $a_j = p_j$ 를 만족한다면, j 를 선택한 i 의 조상 k 가 존재합니다. i 에서 선택한 j' 을 k 에서 선택한 j 와 바꿀 수 있으므로, c_j 가 최소인 j 를 선택하지 않을 이유가 없습니다.
    - 교환이 가능한 이유는 k 의 subtree가 i 의 subtree를 포함하기 때문입니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리



파란 정점은 c_j 가 최소인 정점입니다.

편의상 조상/자손 관계인 정점들을 간선으로 이었습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- 이제 모든 i에 대해 $1 \le L_i \le R_i \le 2$인 경우에 문제를 풀어 봅시다.
- i번 정점에서 $S_i \ge q_i$를 만족시키도록 i에서 i의 자손 j를 선택하는 시행을 마쳤다고 합시다.
- i번 서브트리에서 각 $r = 1, 2$에 대해서 $c_j = r$을 만족하는 모든 정점 j가 선택받을 수 있는 횟수의 총합이 중요합니다.
  - r의 가짓수가 2가지이기 때문에 $r = 1$에 대해서만 구해도 괜찮습니다.
- i의 서브트리 내부 정점을 $l_1, l_2, \cdots, l_m$라고 합시다.
- 모든 $(c_{l_1}, c_{l_2}, \cdots, c_{l_m})$에 대해 $c_j = 1$을 만족하는 j를 선택할 수 있는 횟수의 총합이 k인 $(c_{l_1}, c_{l_2}, \cdots, c_{l_m})$의 경우의 수를 $\mathrm{dp}[i][k]$로 정의합시다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

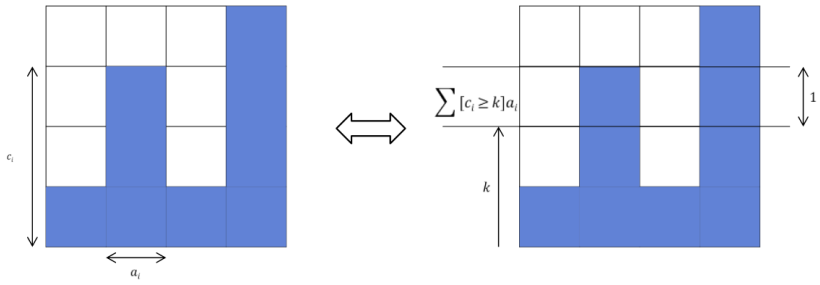
- i에서 i의 자손들을 선택하기 전 $c_j = 1$을 만족하는 자손 j를 선택할 수 있는 횟수의 총합이 k인 $(c_{l_1}, c_{l_2}, \ldots, c_{l_m})$의 경우의 수를 $\mathrm{dp}'[i][k]$로 정의합시다.
- $\mathrm{dp}[i][0] = \sum_{k \le w_i} \mathrm{dp}'[i][k]$이고, $k \ge 1$에 대해 $\mathrm{dp}[i][k] = \mathrm{dp}'[i][k + w_i]$입니다.
- 따라서 $\mathrm{dp}'[i][k]$를 모든 k에 대해서 구하는 것은 곧 $\mathrm{dp}[i][k]$를 모든 k에 대해서 구하는 것과 같습니다.

- i의 자식들을 $h_1, h_2, \ldots, h_{d_i}$ 라고 하면, $\mathrm{dp}'[i][k] = \sum_{\sum k_j = k} \left(\prod_{j=1}^{d_i} \mathrm{dp}[h_j][k_j] \right)$를 만족합니다.
- 이러한 형태의 DP 점화식은 두 서브트리의 DP 배열을 합치는 것을 반복하여 풀 수 있습니다.
- 크기가 A, B인 두 DP 배열을 합치는데 $\mathcal{O}(AB)$의 시간이 필요합니다.
- 각 정점의 DP 배열 크기는 자신의 서브트리 내부 p 값의 합보다 작거나 같습니다.
- tree optimization (a.k.a. 검은 돌 트릭)을 사용하면 $\mathcal{O}\left((\sum p_i)^2\right)$에 DP 값을 모두 구할 수 있습니다.
- 자세한 내용은 `https://cubelover.tistory.com/31`에서 확인할 수 있습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- c_i 값이 2개인 경우에 해당 문제를 풀었습니다. 이를 활용할 수 있는 방법을 찾아봅시다.
- $\sum_i c_i a_i = \sum_i \sum_{k=1}^{\infty} [c_i \geq k]_{\mathbf{1}}\, a_i = \sum_{k=1}^{\infty} \sum_i [c_i \geq k]_{\mathbf{1}}\, a_i$ 입니다.
- $c_i \leq 250$ 이므로, 모든 $1 \leq k \leq 250$ 인 k 에 대해서 $\sum_i [c_i \geq k]_{\mathbf{1}}\, a_i$ 를 구하면 됩니다.
- 이는 각 k 에 대해서 $c_i \geq k$ 인 정점 i 가 선택되는 횟수의 합을 구하는 것과 같습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$개의 트리



