UOSPC 2024 Editorial

서울시립대 알고리즘 소모임 AI林

문제목록

	Open Contest				
번호	**문제 제목**	**의도한 난이도**	**번호**	**문제 제목**	**의도한 난이도**
A	UOS 문자열	Easy	G	이상한 시행	Hard
B	슈팅 연습	Easy	H	타임머신	Hard
C	자석 놀이	Medium	I	U, Our Star!	Hard
D	드론 조작	Medium	J	Counting Portal	Hard
E	일하기 시러	Hard	K	블록 쌓기	Hard
F	풍성한 트리	Hard	L	동전 뒤집기	Hard

A. UOS 문자열

- 종류 – string, math
- 출제진 의도 – Easy

- 제출 114회 / 맞은 사람 82명 (정답률 77.193%)
- First Solve: seawon0808, 0분
- 출제자: qqq831347

UOS 문자열

- UOS가 무한히 반복되어 나타나는 문자열의 x번째 문자를 구하는 문제입니다.
- x가 너무 크기 때문에 직접 문자열을 만들어 보는것은 불가능합니다.
- 3글자마다 반복되기 때문에 x%3번째 문자를 구하는것과 같고 이를 이용하면 해결할 수 있습니다.

B. 슈팅 연습

- 종류 – implement, simulation
- 출제진 의도 – Easy

- 제출 69회 / 맞은 사람 44명 (정답률 63.768%)
- First Solve: seawon0808, 4분
- 출제자: ljwljw8541

슈팅 연습

- 문제의 설명대로 구현하면 되는 시뮬레이션 문제입니다.
- 현재 커서에와 화면에 나타나 있는 과녁들과의 거리를 계산하여 가장 먼 과녁을 찾습니다.
- 가장 먼 과녁을 없애 점수를 얻고, 새 과녁을 추가합니다.
- 이를 M번 반복한 결과가 정답이 됩니다.

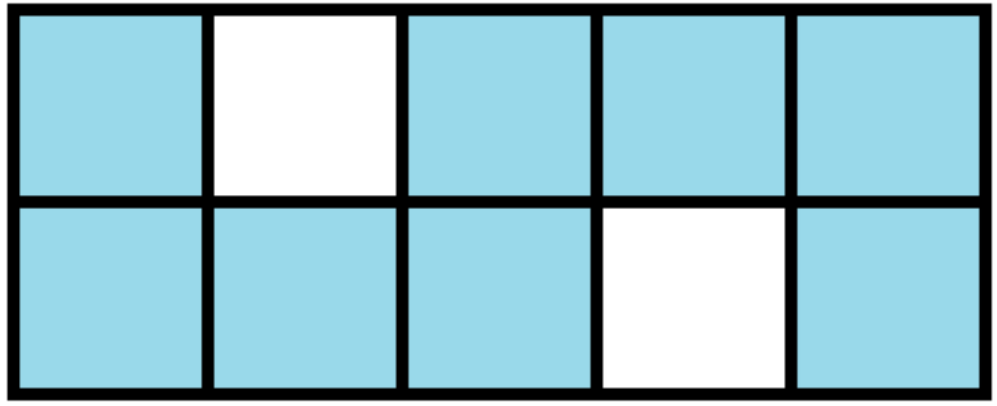
C. 자석 놀이

- 종류 – dp
- 출제진 의도 – Medium

- 제출 66회 / 맞은 사람 21명 (정답률 31.818%)
- First Solve: jeoffrey0522, 8분
- 출제자: qqq831347

자석 놀이

- 한번 이상 지나간 칸들의 수들의 합을 최대화 하는 문제입니다.
- 이 때 그런 칸들의 형태는 맨 왼쪽 위 칸과 맨 오른쪽 아래 칸을 포함하며 상하좌우로 인접한 형태가 됩니다.



자석 놀이

- 각 열만 보면 1행만 포함된 경우, 2행만 포함된 경우, 두 행 모두 포함된 경우로 나눌 수 있습니다.

- 이 때 dp[i][0], dp[i][1], dp[i][2]를 i열까지 봤으며 i열의 1행만 포함된 경우, 2행만 포함된 경우, 두 행 모두 포함된 경우의 최댓값이라고 정의할 수 있습니다.

