2017/08/14

최백준 choi@startlink.io

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (0초)

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5					

- 1초:

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	

- 2초:

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8						

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18	16			

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 이런식으로 BFS를 진행한다.

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 덱을 사용해 순간 이동은 덱의 앞에, 걷기는 덱의 뒤에 넣는 방법도 생각해 볼 수 있다.

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- C++ (큐): https://gist.github.com/Baekjoon/9a2f48a57c0a2782869e4b69026a0972
- C++ (덱): https://gist.github.com/Baekjoon/8e524d4a7cfd61263c94e3c6666f99b1
- Java (큐): https://gist.github.com/Baekjoon/9134f710bf6d99ff7f1df27e36b208e6
- Java (덱): https://gist.github.com/Baekjoon/f83e1a3efaac26be2e254352ee098751

알고스팟

https://www.acmicpc.net/problem/1261

- 미로는 N*M크기이고, 총 1*1크기의 방으로 이루어져 있다
- 빈 방은 자유롭게 다닐 수 있지만, 벽은 부수지 않으면 이동할 수 없다
- (x, y)에 있을 때, 이동할 수 있는 방은 (x+1, y), (x-1, y), (x, y+1), (x, y-1) 이다
- (1, 1)에서 (N, M)으로 이동하려면 벽을 최소 몇 개 부수어야 하는지 구하는 문제

알고스팟

https://www.acmicpc.net/problem/1261

- 처음 상태

0	0	1	1	1	1
0	1	0	0	0	0
0	0	1	1	1	1
1	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	0

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 부수지 않고 이동할 수 있는 곳

0	0	1	1	1	1
0	1	0	0	0	0
0	0	1	1	1	1
1	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	0

0	0				
0					
0	0				

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 1개 부수고 이동할 수 있는 곳

0	0	1	1	1	1
0	1	0	0	0	0
0	0	1	1	1	1
1	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	0

0	0	1			
0	1	1	1	1	1
0	0	1			
1	1	1	1	1	
1		1	1		
	1	1	1		

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 2개 부수고 이동할 수 있는 곳

0	0	1	1	1	1
0	1	0	0	0	0
0	0	1	1	1	1
1	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	0

0	0	1	2	2	2
0	1	1	1	1	1
0	0	1	2	2	2
1	1	1	1	1	2
1	2	1	1	2	2
2	1	1	1	2	2

알고스팟

https://www.acmicpc.net/problem/1261

- BFS탐색을 벽을 부순 횟수에 따라서 나누어서 수행해야 한다.
- C/C++: https://gist.github.com/Baekjoon/3d8ed2a3976c7affbd73
- Java: https://gist.github.com/Baekjoon/e66c8bed6b5d440d5bafbfe882acc19e
- 시간 복잡도: O(N^2)

알고스팟

https://www.acmicpc.net/problem/1261

- 어차피 벽을 뚫는다와 안 뚫는다로 나누어지기 때문에, 덱을 사용한다
- 벽을 뚫는 경우에는 뒤에, 안 뚫는 경우에는 앞에 추가한다.
- C/C++: https://gist.github.com/Baekjoon/9da1eed82383645026cc
- Java: https://gist.github.com/Baekjoon/0cc05c42a1ab3b3b325f50e1de371442

- 시간 복잡도: O(N^2)

숨바꼭질 2

https://www.acmicpc.net/problem/12851

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간을 구하는 문제, 그리고 그러한 방법의 개수도 구해야 한다

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (1초)

숨바꼭질 2

https://www.acmicpc.net/problem/12851

```
while (!q.empty()) {
    int now = q.front(); q.pop();
    for (int next : {now-1, now+1, now*2}) {
        if (0 <= next && next <= MAX) {
            if (check[next] == false) {
                q.push(next); check[next] = true;
                dist[next] = dist[now] + 1;
                cnt[next] = cnt[now];
            } else if (dist[next] == dist[now] + 1) {
                cnt[next] += cnt[now];
            }
        }
    }
}
```

숨바꼭질 2

https://www.acmicpc.net/problem/12851

- C++: https://gist.github.com/Baekjoon/39ca797291fc85129c6f1280bf261b5b
- Java: https://gist.github.com/Baekjoon/e170e8d3cfeb258f001fa951bfa747e7

벽 부수고 이동하기

https://www.acmicpc.net/problem/2206

- NxM의 행렬로 나타내는 지도에서 (1, 1)에서 (N,M)으로 최단 거리로 이동하는 문제
- 0은 빈 칸, 1은 벽
- 단, 벽은 한 번 부수고 지나갈 수 있다

벽 부수고 이동하기

https://www.acmicpc.net/problem/2206

- 벽을 부순다는 조건이 없으면 일반적인 미로 탐색 문제이다
- 어떤 칸에 방문했을 때, 벽을 부순 적이 있는 경우와 아직 부순 적이 없는 경우는 다른 경우 이기 때문에
- 상태 (i, j) 대신에 (i, j, k) (k == 0이면 벽을 부순 적이 없음, 1이면 있음) 으로 BFS 탐색을 진행한다.

벽 부수고 이동하기

https://www.acmicpc.net/problem/2206

- C++: https://gist.github.com/Baekjoon/488504115d1acaf9dc319d05c4b59418
- Java: https://gist.github.com/Baekjoon/ac92846bffeeb63f6f8c314c5415b9b7

탈출

https://www.acmicpc.net/problem/3055

- 지도는 R행 C열이다
- 비어있는 곳은 '.'
- 물이 차있는 지역은 '*'
- 돌은 'X'
- 비버의 굴은 'D'
- 고슴도치의 위치는 'S'