각 직사각형들은 너비가 a_i, 높이가 c_i 이다. 전체 파란 영역의 넓이는 $\sum_i c_i a_i$ 이다

k 층에 있는 파란 영역의 넓이는 $\sum_i [c_i \geq k] a_i$ 이다

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$개의 트리

- k와 $(c_1, c_2, \ldots, c_N)$을 고정한 상태로 위의 그리디한 전략을 다시 살펴 봅시다.
- 각 정점은 자손 j 중 $c_j < k$를 만족하는 정점들을 먼저 선택한 이후에 $c_j \geq k$를 만족하는 정점들을 선택합니다.
- 따라서 서브트리 내부에 $c_j < k$를 만족하는 모든 정점 j에 대해서 선택받을 수 있는 횟수의 총합을 알고 있으면 $c_j < k$를 만족하는 정점이 몇 번 선택받는지 알 수 있습니다.
- 이는 $c_j < k$인 정점들의 c_j를 0, $c_j \geq k$인 정점들의 c_j를 1로 정의한 경우의 그리디 알고리즘이 작동하는 형태와 같습니다.
- 이를 활용하면 $\sum_i [c_j \geq k]_{\mathbf{1}} a_i$를 쉽게 구할 수 있습니다.
- 정리하자면, 최적 $(a_1, a_2, \ldots, a_n)$의 $\sum_i [c_j \geq k]_{\mathbf{1}} a_i$는 $c_j := [c_j \geq k]$로 정의한 문제의 답과 같습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$개의 트리

- 이제 원래 문제로 돌아와서, $1 \le k \le 250$을 만족하는 모든 k에 대해 $\sum_i [c_i \ge k]_1 \, a_i$를 구합시다.
- 다음과 같이 부분문제를 정의합니다:

$$\mathrm{dp}[i][k][x] = (i\text{번 정점의 서브트리에 대해서 그리디 알고리즘을 모두 시행했을 때 서브트리 내부 정점 } j \text{ 중 } c_j < k\text{를 만족하는 } j\text{를 선택할 수 있는 횟수가 } x\text{회인 경우의 수})$$

- $1 \le L_i \le R_i \le 2$를 만족하는 경우의 부분문제 정의와 유사함을 확인할 수 있습니다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- 각 k에 대해서 시간 복잡도 $\mathcal{O}\left(C(\sum p_i)^2\right)$에 해당 문제를 해결할 수 있습니다.
- 따라서 c_i의 최댓값을 C라고 했을 때, 최종적인 시간 복잡도는 $\mathcal{O}\left(C(\sum p_i)^2\right)$입니다.
- 다음 슬라이드에 비슷한 문제들을 추천합니다. 보고 싶지 않다면 빠르게 넘깁시다.

G. $\prod_{i=1}^{N}(R_i - L_i + 1)$ 개의 트리

- 이와 비슷한 문제로는 Atcoder의 median pyramic hard와 바보 자물쇠가 있습니다.
- 모두 재미있는 문제들이니 한번 풀어 봅시다!

H. 감옥

`geometry`, `binary_search`

출제진 의도 – **Medium**

- 제출 123번, 정답 17팀 (정답률 13.821%)
- 처음 푼 팀: **성호 없는 성호팀** (성호야, 어디, 갔어), 111분
- 출제자: gs22123
- 가장 짧은 정해: 2,026B[C++]

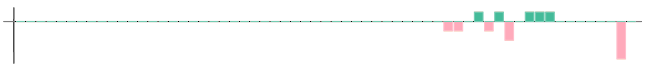
- 주어지는 다각형의 꼭짓점들은 원점 O를 기준으로 각도 정렬이 되어 있다고 생각할 수 있습니다.
- 쿼리로 주어지는 점에 대해서 이분 탐색을 통해 위 정렬에서 어느 두 꼭짓점 사이에 위치하게 되는지 구할 수 있습니다.
- 어느 두 꼭짓점 사이에 위치해 있는지를 알았으면 이후에 세 꼭짓점의 CCW를 구해서 점이 다각형 내부에 있는지, 또는 외부에 있는지를 판별할 수 있습니다.
- 총 $\mathcal{O}\left(N + Q \log N\right)$에 문제를 해결할 수 있습니다.