자석 놀이

- 1행만 포함된 열과 2행만 포함된 열만 인접하지 않으면 전부 가능하므로 다음과 같은 점화식을 세울 수 있습니다.

- dp[i][0]=max(dp[i][0],dp[i][2])+arr[i][0]
- dp[i][1]=max(dp[i][1],dp[i][2])+arr[i][1]
- dp[i][2]=max(dp[i][0],dp[i][1],dp[i][2])+arr[i][0]+arr[i][1]

자석 놀이

- 마지막열의 2행을 포함해야하므로 답은 max(dp[n][1], dp[n][2])입니다.

- 1열 1행을 포함해야 하므로 dp[1][1]의 값을 충분히 작은값으로 설정하는 등의 예외처리가 필요함에 유의해야 합니다.

D. 드론 조작

- 종류 – brute_force, simulation
- 출제진 의도 – Medium

- 제출 40회 / 맞은 사람 33명 (정답률 82.5%)
- First Solve: seawon0808, 10분
- 출제자: qqq831347

드론 조작

- 단순히 드론의 조작을 구현하면 되는 시뮬레이션 문제입니다.
- 여러가지 구현방법이 있을 수 있지만 의도한 방법은 다음과 같습니다.

드론 조작

- 명령어를 받고 이동을 시도할 위치를 구합니다.
- 이동을 시도할 위치에 장애물이 있는지 확인합니다. 이 때 장애물의 위치는 2차원 배열 등으로 미리 저장해둘 수 있습니다. 단, 음수 좌표가 있으므로 500을 더하는 등의 방법이 필요합니다.
- 장애물이 없다면 그 위치로 드론을 이동시킵니다.

E. 일하기 시러

- 종류 – dp, knapsack
- 출제진 의도 – Hard

- 제출 45회 / 맞은 사람 9명 (정답률 20%)
- First Solve: dabbler1, 25분
- 출제자: ljwljw8541

일하기 시러

- i번째 과일을 x번 수확한다고 생각해 봅시다.
- 총 $A_i + (x-1)B_i$ 일이 걸리고, xC_i 만큼의 돈을 벌 수 있습니다.
- M과 K가 충분히 작으므로, 모든 경우에 대해서 구할 수 있습니다.
- 날짜와 가격에 대해 냅색 알고리즘을 수행하면 됩니다.