탈출

https://www.acmicpc.net/problem/3055

- 먼저, 물이 언제 차는지 미리 구해놓은 다음에
- 고슴도치를 그 다음에 이동시킨다

탈출

https://www.acmicpc.net/problem/3055

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

5	-1	1	0
4	3	2	1
3	2	-1	0
2	1	0	1
3	2	1	2

- 고슴도치의 이동

0			

탈출

https://www.acmicpc.net/problem/3055

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

5	-1	1	0
4	3	2	1
3	2	-1	0
2	1	0	1
3	2	1	2

- 고슴도치의 이동

1			
0			
1			

탈출

https://www.acmicpc.net/problem/3055

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

5	-1	1	0
4	3	2	1
3	2	-1	0
2	1	0	1
3	2	1	2

- 고슴도치의 이동

2			
1			
0			
1			

탈출

https://www.acmicpc.net/problem/3055

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

5	-1	1	0
4	3	2	1
3	2	-1	0
2	1	0	1
3	2	1	2

- 고슴도치의 이동

3			
2			
1			
0			
1			

탈출

https://www.acmicpc.net/problem/3055

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

5	-1	1	0
4	3	2	1
3	2	-1	0
2	1	0	1
3	2	1	2

- 고슴도치의 이동

3	4		
2			
1			
0			
1			

탈출

https://www.acmicpc.net/problem/3055

- C++: https://gist.github.com/Baekjoon/f540851ce65619a8b07c65ac3f2ca245
- Java: https://gist.github.com/Baekjoon/cd2de8bd2b931622f304df1a2a6c2f1b

탈옥

https://www.acmicpc.net/problem/9376

- 빈 칸, 벽, 문으로 이루어진 지도가 주어진다.
- 두 죄수가 탈옥하기 위해서 열어야 하는 문의 최소 개수를 구하는 문제

탈옥

https://www.acmicpc.net/problem/9376

- 두 지도를 상하좌우로 한 칸씩 확장하면
- 두 죄수의 탈옥 경로는
- 어딘가에서 만나서 함께 이동하는 꼴이 된다
- 따라서, 지도의 밖에서 BFS 1번, 각 죄수별로 1번씩 BFS를 수행한다.
- 그 다음, 정답을 합친다
- 이 때, 문에서 만나는 경우는 조심해야 한다

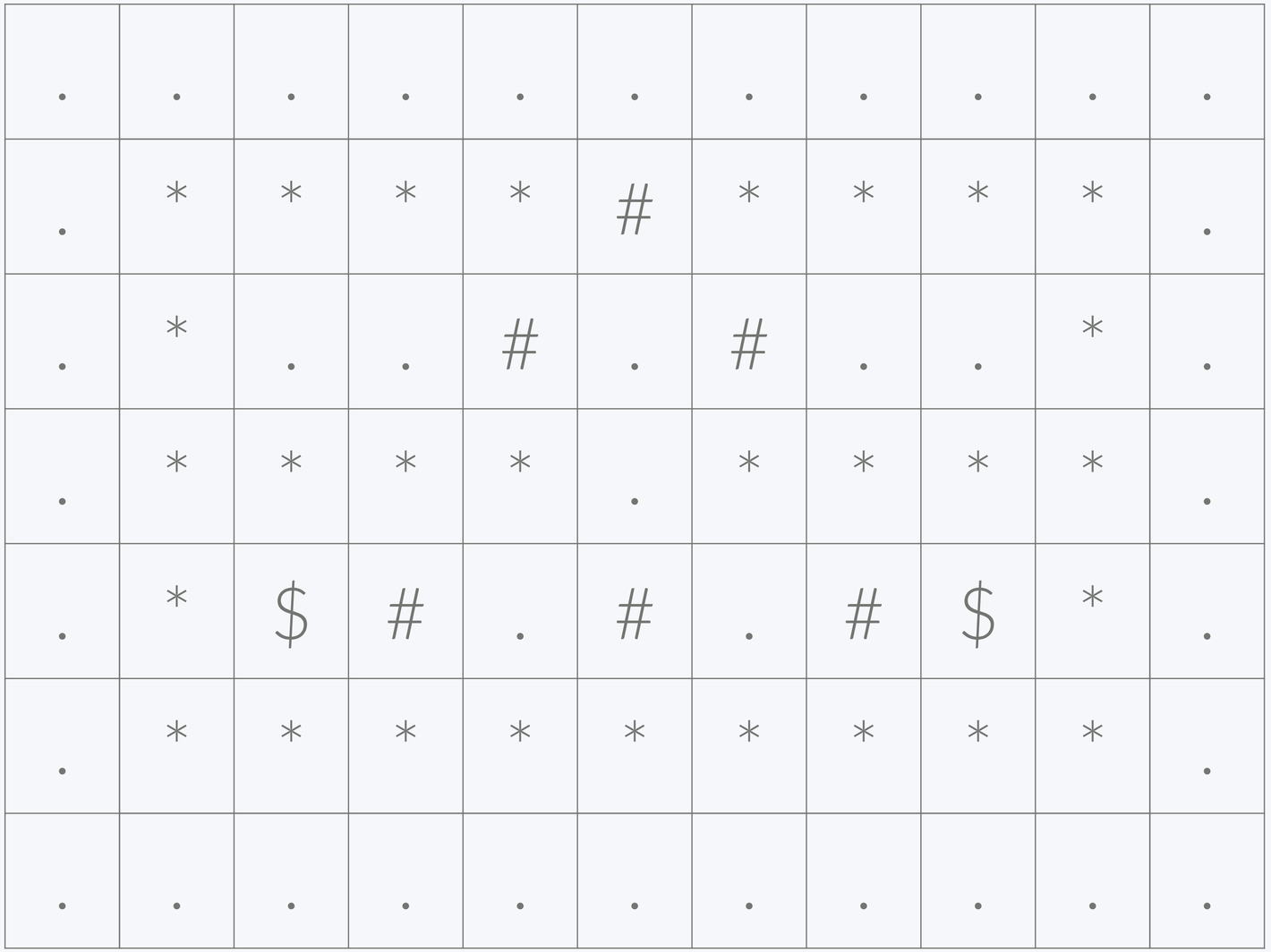
탈옥

https://www.acmicpc.net/problem/9376

*	*	*	*	#	*	*	*	*
*	.	.	#	.	#	.	.	*
*	*	*	*	.	*	*	*	*
*	$	#	.	#	.	#	$	*
*	*	*	*	*	*	*	*	*