I. 러시안 회전초밥

`suffix_tree`, `dfs`, `prefix_sum`, `binary_search`

출제진 의도 – **Hard**

- 제출 14번, 정답 5팀 (정답률 35.714%)
- 처음 푼 팀: **NewTrend** (Karuna, blackking26, arnold518), 229분
- 출제자: queued_q
- 가장 짧은 정해: 3,190B[C++]



느린 풀이 — $\mathcal{O}\left(N^3\right)$

진우는 모든 가능성에 대해 필요한 스킵 횟수 중 최댓값만큼 스킵권을 구매해야 합니다.

- N개의 가능한 시작 초밥에 대해 각각 시뮬레이션을 진행합시다.
  - 각 시뮬레이션은 독립적인 N개의 **평행우주**에 비유할 수 있습니다.
- 시뮬레이션 속의 진우는 지금까지 마주친 초밥이 각각 와사비 초밥인지 아닌지를 알 수 있습니다.
  - 이것을 해당 평행우주의 **관측 정보**라고 부릅시다.
- 시뮬레이션 속의 진우는 자신이 속한 평행우주가 무엇인지 모르므로, 지금까지의 관측 정보와 일관되는 평행우주의 집합 X를 관리할 것입니다. 맨 처음에는 N개의 모든 평행우주가 담겨 있습니다.

시뮬레이션 과정은 다음과 같습니다. 진우가 지금까지 i 개의 초밥을 마주쳤다고 합시다.

- 만약 K 개의 초밥을 먹었다면 시뮬레이션을 종료합니다. 그렇지 않다면 $i+1$ 번째 초밥을 먹거나 스킵할 차례입니다.
- 집합 X 안의 어떤 평행우주에서도 $i+1$ 번째 초밥에 와사비가 없다면, 초밥을 먹어도 안전합니다.
- 그렇지 않다면 눈앞의 초밥에 와사비가 들었을 가능성이 있으므로 스킵합니다.
  - 초밥을 스킵하고 나면 그 초밥에 와사비가 들었는지 알 수 있습니다. 이 정보와 일관되지 않는 평행우주를 집합 X 에서 지웁니다.

- 각 평행우주의 시뮬레이션이 끝나면 스킵한 횟수를 세고, 그중 최댓값이 답이 됩니다.
  - 단, N개의 초밥을 지나는 동안 K개의 초밥을 먹지 못하는 평행우주가 존재한다면 -1을 출력합니다.
- 시간 복잡도 분석은 다음과 같습니다.
  - $i+1$번째 초밥을 먹을지 결정하는 단계에는 $\mathcal{O}(|X|) = \mathcal{O}(N)$의 시간이 듭니다.
  - i를 증가시키면서 위 과정을 진행해야 하므로, 한 번의 시뮬레이션에 $\mathcal{O}\left(N^2\right)$의 시간이 듭니다.
  - 가능한 모든 시작 초밥에 대해 시뮬레이션을 수행해야 하므로 총 $\mathcal{O}\left(N^3\right)$의 시간이 듭니다.

개선된 풀이 — $\mathcal{O}\left(N^2\right)$

- **관찰 1.** 관측 정보는 주어진 원형 문자열 C의 부분 문자열로 나타낼 수 있습니다.
- **관찰 2.** 두 평행우주에서 어떤 시점까지의 관측 정보가 동일하면, 그때까지의 시뮬레이션 과정은 일치합니다. 이러한 평행우주들을 하나로 묶어서 시뮬레이션할 수 있습니다.

원형 문자열 C의 cyclic shift를 담은 trie를 구성합시다.

- Trie에서 문자열 S에 대응되는 정점을 찾으면, 이는 어떤 시점까지의 관측 정보가 S인 평행우주 묶음을 나타냅니다.
- 다음 초밥이 와사비 초밥인지 아닌지에 따라 평행우주 묶음은 둘로 나뉩니다.

