브루트 포스

190401(월) - 전현승

목차

- 브루트 포스
- 2309 – 일곱 난쟁이
- 1065 – 한수
- 16439 – 치킨치킨치킨
- 14500 – 테트로미노

브루트 포스

- == 그냥 다 해보기
  - 조합 가능한 모든 문자열을 하나씩 대입해 보는 방식으로 문제를 푸는 것인데, 얼핏 무식하다고 생각할 수도 있겠지만 성공한다는 가정 하에 항상 정확도 100%를 보장한다는 점에서, 자원만 충분하면 가장 무서운 방법이다. 이론적으로 가능한 모든 경우의 수를 다 검색해 보는 것이라 정확도 100%가 항상 보장되니, 암호학에서는 가장 확실한 방법으로 통용되고 있다. (나무위키)
- 완전탐색 중 기초
- BFS(큐)/비트마스크를 이용한 완전탐색, 백트래킹 (재귀), …
- 매우 쉽다
- 어려운 브루트포스 문제는 구현이 어려운 경우가 대부분

브루트 포스

- …, 자원만 충분하면 가장 무서운 방법이다. …
- 굉장히 쉬운 방법임과 동시에 시간복잡도는 최악
  - (모든 경우를 다 해보는 거니까)
- N제한이 작은 경우 생각은 해볼만함
- N제한을 잘 보고, 시간복잡도 분석이 필수

2309 – 일곱 난쟁이

- 아홉 난쟁이의 키가 주어졌을 때, 합이 100이 되는 일곱 난쟁이 찾기

- 9C7 = 36 → 브루트포스로 충분히 가능
- 7중 for문 / 2중 for문 / 재귀함수

- http://boj.kr/b539502ae9774b3b94de97819efdc0a8 (재귀)

1065 – 한수

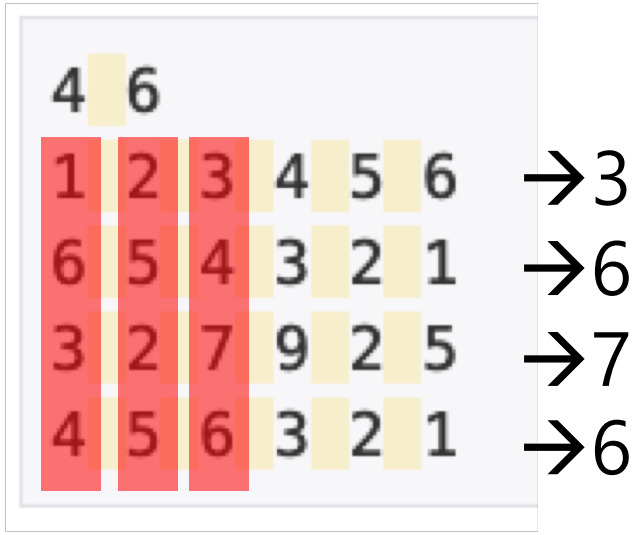
- 한수 : 자리수가 등차수열을 이루는 수
- N이 주어졌을 때, 1보다 크거나 같고, N보다 작거나 같은 한수의 개수?
  (N : 1,000보다 작거나 같은 자연수)

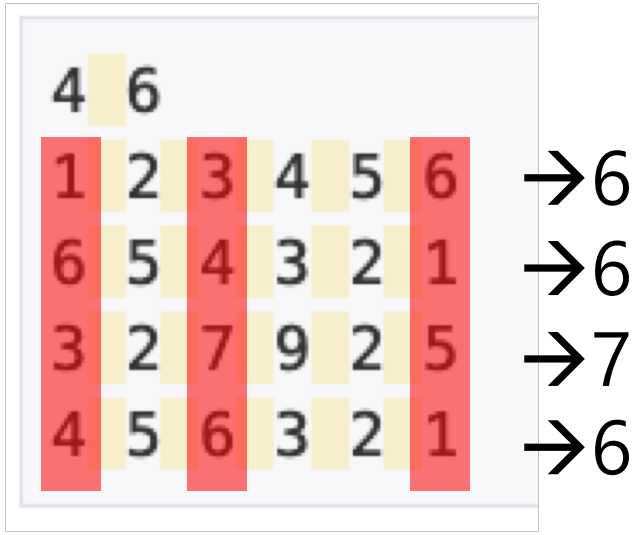
- N이 커봐야 1000이고, 한수인지 아닌지 판별은 O(1)에 가능
- 따라서 1부터 N까지 다 돌려보면 된다

- http://boj.kr/09c9c317c27c41c9abc4e1f864b7f8f1

16439 – 치킨치킨치킨

- 고리 회원 N명, 치킨 종류 M종류
- 회원마다 특정 치킨의 선호도가 있습니다. 한 사람의 만족도는 시킨 치킨 중에서 선호도가 가장 큰 값으로 결정됩니다.
- 최대 세 가지 종류만 시킬 때, 만족도 최대화