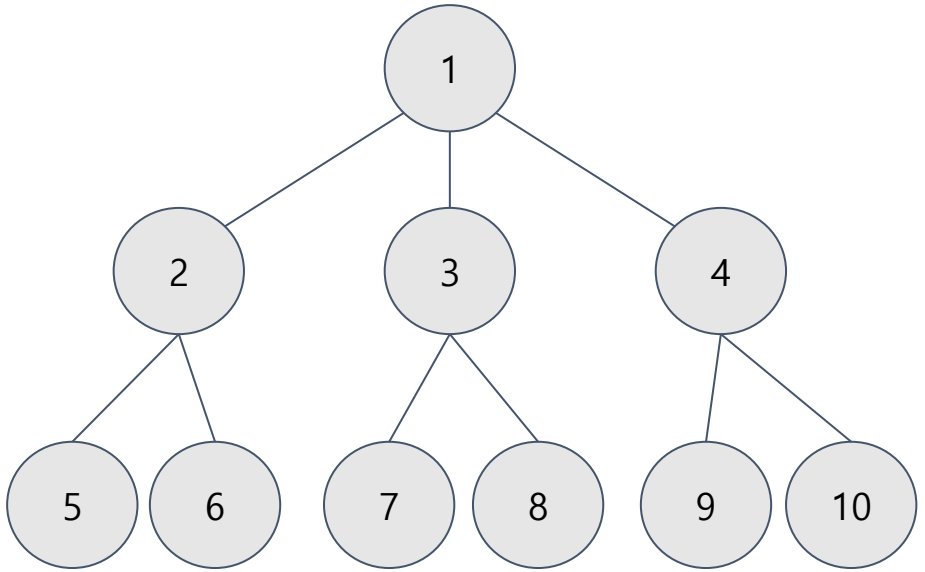
F. 풍성한 트리

- 종류 – Tree, Graph
- 출제진 의도 – Hard

- 제출 34회 / 맞은 사람 11명 (정답률 32.353%)
- First Solve: dabbler1, 32분
- 출제자: sangyoon9

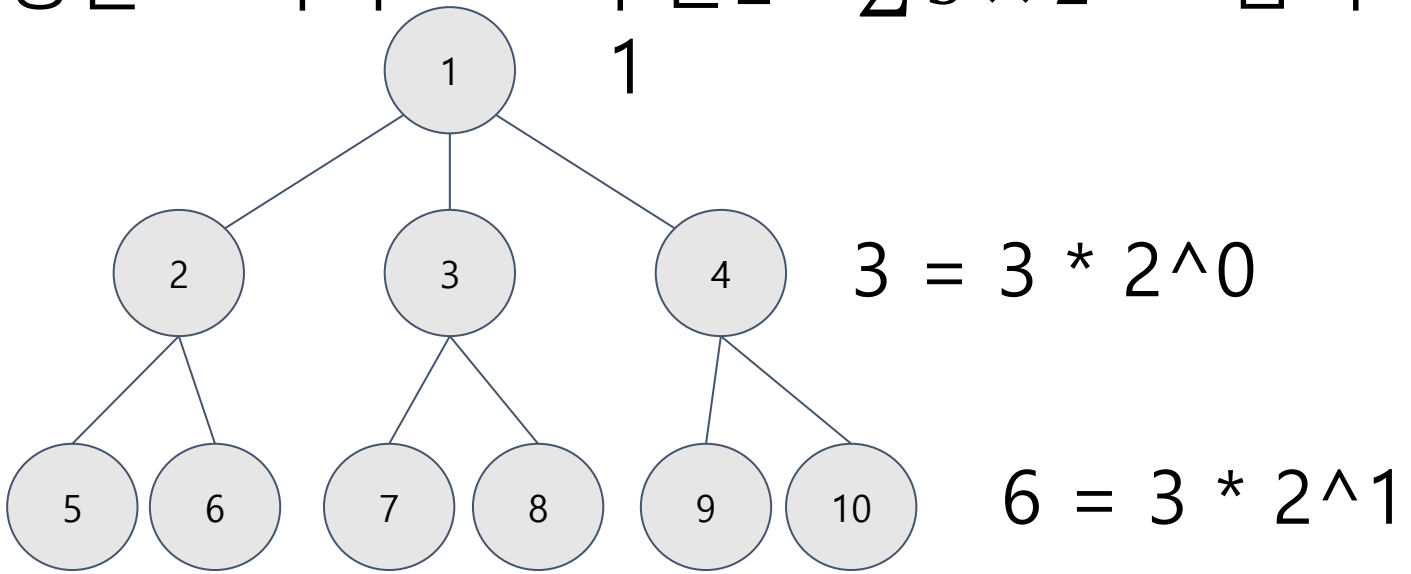
풍성한 트리

- ‘풍성한 트리’의 정의를 생각해 보았을 때, 풍성한 트리는 리프 노드의 차수가 1이고, 리프 노드를 제외한 모든 노드의 차수가 3인 다음과 같은 모양의 트리임을 알 수 있습니다.



풍성한 트리

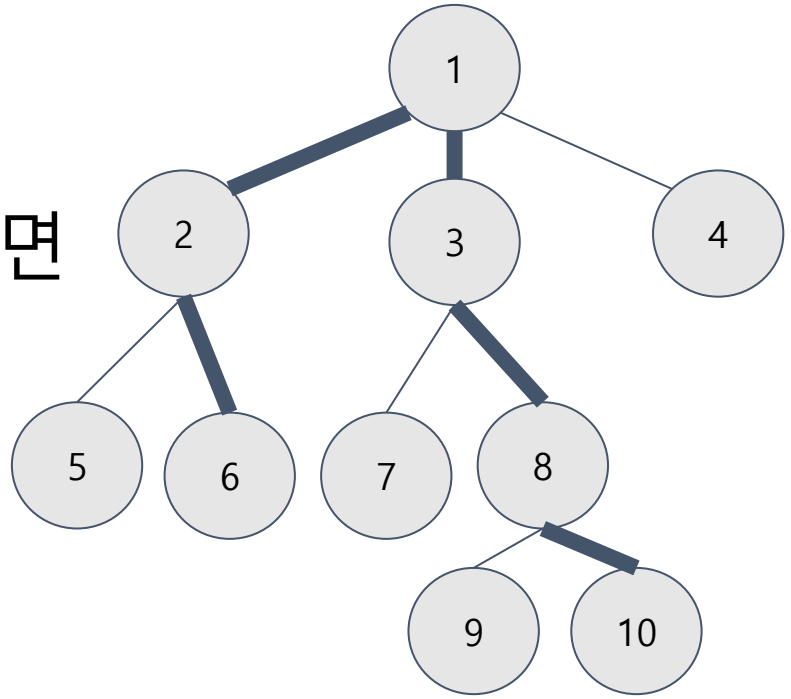
- 주어진 트리가 풍성한 트리인지 확인하기 위한 조건은 다음과 같습니다.
- Depth가 D인 풍성한 트리의 노드 수는$1 + \sum 3 \times 2^{D-1}$ 입니다.