- Trie를 DFS 탐색하며 시뮬레이션을 진행합니다.
  - 자식 정점 중 와사비 초밥이 존재한다면, 다음 초밥에 와사비가 들었을 가능성이 있으므로 스킵합니다.
  - 그렇지 않다면 초밥을 먹어도 안전합니다.
  - 자식 정점을 재귀적으로 탐색하면서, 먹은 초밥의 개수가 K개가 되면 스킵 횟수를 정답에 반영한 뒤 되돌아갑니다.
- Trie의 정점 개수는 $\mathcal{O}\left(N^2\right)$이고, 다음 초밥을 먹을지 결정하는 단계에 $\mathcal{O}\left(1\right)$만이 소요되므로 시간 복잡도는 $\mathcal{O}\left(N^2\right)$입니다.

빠른 풀이 — $\mathcal{O}(N \log N)$

- Trie에서 갈라지는 가지가 없는 경로를 압축하면 더 효율적인 자료 구조인 suffix tree가 됩니다.
- Ukkonen's algorithm을 이용하면 suffix tree를 $\mathcal{O}(N)$에 구성할 수 있지만, 해당 알고리즘은 이해와 구현이 까다롭습니다.
- 대신 suffix array와 LCP array를 이용해 suffix tree의 깊이 우선 탐색을 시뮬레이션할 수 있습니다.

- 관측 정보가 S로 일치하는 평행우주 묶음은 S를 prefix로 갖는 cyclic shift의 집합입니다.
- 이는 suffix array 상에서 구간으로 나타낼 수 있습니다.
- LCP array에 구간 최솟값 쿼리를 수행해서 구간의 LCP (Longest Common Prefix) 를 구하면,
  - 해당 길이까지는 구간 내 prefix가 일치하다가
  - 문자 하나가 추가되는 순간 prefix가 일치하는 구간이 둘로 갈라집니다.
- 이러한 구간들 사이의 포함 관계는 suffix tree의 구조와 동치이므로, 분할 정복을 수행하면 suffix tree를 깊이 우선 탐색하는 것과 동일합니다.

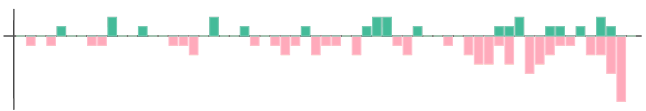
- Suffix tree에서 압축한 경로를 지날 때에는, 누적 합을 이용해서 추가로 먹을 초밥과 스킵할 초밥의 개수를 빠르게 구할 수 있습니다.
- 만약 경로를 지나는 도중에 K개 이상의 초밥을 먹게 된다면, 이분 탐색을 통해 처음으로 K개의 초밥을 먹게 되는 시점을 구할 수 있습니다.
- 구현에 따라 $\mathcal{O}\left(N \log N\right)$ 또는 $\mathcal{O}\left(N \log^2 N\right)$ 등의 시간 복잡도에 문제를 해결할 수 있습니다.

J. Distance Sum Maximization

`trees`

출제진 의도 – **Medium**

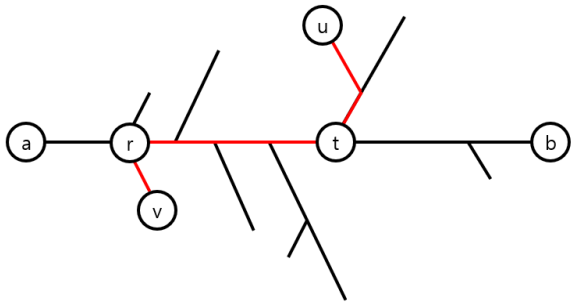
- 제출 104번, 정답 24팀 (정답률 23.077%)
- 처음 푼 팀: **사상 첫 포핏** (얘네는 그냥, PS를, 잘함), 24분
- 출제자: `cki86201`
- 가장 짧은 정해: 883B^{C++}, 1,758B^{Python}



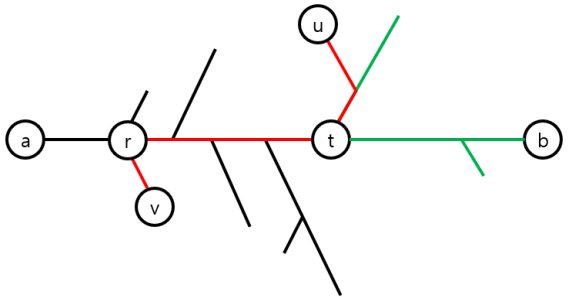
- 쿼리로 u, v가 주어질 때 $\mathrm{dist}(u, x) + \mathrm{dist}(v, x)$의 최댓값을 구하는 문제입니다.
- 트리의 지름을 하나 찾아서 양 끝점을 a, b라 합시다.
- $\mathrm{dist}(a, x) + \mathrm{dist}(b, x)$가 가장 큰 x를 c라 합시다.
- u, v에 상관없이 $x = a$, b, c만 고려하면 됩니다.
- 왜 그럴까요?