탈옥

https://www.acmicpc.net/problem/9376



탈옥

https://www.acmicpc.net/problem/9376

밖에서 부터

0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	1	-	-	-	-	0
0	-	2	2	2	1	2	2	2	-	0
0	-	-	-	-	1	-	-	-	-	0
0	-	3	3	2	2	2	3	3	-	0
0	-	-	-	-	-	-	-	-	-	0
0	0	0	0	0	0	0	0	0	0	0

죄수 1부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	0	1	1	2	2	3	3	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

죄수 2부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	3	3	2	2	1	1	0	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

탈옥

https://www.acmicpc.net/problem/9376

밖에서 부터

0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	1	-	-	-	-	0
0	-	2	2	2	1	2	2	2	-	0
0	-	-	-	-	1	-	-	-	-	0
0	-	3	3	2	2	2	3	3	-	0
0	-	-	-	-	-	-	-	-	-	0
0	0	0	0	0	0	0	0	0	0	0

죄수 1부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	0	1	1	2	2	3	3	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

죄수 2부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	3	3	2	2	1	1	0	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

탈옥

https://www.acmicpc.net/problem/9376

- C++: https://gist.github.com/Baekjoon/682466914543f0aeccc3da8941c015d1
- Java: https://gist.github.com/Baekjoon/99c1122a37a2e025b1994eb52342a35f

열쇠

https://www.acmicpc.net/problem/9328

- BFS를 큐 27개로 수행해야 한다.
- 큐 1개: 일반적인 BFS
- 큐 26개: 문을 열기 위해 기다리는 큐

열쇠

https://www.acmicpc.net/problem/9328

- C++: https://gist.github.com/Baekjoon/a2442e638027d8c8174c
- Java: https://gist.github.com/Baekjoon/c5f9a4756842bcbd8da639ae89f5af7e

크리보드

https://www.acmicpc.net/problem/11058

1. 화면에 A를 출력한다.
2. Ctrl-A: 화면을 전체 선택한다
3. Ctrl-C: 전체 선택한 내용을 버퍼에 복사한다
4. Ctrl-V: 버퍼가 비어있지 않은 경우에는 화면에 출력된 문자열의 바로 뒤에 버퍼의 내용을 붙여넣는다.

- 크리보드의 버튼을 총 N번 눌러서 화면에 출력된 A개수를 최대로하는 프로그램을 작성하시오.

크리보드

https://www.acmicpc.net/problem/11058

- D[N] = 크리보드의 버튼을 총 N번 눌러서 화면에 출력된 A개수의 최대값

크리보드

https://www.acmicpc.net/problem/11058

- D[i] = 크리보드의 버튼을 총 i번 눌러서 화면에 출력된 A개수의 최대값
- Ctrl + A, Ctrl + C, Ctrl + V 는 총 3번이다.
- 이것을 총 몇 번 반복할지를 j번이라고 한다

크리보드

https://www.acmicpc.net/problem/11058

- D[i] = 크리보드의 버튼을 총 i번 눌러서 화면에 출력된 A개수의 최대값
- Ctrl + A, Ctrl + C, Ctrl + V 는 총 3번이다.
- 이것을 총 몇 번 반복할지를 j번이라고 한다
- D[i] = j*(D[i-j-1])

크리보드

https://www.acmicpc.net/problem/11058

- C++: https://gist.github.com/Baekjoon/c2cf266e84d4f8fe52adab8486d054c2
- Java: https://gist.github.com/Baekjoon/0597c1e1ab1c4906ba6bebd3eab6a7d1

BOJ 거리

https://www.acmicpc.net/problem/12026

- 스타트는 BOJ를 외치면서 링크를 만나러 가려고 한다
- 따라서, 스타트는 B, O, J, B, O, J, B, O, J, ... 순서로 보도블럭을 밟으면서 점프를 할 것이다
- 스타트가 링크를 만나는데 필요한 에너지 양의 최소값을 구하는 프로그램을 작성하시오.

BOJ 거리

https://www.acmicpc.net/problem/12026

- D[N] = N에 오는 최소 에너지

BOJ 거리

https://www.acmicpc.net/problem/12026

- C++: https://gist.github.com/Baekjoon/d5648836057d6621aaa8baacb5762160
- Java: https://gist.github.com/Baekjoon/b5e6050d7ca94c08f3f6ddde384a6665

ABC

https://www.acmicpc.net/problem/12969

- D[i][a][b][p] = 길이가 i이고, A의 개수가 a개, B의 개수가 b개, s[i] < s[j] 쌍이 p개 있는 문자열이 가능한가?

ABC

https://www.acmicpc.net/problem/12969

- D[i][a][b][p] = 길이가 i이고, A의 개수가 a개, B의 개수가 b개, s[i] < s[j] 쌍이 p개 있는 문자열이 가능한가?
- i번째 글자가 A인 경우
- D[i+1][a+1][b][p]
- i번째 글자가 B인 경우
- D[i+1][a][b+1][p+a]
- i번째 글자가 C인 경우
- D[i+1][a][b][p+a+b]

ABC

https://www.acmicpc.net/problem/12969

- C/C++: https://gist.github.com/Baekjoon/6d73e5e30f54b5f2c2d20a455be168df
- Java: https://gist.github.com/Baekjoon/c9ae879b8fda3bb0ce0b89e31b726168

출근 기록

https://www.acmicpc.net/problem/14238

- A: 아무 때나 가능
- B: 출근을 하면 다음날 쉬어야 함
- C: 출근한 다음날과 다다음날 쉬워야 함
- 출근 기록 S의 모든 순열 중에서 올바른 출근 기록인 것 아무거나 찾는 문제

출근 기록

https://www.acmicpc.net/problem/14238

- A: 아무 때나 가능
- B: 출근을 하면 다음날 쉬어야 함
- C: 출근한 다음날과 다다음날 쉬워야 함
- 출근 기록 S의 모든 순열 중에서 올바른 출근 기록인 것 아무거나 찾는 문제

출근 기록

https://www.acmicpc.net/problem/14238

- 출근 기록에서 중요한 정보는 A의 개수(a), B의 개수(b), C의 개수 (c)
- B와 C때문에, 전날과 전전날 누가 일했는지가 필요하다.

출근 기록

https://www.acmicpc.net/problem/14238

- D[a][b][c][p1][p2] = A, B, C의 개수가 (a, b, c) 이고, 전날 일한 사람이 p1, 그 전날 일한 사람이 p2인 것이 가능한가?