풍성한 트리

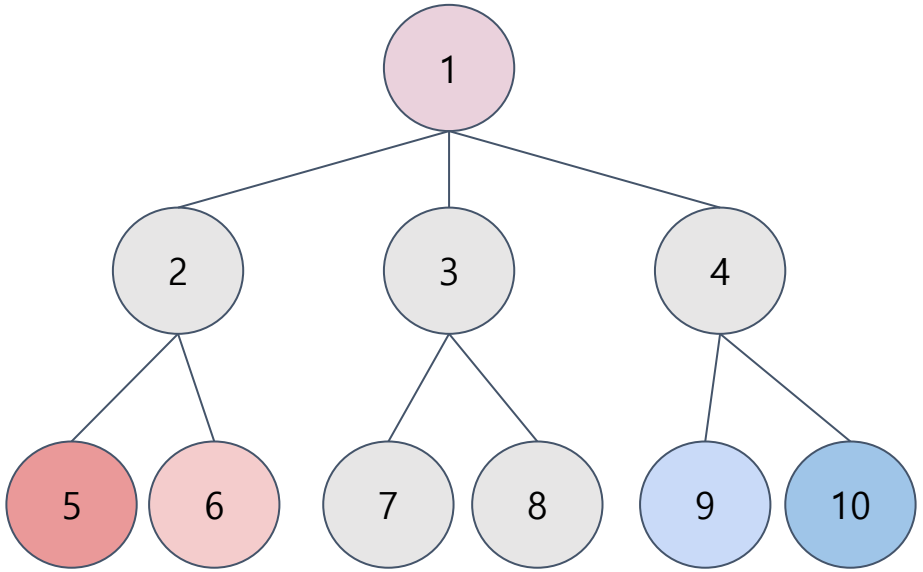
- 따라서 주어진 노드 수를 바탕으로 트리의 깊이 D를 역산할 수 있습니다. 트리에서 가장 멀리 떨어진 두 노드의 거리가 D*2고 모든 노드의 차수가 1 또는 3이라면 주어진 트리는 풍성한 트리임이 보장됩니다.

- 어떠한 리프 노드라도 정점과의 깊이가 달라지면 가장 멀리 떨어진 두 노드의 거리는 D*2를 초과하기 때문입니다.



풍성한 트리

- 이후 가장 멀리 떨어진 두 (리프)노드로부터 거리가 모두 D인 노드를 찾으면 그 노드는 풍성한 트리의 유일한 루트 노드입니다.
- 모든 풍성한 트리 입력에 대하여 가능한 루트 노드는 1개뿐입니다.



5에서 거리가 2인 노드 (1,5)
10에서 거리가 2인 노드 (1,9)
: 공통으로 거리가 2인 노드 (1)

G. 이상한 시행

- 종류 – math, ad_hoc
- 출제진 의도 – Hard

- 제출 26회 / 맞은 사람 18명 (정답률 69.231%)
- First Solve: seawon0808, 17분
- 출제자: qqq831347

이상한 시행

- 수열의 항 중 하나를 골라 -(수열의 합)으로 바꾸는 시행을 반복해 수열의 합을 최대화 하는 문제입니다.

- 제목대로 이상해보이는 시행이기에 어떤 관찰이 필요합니다.