- $\mathrm{dist}(u, x) + \mathrm{dist}(v, x) = \mathrm{dist}(u, v) + 2 \times \mathrm{dist}(u \sim v, x)$ 입니다.
- 이 때 $\mathrm{dist}(u \sim v, x)$ 는 (u와 v를 잇는 경로 중 x와 가장 가까운 정점)과 x의 거리입니다.
- $\mathrm{dist}(u, v)$ 는 x와 무관하기 때문에, 경로 $u \sim v$에서 가장 먼 점을 찾으면 됩니다.

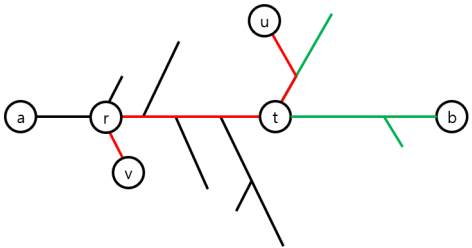
- 아래와 같이 u, v 쿼리가 들어왔다고 가정합니다.
- 지름과 만나는 점을 t, r 이라고 합시다.



– 초록색 영역의 경우, $x = b$만 고려하면 해결됩니다.

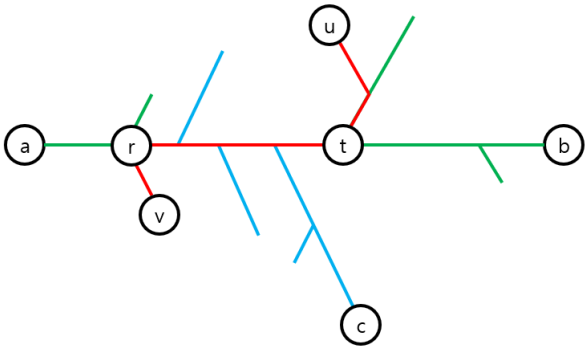


- 이 영역에서 임의의 정점 x에 대해 $\mathrm{dist}(u \sim v, x) \le \mathrm{dist}(t, x)$ 입니다.
- $\mathrm{dist}(u \sim v, b) < \mathrm{dist}(u \sim v, x)$ 으로 가정하면 $\mathrm{dist}(t, b) < \mathrm{dist}(t, x)$ 가 되고,
- $\mathrm{dist}(a, b) < \mathrm{dist}(a, x)$ 가 되어서 a, b가 지름의 두 끝점이라는 가정에 모순입니다.



J. Distance Sum Maximization

- 양쪽 초록색 영역은 $x = a, x = b$로 고려할 수 있습니다.
- 파란색 부분, 즉 r과 t 사이에서 분기하는 정점들만 잘 고려하면 됩니다.



- $\mathrm{dist}(a, b) = C$라 할 때, 다음과 같이 $X_0, X_1, \cdots, X_C$를 정의합니다.
- X_i: $\mathrm{dist}(a, z) = i$ 인 $a \sim b$ 경로상의 정점 z에 대해, z에서 $a \sim b$ 경로의 간선을 하나도 거치지 않고 갈 수 있는 가장 멀리 떨어진 정점까지 거리
- $\mathrm{dist}(a, r) = l, \mathrm{dist}(a, t) = r$ 이라 하면, $\mathrm{dist}(u, v) + 2 \times \max_{l \le m \le r} X_m$ 을 구하면 됩니다.

- 그런데 사실 $\max_{l \le m \le r} X_m$ 대신 $\max_{0 \le m \le C} X_m$ 을 구해도 됩니다.
- 추가로 고려되는 값들 $(X_0, \cdots, X_{l-1}, X_{r+1}, ..., X_C)$ 는 위에서 고려한 초록색 영역으로부터 구해지는 답보다 작기 때문입니다. 즉 답에 영향을 주지 않습니다.
- $\max_{0 \le m \le C} X_m$ 는 a와 b 사이 경로에서 가장 먼 점 c를 구하면 됩니다.