출근 기록

https://www.acmicpc.net/problem/14238

- C++: https://gist.github.com/Baekjoon/581e5620e954a7355a5d729fc7815c55
- Java: https://gist.github.com/Baekjoon/1fe1fe1a432c5af5df71dbab95dbb60a

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- N명의 사람을 한 줄로 세우는데, 양 옆의 사람이 자신보다 크거나 작게 세우는 경우의 수

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- 예를 들어, 4명이 있고, 키가 1, 2, 3, 4라면
- 1324, 2143, 3142, 2314, 3412, 4231, 4132, 2413, 3241, 1423
- 와 같은 10가지 배치가 가능하다.

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- Tall(n) = 1번 사람 > 2번 사람으로 줄을 서는 경우의 수
- Short(n) = 1번 사람 < 2번 사람으로 줄을 서는 경우의 수

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- Tall(n) = 1번 사람 > 2번 사람으로 줄을 서는 경우의 수
- Short(n) = 1번 사람 < 2번 사람으로 줄을 서는 경우의 수
- Tall(n)과 Short(n)과의 관계는?

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- Tall(n) = 1번 사람 > 2번 사람으로 줄을 서는 경우의 수
- Short(n) = 1번 사람 < 2번 사람으로 줄을 서는 경우의 수
- Tall(n)과 Short(n)과의 관계는?
- Tall(n) = Short(n)
- 모든 키가 k인 사람을 n-k+1로 바꾸면 되기 때문
- 즉, n = 1인 경우를 제외하면, Tall(n) + Short(n) = 2*Tall(n)

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- Tall(n) = 1번 사람 > 2번 사람으로 줄을 서는 경우의 수
- Short(n) = 1번 사람 < 2번 사람으로 줄을 서는 경우의 수
- Tall(0) = 1이라고 두면, Tall(1) = Tall(2) = 1
- Tall(n)을 구하기 위해, 가장 키가 큰 사람의 위치를 k라고 하자. (1, 3, 5, …)
- 왼쪽에 k-1명을 세워야 하는 경우의 수: Tall(k-1)
- 오른쪽에 n-k명을 세워야 하는 경우의 수: Short(n-k)
- Comb(n-1, k-1) * Tall(k-1) * Short(n-k)
- = Comb(n-1, k-1) * Tall(k-1) * Tall(n-k)

홍준이의 친위대

https://www.acmicpc.net/problem/3948

- C++: https://gist.github.com/Baekjoon/dc618704171acdc38ce3f769a92f61a8
- Java: https://gist.github.com/Baekjoon/5c00966d53b681737ff034f0c786a9d9

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- 수열 S의 크기 n과 k가 주어졌을 때, 인접한 비트의 개수가 k인 수열 S의 개수

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- D[n][k][l] = N자리, 인접한 1의 개수 k개, 마지막 수 l

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- D[n][k][l] = N자리, 인접한 1의 개수 k개, 마지막 수 l
- 마지막 수가 0인 경우
- 마지막 수가 1인 경우

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- D[n][k][l] = N자리, 인접한 1의 개수 k개, 마지막 수 l
- 마지막 수가 0인 경우
- D[n-1][k][0] + D[n-1][k][1]
- 마지막 수가 1인 경우
- D[n-1][k][0] + D[n-1][k-1][1]

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- 초기값
- D[1][0][0] = 1
- D[1][0][1] = 1

인접한 비트의 개수

https://www.acmicpc.net/problem/2698

- C++: https://gist.github.com/Baekjoon/b0db2fb13f65e6ffc2ea6191b5af9f49
- Java: https://gist.github.com/Baekjoon/ce292440d87aa549d57f774f84727c0f

같은 탑

https://www.acmicpc.net/problem/1126

- N개의 조각이 주어졌을 때, 두 개의 탑을 만든다
- 이 때, 두 탑의 높이를 같게 만드려고 한다.
- 가능한 탑의 높이 중 최대값을 구하는 문제

같은 탑

https://www.acmicpc.net/problem/1126

- 모든 조각의 높이의 합은 500,000을 넘지 않는다
- 즉, 두 탑의 최대 높이는 500,000/2 = 250,000 이다.

같은 탑

https://www.acmicpc.net/problem/1126

- 각각의 조각에 대해서 다음과 같은 세 가지를 결정할 수 있다.