이상한 시행

- n=3에서 시행하는 경우를 식으로 나타내 봅시다.

a_1	a_2	a_3	$-S$
a	b	c	$-a-b-c$
a	$-a-b-c$	c	b

- a_2에 시행을 했더니 a_2와 -S의 값이 바뀐것을 알 수 있습니다.
- 계산을 해보면 더 큰 n에 대해서도 성립함을 알 수 있습니다.

이상한 시행

- 즉, 시행의 결과 수열의 값은 $a_1, a_2, a_3, \ldots, a_n, -sum(a)$ 의 값 중 n개를 고른것과 같습니다.

- 따라서 그 n+1개 수 중 큰 n개를 골라 합한것이 답이 됩니다.

H. 타임머신

- 종류 – graph, BFS
- 출제진 의도 – Hard

- 제출 29회 / 맞은 사람 10명 (정답률 34.483%)
- First Solve: imax, 50분
- 출제자: qqq831347

타임머신

- 음수 가중치 간선이 단 하나 있고 나머지 간선들의 가중치가 1인 방향 그래프에서 최단 경로를 찾는 문제입니다.

- 음수 간선 가중치가 있는 일반적인 그래프에 대해 빠르게 최단경로를 구하는 알고리즘은 없기 때문에, 음수 간선이 단 하나라는 점을 이용합니다.

타임머신

- 1. 타임머신을 쓰지 않는 경우:
- 그냥 BFS를 돌려보면 됩니다.

타임머신

- 2. 타임머신을 쓰는 경우:

- 1번 정점에서 a번 정점까지 가는 최단거리와 b번 정점에서 N번 정점까지 가는 최단 거리를 계산해주면 됩니다.

- 이 때 b번 정점에서 a번 정점까지의 최단거리가 c보다 작다면 음수 사이클이 있는것입니다. 하지만 1번 정점에서 a번 정점으로 갈 수 있는 경로와 b번 정점에서 N번 정점으로 갈 수 있는 경로가 있어야 답이 -inf가 나옴에 유의해야 합니다.

I. U, Our Star!

- 종류 – dp, knapsack
- 출제진 의도 – Hard

- 제출 13회 / 맞은 사람 8명 (정답률 61.538%)
- First Solve: hyperbolic, 69분
- 출제자: pwj6800

U, Our Star!

- 최대 적립금을 얻기 위해 필요한 비용 중 가장 작은 비용을 k라고 합시다.
- k부터 199,999까지의 비용이 가능한지 순서대로 확인합니다.
- 모두 불가능하다면, $k - X$부터 $k - 1$까지의 비용이 가능한지 순서대로 확인합니다.
- 모두 불가능하다면, $k - 2X$부터 $k - X - 1$까지의 비용이 가능한지 순서대로 확인하고, 이 과정을 반복합니다.

U, Our Star!

- 문제의 조건에 따르면, 최대 적립금을 얻기 위해 지불해야 하는 비용의 최소 비용의 조건은 최대 100,000입니다.
- 각 물건의 비용도 최대 100,000입니다.
- 따라서 dp[i]를 지불 비용 i가 가능한지 여부라고 하면, 199,999까지의 비용만을 고려하면 됩니다.
- 고려해야 하는 구매 가능한 상품의 수는 최대 100이므로, 냅색을 이용해 계산할 수 있습니다.

J. Counting portal

- 종류 – prefix sum, two pointer
- 출제진 의도 – Hard

- 제출 14회 / 맞은 사람 5명 (정답률 35.714%)
- First Solve: sjhi00, 52분
- 출제자: qqq831347

Counting portal

- 임의의 두 꼭짓점을 잡았을 때, 지옥문이 되는지 판단하는 것은 누적합을 통해 $O(1)$로 수행이 가능합니다.
- 임의의 두 꼭짓점을 잡는 경우의 수는 $O(N^2M^2)$이므로, 시간 초과가 발생합니다.

Counting portal

- 임의의 두 행 $i, j (i < j)$를 고정시켰을 때, 가능한 순서쌍의 개수를 찾아봅시다.
- 먼저 $l = 1, r = 4$ 로 놓습니다. 오른쪽 아래 꼭짓점을 (j, r)로 고정합니다.
  $l \leq c \leq r - 3$인 모든 c에 대해, 왼쪽 위 꼭짓점을 (i, c)라고 두었을 때 가능한 순서쌍의 수를 구합니다.