- 아래 방법으로 LCA 등을 사용하지 않고 BFS(DFS)를 4번 사용하여 해결할 수 있습니다.
  1. 1번 정점에서 BFS 수행 후 가장 먼 점을 a라 합니다.
  2. a번 정점에서 BFS를 수행하여 $\mathrm{dist}(a, x)$를 모두 구하고, 이 값이 가장 큰 정점을 b라 합니다.
  3. b번 정점에서 BFS를 수행하여 $\mathrm{dist}(b, x)$를 모두 구하고, $\mathrm{dist}(a, x) + \mathrm{dist}(b, x)$가 가장 큰 x를 c라 합니다.
  4. c에서 BFS를 수행하여 $\mathrm{dist}(c, x)$를 구합니다.
- 모든 x에 대해 $\mathrm{dist}(a, x)$, $\mathrm{dist}(b, x)$, $\mathrm{dist}(c, x)$를 알고 있으므로 쿼리를 $\mathcal{O}(1)$에 처리할 수 있습니다.
- 시간복잡도는 $\mathcal{O}(N + Q)$입니다.

K. 스레드

`combinatorics`, `fft`, `generating_function`

출제진 의도 – **Hard**

- 제출 8번, 정답 2팀 (정답률 25.000%)
- 처음 푼 팀: **NewTrend** (Karuna, blackking26, arnold518), 266분
- 출제자: `jh05013`
- 가장 짧은 정해: 2,887B[C++]

X의 값에 실질적인 기여를 하는 스레드는 다음과 같습니다. 이들을 "중요한 스레드"라고 부릅시다.

- X에 마지막으로 값을 쓴 스레드 I_1
- I_1 이 읽은 값을 쓴 스레드 I_2
- I_2 가 읽은 값을 쓴 스레드 I_3

⋮

- X의 최종 값이 k라면 중요한 스레드 역시 k개입니다.
- 일반성을 잃지 않고 $I_1, \cdots, I_k$가 순서대로 스레드 $k, \cdots,$ 스레드 1인 스레드 실행의 경우의 수를 구합시다. 그 값에 $\binom{N}{k} k!$를 곱하면 됩니다.

- 우선 중요한 스레드 $1, 2, \cdots, k$의 읽기와 쓰기를 순서대로 나열합니다.
- 이 $2k$ 개의 연산 전후로 W0, R0, W1, $\cdots$, Wk 라는 표식을 붙입시다.
- 그다음 "잉여" 스레드 $k+1, \cdots, N$의 읽기와 쓰기를 적절히 배치하되...
  1. 잉여 스레드의 읽기는 그 스레드의 쓰기보다 먼저 와야 합니다.
  2. 잉여 스레드의 쓰기는 R.. 에 와야 합니다. 특히 Wk에는 아무것도 올 수 없습니다.
  3. 일반성을 잃지 않고, 잉여 스레드 $i+1$의 쓰기는 잉여 스레드 i의 쓰기보다 먼저 와야 한다고 합시다. 즉 잉여 스레드는 쓰는 시점에 대해 정렬되어 있습니다. 맨 마지막에 $(N-k)!$을 곱하면 됩니다.

- 잉여 스레드 $k+i-1$ 까지 읽기와 쓰기를 배치한 상태에서, $k+i$ 의 읽기와 쓰기를 배치해 봅시다.
- 쓰기를 특정 Rz에 배치합니다. 이때 잉여 스레드 $k+1, \cdots, k+i-1$ 의 읽기와 쓰기는 Rz 또는 그 이전에 와야 하며, $k+i$ 의 쓰기는 Rz의 맨 뒤에 배치해야 합니다.
- 읽기를 배치합니다.

  $k+i$ 의 쓰기를 Rz에 배치했을 때,
- 그보다 전에 있는 중요한 스레드의 연산은 $2z+1$ 개입니다.
- 그보다 전에 있는 잉여 스레드의 연산은 $2i-2$ 개입니다.
- 따라서 읽기를 배치하는 경우의 수는 $2(i+z)$ 입니다.

- 따라서 답은 0 이상 $k-1$ 이하의 정수로 이루어진 모든 단조증가수열 $z_1, \cdots, z_{N-k}$ 에 대해 $\prod_{i=1}^{N-k} 2(i+z_i)$ 의 합입니다.
- 이제 z_i 를 $y_i := i + z_i$ 로 대체하면, $y_1, \cdots, y_{N-k}$ 는 0 이상 $N-1$ 이하의 정수로 이루어진 강한 증가수열이 됩니다.
- 위 변환은 두 종류의 수열 사이 일대일 대응이므로, 모든 y 에 대해 $\prod_{i=1}^{N-k} 2y_i$ 를 합하면 됩니다.
- 결국 우리가 풀어야 되는 문제는 $A_k := \sum_{y} \prod_{i=1}^{k} 2y_i$ 라고 할 때 $A_0, \cdots, A_N$ 을 모두 구하는 것입니다.