- 첫 번째 탑에 조각을 올려놓는다
- 두 번째 탑에 조각을 올려놓는다
- 조각을 탑 위에 올려놓지 않는다

같은 탑

https://www.acmicpc.net/problem/1126

- 문제를 일반화 할 수 있다

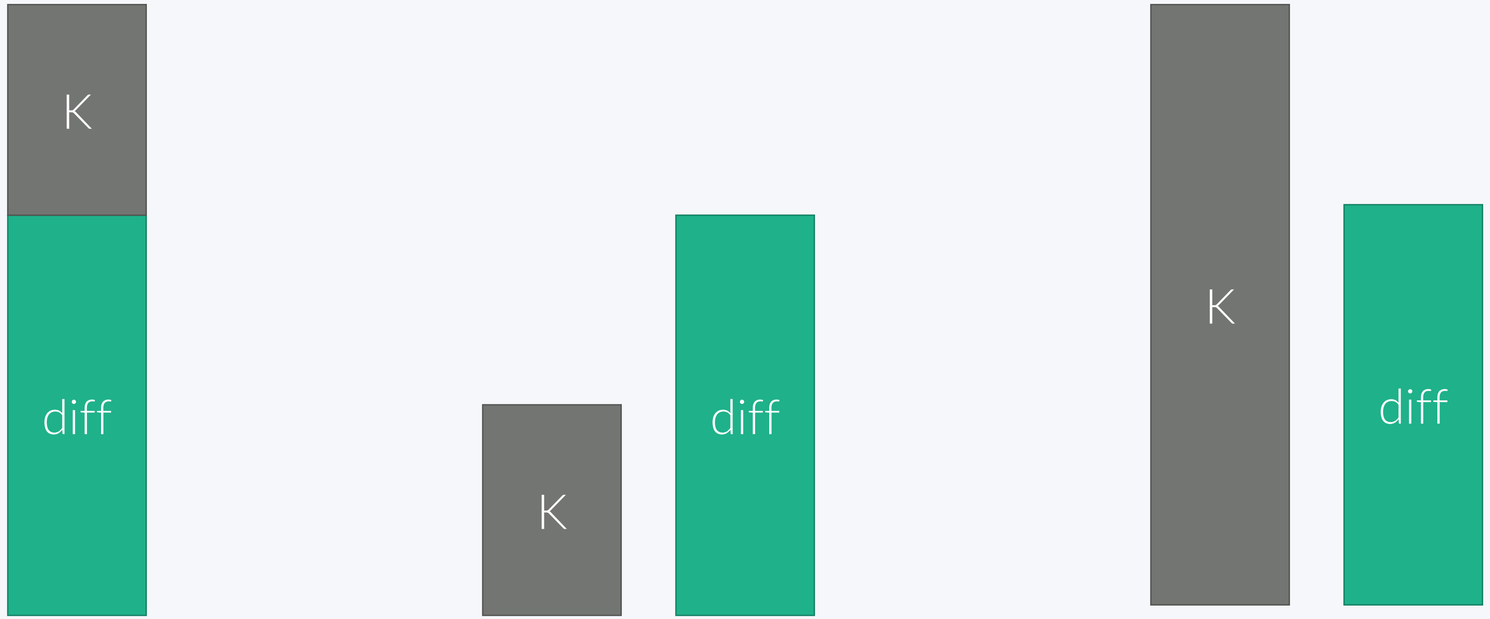
- 탑 하나의 높이는 diff이고, 또 다른 탑의 높이는 0이다.
- 여기서, 조각을 적절히 놓아서 만들 수 있는 가장 큰 두 탑의 높이
- 이 때, 두 탑의 높이는 같아야 한다.

- D[N][diff] = 조각이 N개 남았고, 높은 탑의 높이가 diff

같은 탑

https://www.acmicpc.net/problem/1126

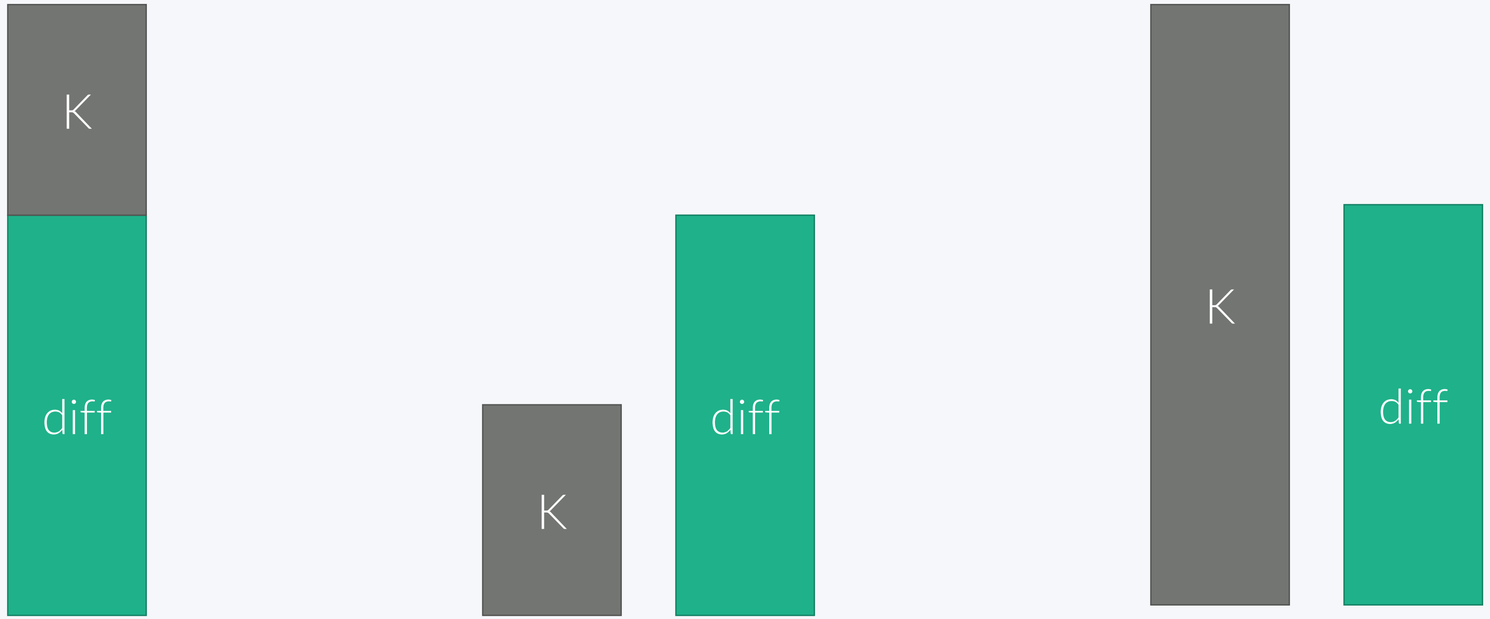
- 탑 하나의 높이는 diff이고, 또 다른 탑의 높이는 0이다.
- 조각의 높이는 K이다.