Counting portal

- $(i, l), (j, r)$을 꼭짓점으로 잡았을 때 지옥문이 가능하다면, $r - l - 2$개만큼의 가능한 순서쌍이 존재합니다. r을 증가시킨 후, 다음 구간을 확인합니다.
- 불가능하다면, $l \leq c \leq r - 3$ 인 모든 c 에 대해, (i, c), (j, r)을 꼭짓점으로 갖는 지옥문은 존재하지 않습니다. 또한, r보다 큰 k에 대해서, $(i, c), (j, k)$를 꼭짓점으로 갖는 지옥문 또한 존재하지 않습니다. 따라서 l을 증가시킵니다.
- 이 때, $l + 3 \leq r$을 만족하도록 항상 유지합니다.
- 이 과정을 반복하여 임의의 두 행 $i, j (i < j)$를 고정시켰을 때, 가능한 순서쌍의 개수를 모두 찾을 수 있습니다.

Counting portal

- 따라서 임의의 두 행을 찾는 데 $O(N^2)$, 두 행을 고정시켰을 때 가능한 순서쌍의 개수를 찾는 데 $O(M)$이 걸리므로 총 시간복잡도는 $O(N^2M)$입니다.

K. 블록 쌓기

- 종류 – greedy
- 출제진 의도 – Hard

- 제출 10회 / 맞은 사람 4명 (정답률 40%)
- First Solve: sjhi00, 29분
- 출제자: qqq831347

블록 쌓기

- 연속된 칸에 블록을 1개씩 쌓는 시행을 최소로 하여 i번째 칸에 올라온 블록이 a_i개가 되도록 만드는 문제입니다.
- 각 시행마다 연속된 블록길이의 제곱만큼의 비용이 들고 총 비용도 최소화 해야 합니다.
- 먼저 시행을 최소로 하려면 어떻게 해야하는지 생각해봅시다.

블록 쌓기

- b_i를 i번째 칸에 쌓은 블록의 수라고 가정하고 $d1_i = a_i - a_{i-1}$, $d2_i = b_i - b_{i-1}$이라고 정의합시다. 편의상 $b_0 = b_{n+1} = 0 = a_0 = a_{n+1}$이라고 가정합니다. 이 때 목표는 $d1_i = d2_i$가 되도록 만드는 것입니다.

- 구간 $[l, r]$에 실행을 할 때 $d2_l$은 1 증가하고 $d2_{r+1}$은 1 감소합니다.

- 즉, $sum(|d2_i|)$의 값은 한 시행 당 최대 2씩 증가합니다. 따라서 최소 시행횟수는 $sum(|d1_i|)/2$의 값보다 작아질 수 없습니다.