- A의 생성함수 $A_N x^N + \cdots + A_1 x + A_0 = (2x+1)(4x+1)\cdots(2(N-1)x+1)$ 입니다. 일차항을 k개 선택했을 때 길이 k의 강한 증가수열이 나옴과 동시에 $\left(\prod_{i=1}^{k} 2y_i\right) x^k$ 가 더해짐을 확인해 보세요.
- 일차식 N개의 곱은 분할 정복과 FFT로 $\mathcal{O}\left(N \log^2 N\right)$에 계산할 수 있습니다.
  - 출제자가 좋아하는 구현은 분할 정복 대신 큐를 사용하여, 큐에서 두 번 뽑은 다음 그 곱을 다시 삽입하는 작업을 반복하는 것입니다.
  - Polynomial shift로 $\mathcal{O}\left(N \log N\right)$에도 풀 수 있습니다.

L. 밤양갱

UCPC 2024

`greedy`, `bipartite_matching`

출제진 의도 – **Medium**

- 제출 179번, 정답 41팀 (정답률 33.621%)
- 처음 푼 팀: **개척단 훈련소** (파괴공작, 무장강도단 고용, 해커), 22분
- 출제자: `sorohue`
- 가장 짧은 정해: 1,167B^{C++}, 1,755B^{Python}

- 만들어야 하는 밤양갱 조각의 수가 i 개일 때의 답을 A_i 라고 합시다.
- $1 \le x < y \le N^2/2$ 를 만족하는 임의의 두 정수 x, y 를 잡습니다.
- 등급의 최댓값이 A_y 이도록 y 개의 조각을 만들었다면, 그중 등급이 A_y 미만인 조각은 $y-1$ 개 존재합니다.
- 이때 $x \le y-1$ 이므로, 등급의 최댓값이 A_y 미만이도록 x 개의 조각을 고르는 방법이 반드시 존재합니다.
- 따라서 $x < y$ 이면 $A_x < A_y$ 입니다.

- 등급의 최댓값이 A_k 이도록 k 개의 조각을 만들었다면, 그 k 개의 조각을 만드는 데에 사용된 모든 칸의 등급은 A_k 이하입니다.
- 따라서 **등급이 작은 칸부터 순서대로** 그래프에 추가하면서 만들 수 있는 최대 조각 수를 세는 것으로 문제를 해결할 수 있습니다.

- 등급이 S 이하인 칸만을 이용해 만들 수 있는 최대 조각 수를 M_S 라고 합시다.
- 등급이 $S+1$ 이하인 칸만을 이용해 조각을 만드는 경우, 등급이 S 이하인 칸을 조각을 만드는 데에 사용할 수 있으므로 적어도 M_S 개의 조각을 만들 수 있습니다.
- 거꾸로 등급이 $S+1$ 이하인 칸만을 이용해 M_{S+1} 개의 조각을 만들었다고 하면, 여기서 등급이 $S+1$ 인 칸을 제거하고 남는 밤양갱 조각의 수는 등급이 $S+1$ 인 칸을 포함하는 조각이 있는 경우 하나 줄어들고 그렇지 않은 경우 조각의 수는 줄어들지 않습니다.
- 이로부터 S 를 늘려감에 따라 만들 수 있는 조각의 수는 그대로이거나 1 늘어남을 알 수 있습니다.

- 만약 칸을 추가했더니 만들 수 있는 조각의 수가 늘어났다면, 기존의 조각 수를 유지한 채로 새로 추가한 칸을 포함하는 새로운 조각을 만들 수 있어야 합니다.
- 새로운 칸을 포함하는 조각을 잡았는데 다른 조각과 겹친다면, 기존에 있던 조각을 분리해 줍시다.
- 새로운 칸이 추가되지 않은 상황에서, 남은 칸으로 더이상 만들 수 있는 조각은 없습니다.

- 따라서 분리한 뒤 남은 칸으로 새로운 조각을 만들어내야 총 조각의 수가 늘어납니다.
- 이와 같이 기존에 있는 조각을 옮기는 과정을 반복해 하나의 칸이 추가되었을 때 조각의 수를 늘릴 수 있을지를 알아낼 수 있습니다.
- 밤양갱의 각 칸을 정점으로 두고 인접한 두 칸에 해당하는 정점을 간선으로 잇는 그래프를 생각해 보면, 해당 그래프가 **이분 그래프**임을 알 수 있습니다.
- 이제 문제를 정점이 추가될 때마다 해당 정점을 포함하도록 새로운 매칭을 잡을 수 있는지 판정하는 문제로 바꿀 수 있습니다.