같은 탑

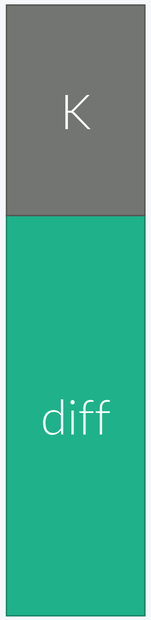
https://www.acmicpc.net/problem/1126

- 블럭을 diff인 탑에 놓는 경우
- 블럭을 0인 탑에 놓는 경우

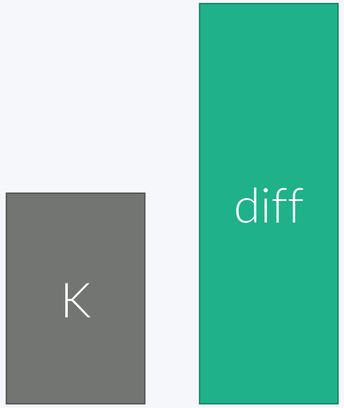


같은 탑

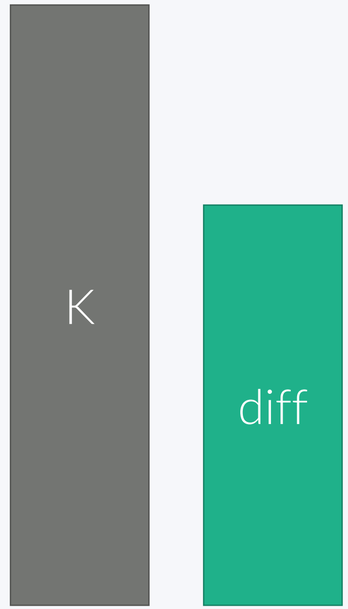
https://www.acmicpc.net/problem/1126



D[K+1][diff+A[K]]



A[K] + D[K+1][diff-A[K]]



diff+ D[K+1][A[K]-diff]

같은 탑

https://www.acmicpc.net/problem/1126

- C/C++: https://gist.github.com/Baekjoon/bfc5bacbe4fae6558cba43eb4da8ec5e
- Java: https://gist.github.com/Baekjoon/16c8b3ec740cbef89c84b2fafddc48f5

ls

https://www.acmicpc.net/problem/5015

- 패턴에 맞는 파일명을 찾는 문제

ls

https://www.acmicpc.net/problem/5015

- D[i][j] = 패턴의 i번째 문자열과 파일명의 j번째 문자열이 매치되는가?

ls

https://www.acmicpc.net/problem/5015

- D[i][j] = 패턴의 i번째 문자열과 파일명의 j번째 문자열이 매치되는가?
- P[i] == S[j]인 경우
- D[i][j] = D[i+1][j+1]

ls

https://www.acmicpc.net/problem/5015

- D[i][j] = 패턴의 i번째 문자열과 파일명의 j번째 문자열이 매치되는가?
- P[i] 가 패턴이 경우
- D[i+1][k] == true인 k가 존재하면 D[i][j] = true

ls

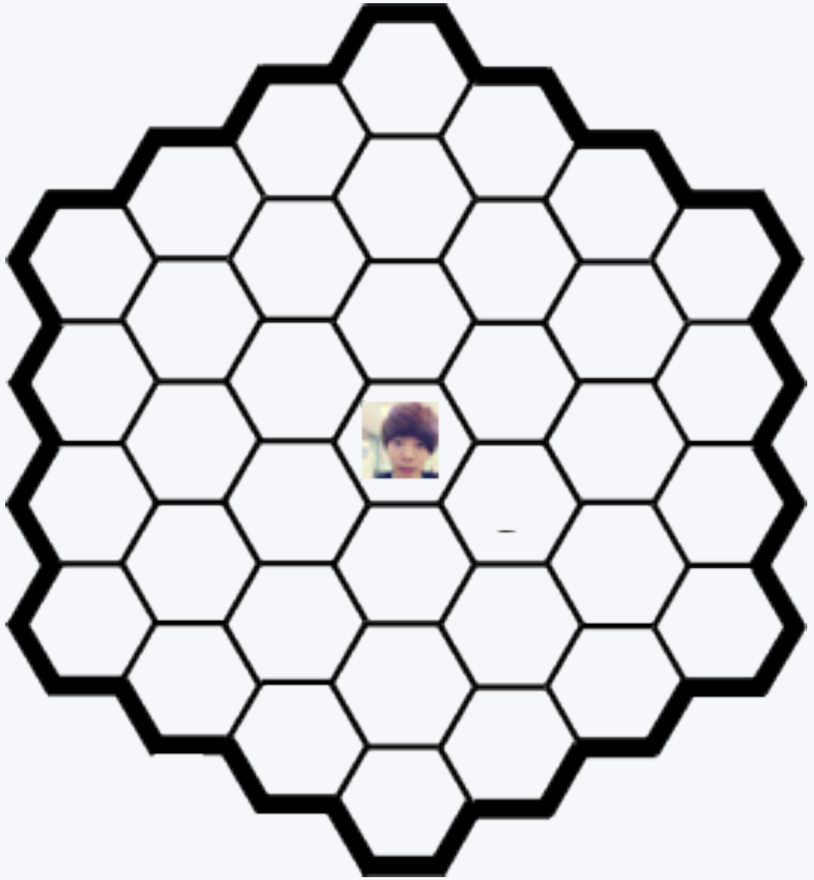
https://www.acmicpc.net/problem/5015

- C++: https://gist.github.com/Baekjoon/ba8a748a1eb07f3bf8a3f49483517e73
- Java: https://gist.github.com/Baekjoon/db61abcdd37b3a2acac63afa08fb157d