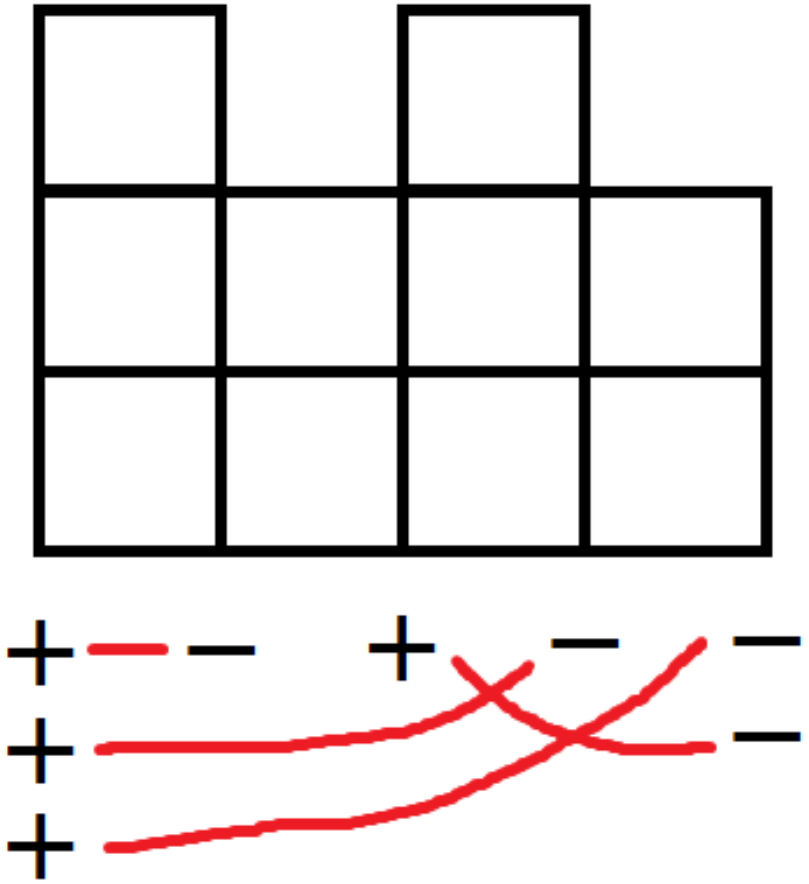
블록 쌓기

- 시행횟수가 최소가 되려면 $d1_i > 0$인 i에 대해서는 항상 $d2_i$가 더해져야 하고 $d1_i < 0$인 i에 대해서는 항상 $d2_i$가 빼져야 합니다.

- 이는 양수와 음수를 앞에서 적당히 매칭해주는것과 같고 비용이 시행 길이의 제곱이므로 그리디하게 매칭해주는것이 최적입니다.

블록 쌓기

- 예제의 경우, 그림과 같습니다.



L. 동전 뒤집기

- 종류 – constructive
- 출제진 의도 – Hard

- 제출 7회 / 맞은 사람 2명 (정답률 28.571%)
- First Solve: hyperbolic, 123분
- 출제자: qqq831347

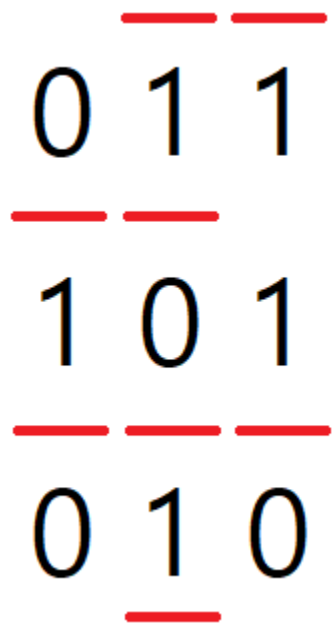
동전 뒤집기

- $N \times N$ 크기의 격자모양으로 동전들이 놓여있고 직사각형 모양으로 동전들을 뒤집는 시행을 $\left\lceil \frac{N+1}{2} \right\rceil^2$ 번 이하로 하여 동전들을 모두 앞면으로 뒤집는 문제입니다.

- 제한 상 홀수인 N이 가능하다면 당연히 짝수인 N도 가능하기 때문에 N을 홀수라고 가정하겠습니다.

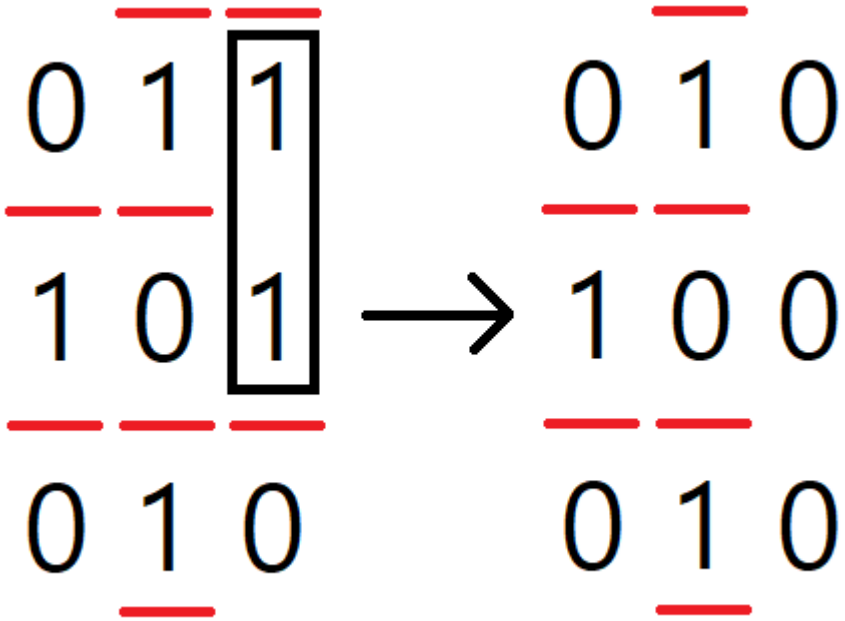
동전 뒤집기

- 0행과 n+1행에 모두 앞면이 위로 향한 동전들밖에 없다고 가정하고 상하로 인접한 동전들을 같게 한다고 생각해봅시다.
- 상하로 인접한것 중 다른것에 빨간 선을 긋는다고 생각하면 빨간선 들을 없앤다고 생각할 수 있습니다.