- DFS를 적용한 이분 매칭 알고리즘을 구현하면 $\mathcal{O}\left(N^2\right)$ 만에 하나의 칸을 추가했을 때의 답을 구할 수 있습니다.
- 칸의 수는 총 N^2 개이므로, $\mathcal{O}\left(N^4\right)$ 에 문제를 해결할 수 있습니다.

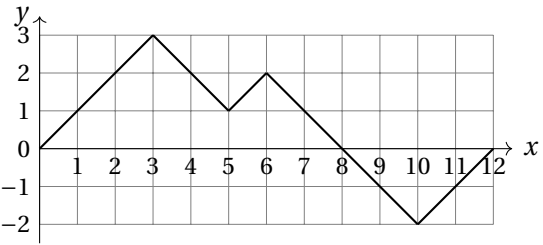
M. 지루함 줄이기

ad_hoc

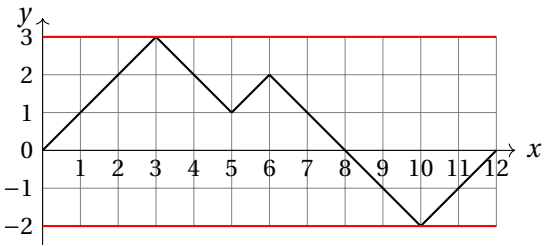
출제진 의도 – **Medium**

- 제출 116번, 정답 39팀 (정답률 33.621%)
- 처음 푼 팀: **NewTrend** (Karuna, blackking26, arnold518), 25분
- 출제자: exqt
- 가장 짧은 정해: 640B^{C++}, 596B^{Python}

관찰 1: 주어진 문자열을 시각적으로 표현해 봅시다. 2차원 좌표계에서 $(0, 0)$에서 출발하여 문자 `0`을 읽으면 $(+1, +1)$만큼 이동하고 `1`을 읽으면 $(+1, -1)$만큼 이동해 봅시다.



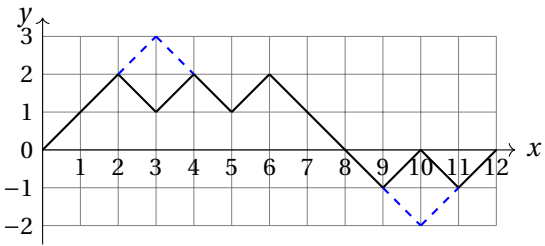
입력이 `000110111100`인 경우



- 그래프에서 최대값과 최소값의 차이가 곧 최종적으로 느끼는 지루함이 됩니다.
- 예제에서 최대값은 3이고 최소값은 -2, 지루함은 $|3-(-2)|=5$가 됩니다.

관찰 2: 그래프에서 인접한 '0'과 '1'을 바꾸는 과정을 수행했을 때 그래프가 어떻게 변할까요?

예를 들어 000110111100에서 3번째와 4번째 문자 위치를 바꾸고 10번째와 11번째 문자를 바꾸면 아래와 같이 변합니다.



이는 볼록한 부분을 반대 방향으로 2만큼 이동시킨다고 생각할 수 있습니다. 이제 그래프에서 볼록한 부분을 잘 이동시켜 최대-최소 차이가 K 이하가 되도록 하는 문제로 바뀌었습니다.

- 최대-최소 차이가 K 만큼 나는 경우는 총 $K+1$ 개가 있습니다.
  - 예를 들어 $K=3$ 이면 $(3,0)$, $(2,-1)$, $(1,-2)$, $(0,-3)$ 입니다.
- 모든 경우의 수에 대해서 빠르게 계산할 수 있는 방법을 생각해 봅시다.
- $Y>0$ 인 영역에서 최대값이 i 이하가 되도록 할 때 걸리는 시간을 U_i, $Y<0$ 인 영역에서 최소값이 $-i$ 가 되도록 할 때 걸리는 시간을 L_i 라고 합시다.
- U_i 는 제일 위에서부터 볼록한 부분을 하나씩 이동시켜 $\mathcal{O}(N)$ 에 구할 수 있고, L_i 도 유사하게 구할 수 있습니다.
- 문제의 정답은 $\min_{i+j=K}(U_i+L_j)$ 가 됩니다.