미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

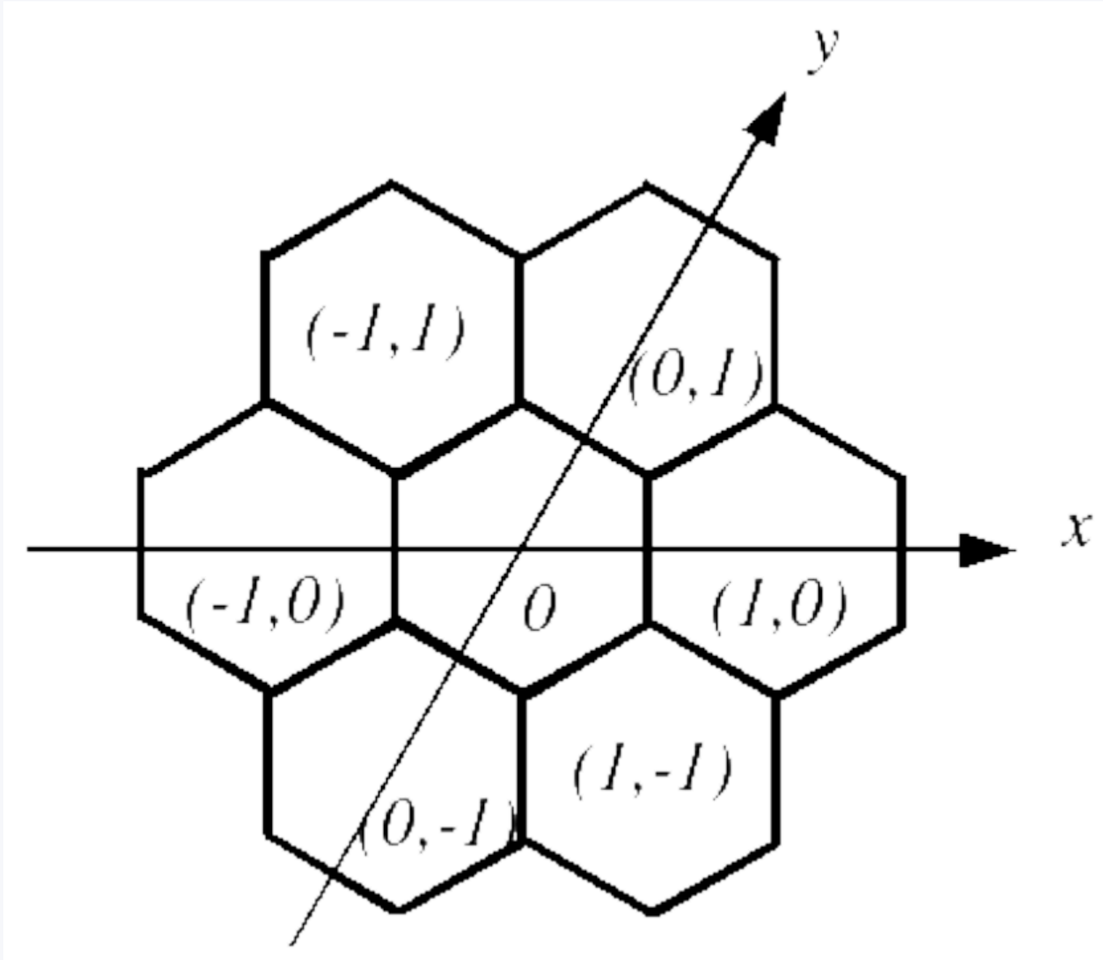
- 상근이가 있는 방에서 시작해서 방을 n번 이동한 뒤, 다시 원래 있던 방으로 돌아오는 경로의 수



미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

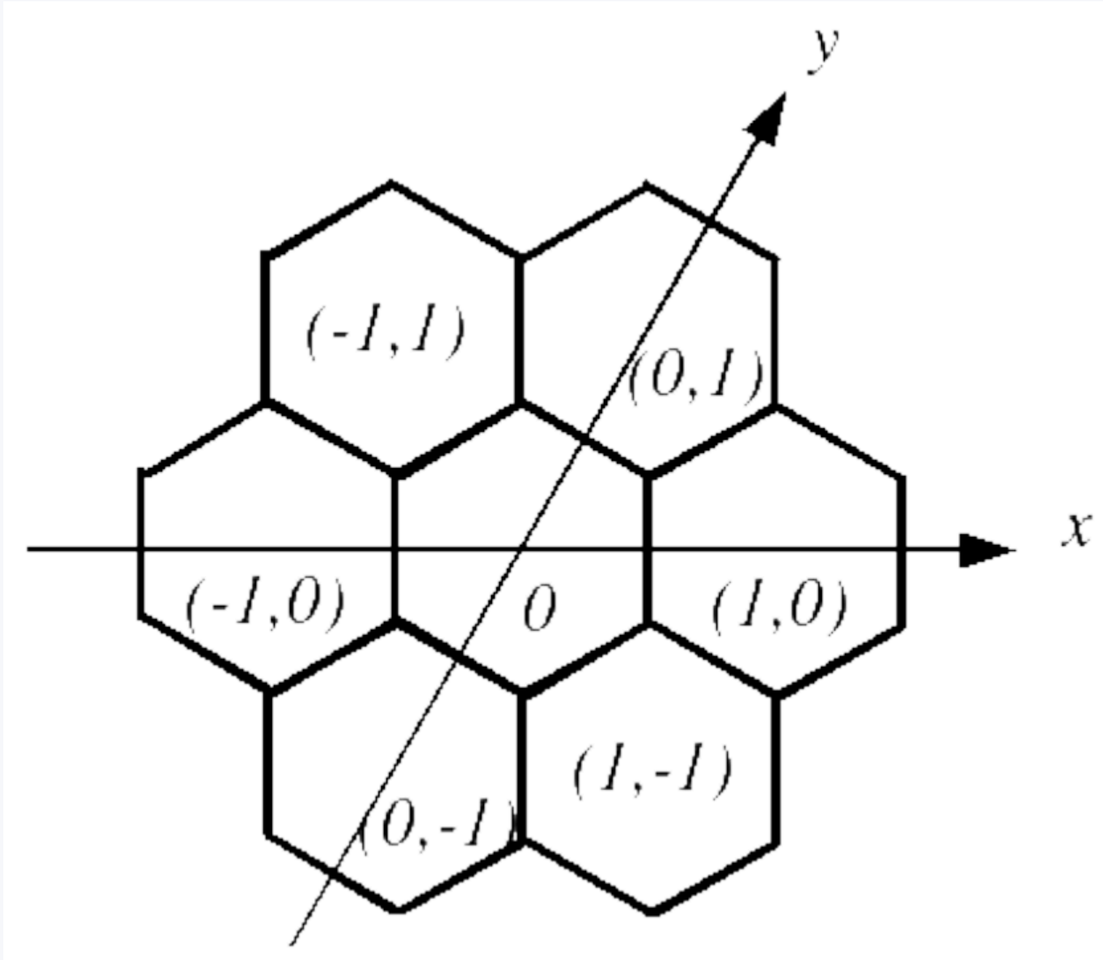
- 상근이가 있는 방에서 시작해서 방을 n번 이동한 뒤, 다시 원래 있던 방으로 돌아오는 경로의 수