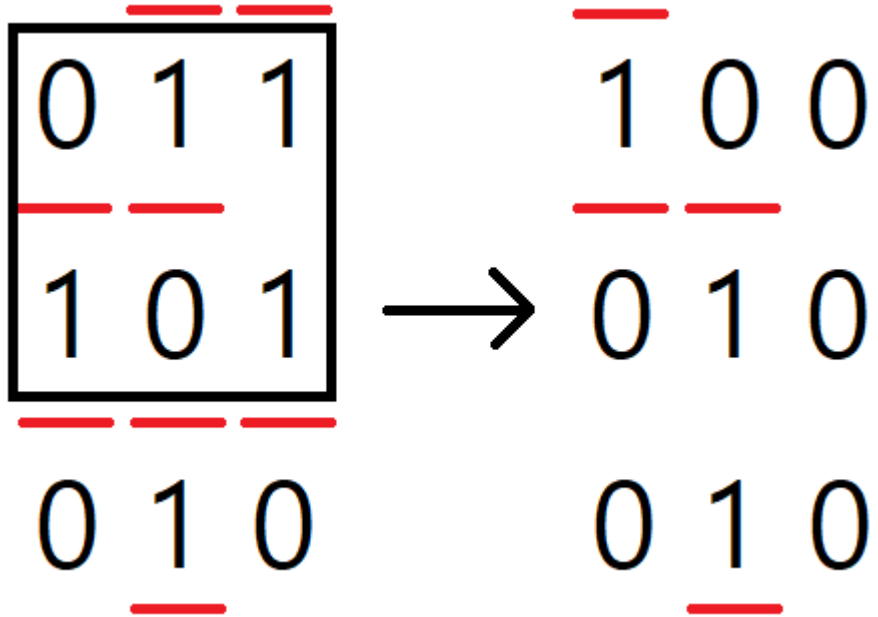
동전 뒤집기

- 이 때 한 열의 동전을 몇 개 뒤집어서 빨간 선을 2개 없앨 수 있습니다.
- 하지만 빨간 선은 최대 n^2개 정도이기 때문에 이것만으로는 부족합니다.



동전 뒤집기

- 모든 열을 다 골라서 뒤집는다고 생각해봅시다.
- 빨간 줄 두 행의 상태가 반전되고, 이를 적절히 이용하면 빨간선의 개수를 크게 줄일 수 있습니다.



동전 뒤집기

- 다음과 같은 알고리즘을 생각해봅시다.
- 먼저 빨간 선이 많이 있는 행 부터 내림차순으로 정렬합니다.
- 앞에서 부터 두개씩 골라 그 빨간선의 개수 합이 n보다 크다면 그 두 줄이 반전되도록 모든 열을 골라 뒤집습니다.
- 이 후 열을 하나씩 골라 빨간선을 2개씩 지웁니다.

동전 뒤집기

- 이 알고리즘이 주어진 시행 횟수 제한 안에서 실행됨을 보일 수 있습니다.
- k쌍을 뒤집었다고 가정합시다.
- 이 때 뒤집은 $2k$개의 줄에서의 빨간 선은 $k(n-1)$개 이하입니다.
- 뒤집지 않은 $n+1-2k$개의 줄에서의 빨간 선은 $(\frac{n-1}{2}+1)+\frac{n-1}{2}*(n-2k)$개 이하 입니다.
- 잘 계산해보면 남은 빨간선의 개수는 $\frac{n^2-1}{2}+1$개 이하이고 총 시행횟수는 $\frac{n^2+1}{4}+k$ 이하가 됩니다.
- 이 때 k는 $\frac{n+1}{2}$ 이하이므로 $\left\lfloor\frac{n^2+1}{4}+k\right\rfloor \leq \left(\frac{n+1}{2}\right)^2$가 성립하게 됩니다.