미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

- D[k][x][y] = (0, 0)에서 출발해서 (x, y)에 k번만에 가는 경로의 수



미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

- 음수가 나올 수 있기 때문에
- D[k][x][y] = (14, 14)에서 출발해서 (x, y)에 k번만에 가는 경로의 수

미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

```
d[0][14][14] = 1;
for (int k=1; k<=14; k++) {
    for (int i=0; i<m; i++) {
        for (int j=0; j<m; j++) {
            for (int l=0; l<6; l++) {
                int x = i+dx[l];
                int y = j+dy[l];
                if (0 <= x && x < m && 0 <= y && y < m) {
                    d[k][i][j] += d[k-1][x][y];
                }
            }
        }
    }
}
```

미로에 갇힌 상근

https://www.acmicpc.net/problem/5069

- C++: https://gist.github.com/Baekjoon/47388bd3482ba795e1dab4cbf803edb0
- Java: https://gist.github.com/Baekjoon/b9513f51513521592e7e37179912af31

소수

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N이 소수가 되려면, 2보다 크거나 같고, N-1보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 1부터 100까지 소수
- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

소수

Prime Number


```
bool prime(int n) {
    if (n < 2) {
        return false;
    }
    for (int i=2; i<=n-1; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N이 소수가 되려면, 2보다 크거나 같고, N/2보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 이유: N의 약수 중에서 가장 큰 것은 N/2보다 작거나 같기 때문
- N = a × b로 나타낼 수 있는데, a가 작을수록 b는 크다.
- 가능한 a중에서 가장 작은 값은 2이기 때문에, b는 N/2를 넘지 않는다.

소수

Prime Number


```
bool prime(int n) {
    if (n < 2) {
        return false;
    }
    for (int i=2; i<=n/2; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N이 소수가 되려면, 2보다 크거나 같고, 루트N 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 이유: N이 소수가 아니라면, $N = a \times b$로 나타낼 수 있다. $(a \leq b)$
- $a > b$라면 두 수를 바꿔서 항상 $a \leq b$로 만들 수 있다.
- 두 수 a와 b의 차이가 가장 작은 경우는 루트 N이다.
- 따라서, 루트 N까지만 검사를 해보면 된다.

소수

Prime Number


```
bool prime(int n) {
    if (n < 2) {
        return false;
    }
    for (int i=2; i*i<=n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

소수

Prime Number

- 컴퓨터에서 실수는 근사값을 나타내기 때문에, 루트 N과 같은 경우는 앞 페이지 처럼 나타내는 것이 좋다.
- 루트 i ≤ N은
- i ≤ N*N 과 같다.
- 어떤 수 N이 소수인지 아닌지 판별하는데 걸리는 시간 복잡도: O(루트N)

소수 찾기

https://www.acmicpc.net/problem/1978

- 입력으로 주어지는 N개의 소수 중에서 소수가 몇 개 인지 구하는 문제

소수 찾기

https://www.acmicpc.net/problem/1978

- C++: https://gist.github.com/Baekjoon/3597219897f6706f9bfb
- Java: https://gist.github.com/Baekjoon/4132a4e944d689e9119b6d081ebcab2b

소수

Prime Number

- 어떤 수 N이 소수인지 아닌지 알아내는데 걸리는 시간 복잡도는 O(루트N) 이었다.
- N = 백만인 경우: 루트N = 1,000
- N = 1억인 경우: 루트 N = 10,000
- 그럼, 1부터 1,000,000까지 모든 소수를 구하는데 걸리는 시간 복잡도는 몇일까?
- 각각의 수에 대해서 소수인지 아닌지 검사해야 한다.
- 각각의 수에 대해서 O(루트N)의 시간이 걸린다.
- 수는 총 N개이기 때문에, O(N루트N)이 걸린다.
- 1,000,000 * 1,000 = 1,000,000,000 = 10억 = 10초
- 너무 긴 시간이 필요하다.

에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 범위 안에 들어가는 모든 소수를 구하려면 에라토스테네스의 체를 사용한다.

1. 2부터 N까지 모든 수를 써놓는다.
2. 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.
3. 그 수는 소수이다.
4. 이제 그 수의 배수를 모두 지운다.

에라토스테네스의 체

Sieve of Eratosthenes

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

에라토스테네스의 체

Sieve of Eratosthenes

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	2	3		5		7		9	
11		13		15		17		19	
21		23		25		27		29	
31		33		35		37		39	
41		43		45		47		49	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

에라토스테네스의 체

Sieve of Eratosthenes

- 3의 배수를 지운다.

	2	**3**		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

에라토스테네스의 체

Sieve of Eratosthenes

- 5의 배수를 지운다.

	2	**3**		**5**		7			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	
		53						59	
61						67			
71		73				77		79	
		83						89	
91						97			

에라토스테네스의 체

Sieve of Eratosthenes

- 7의 배수를 지운다.

	2	**3**		**5**		**7**			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

에라토스테네스의 체

Sieve of Eratosthenes

- 11의 배수는 이미 지워져 있다.
- 2, 3, 5, 7로 인해서
- 11×11은 121로 100을 넘기 때문에
- 더 이상 수행할 필요가 없다.
- 남아있는 모든 수가 소수이다.

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

에라토스테네스의 체

Sieve of Eratosthenes

```
int p[100]; // 소수 저장
int pn=0; // 소수의 개수
bool c[101]; // 지워졌으면 true
int n = 100; // 100까지 소수
for (int i=2; i<=n; i++) {
    if (c[i] == false) {
        p[pn++] = i;
        for (int j = i*i; j<=n; j+=i) {
            c[j] = true;
        }
    }
}
```

에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 모든 소수를 구하는 것이 목표이기 때문에, 구현할 때는 바깥 for문 (i)를 N까지 돌린다.
- 안쪽 for문 (j)는 N의 크기에 따라서, i*i 또는 i*2로 바꾸는 것이 좋다.
- i = 백만인 경우 i*i는 범위를 넘어가기 때문

소수 구하기

https://www.acmicpc.net/problem/1929

- M이상 N이하 소수를 모두 출력하는 문제

소수 구하기

https://www.acmicpc.net/problem/1929

- C++: https://gist.github.com/Baekjoon/3247d67f7eb841d04e40
- Java: https://gist.github.com/Baekjoon/ed716b579054862b599d14fef8a70f1b