16439 – 치킨치킨치킨

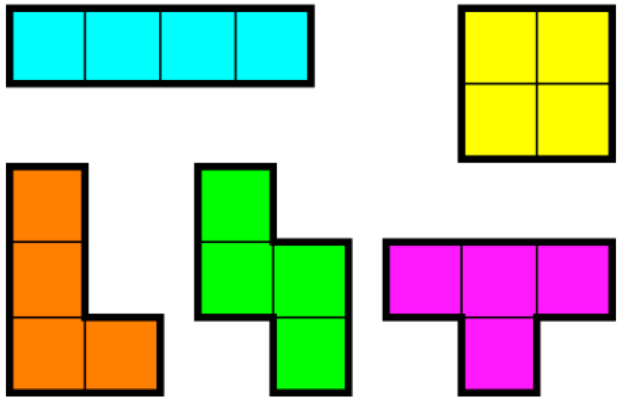
- 고리 회원의 수 N (1 ≤ N ≤ 30), 치킨 종류의 수 M (3 ≤ M ≤ 30)
- 최대 3가지 종류만 시킨다

- 30C3 = 4,060
- 3중 for문 / 재귀

- http://boj.kr/2ca3c2bfaed540d0bbe99c059b7d7ff7 (재귀)

14500 – 테트로미노

- 삼성 기출(이였다고 함)
- 크기가 N×M인 종이 위에 테트로미노 하나를 놓으려고 한다.
- 테트로미노 하나를 적절히 놓아서 테트로미노가 놓인 칸에 쓰여 있는 수들의 합을 최대로
- 회전이나 대칭 가능



예제 입력 2 복사

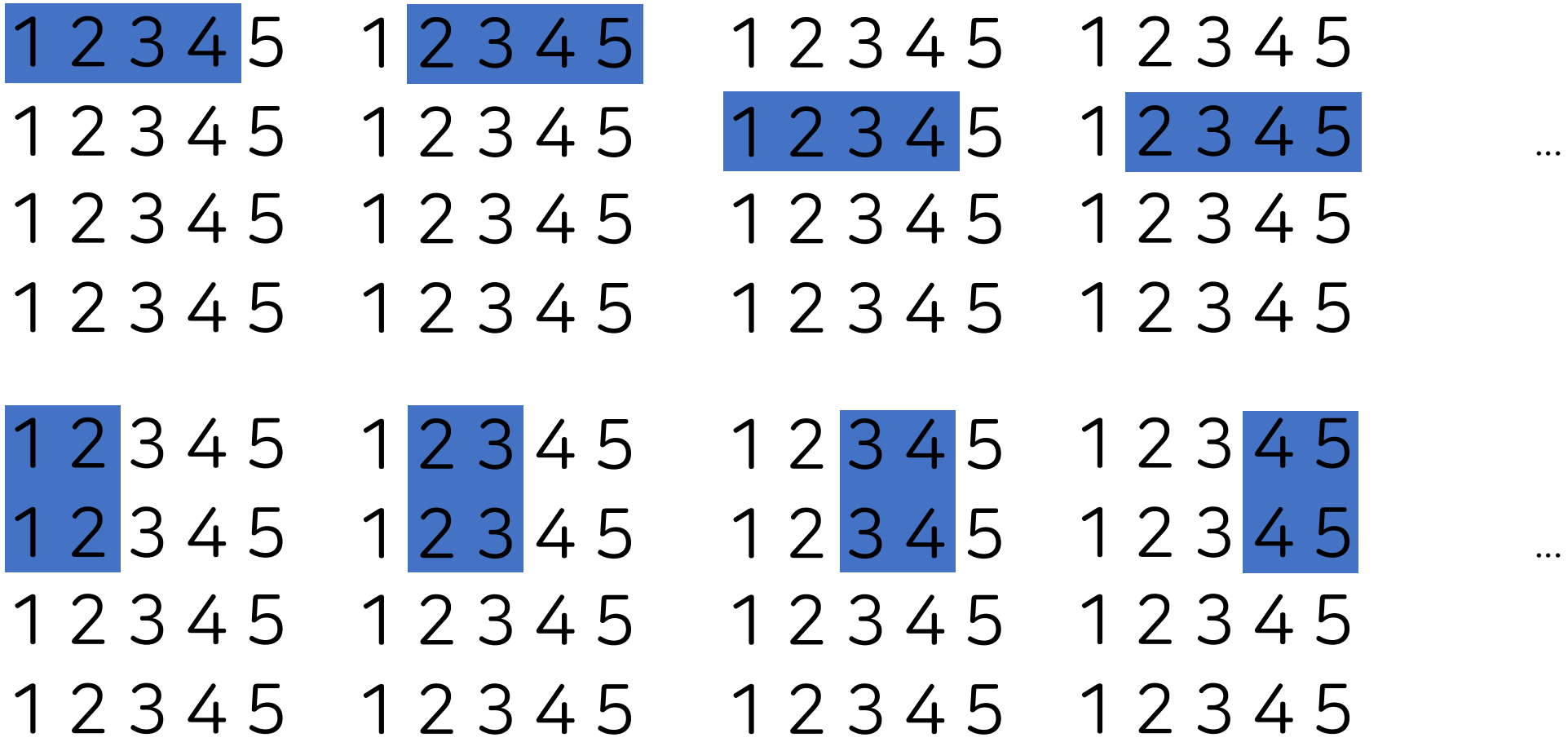
```
4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

14500 – 테트로미노

- $4 \le N, M \le 500$
- 일일이 다 놓아볼 수 있는가?

- 테트로미노 하나당 놓아볼 수 있는 가짓수 = 대략 250,000
- 총 5개 + 각각 회전 고려해도 얼마 안 됨
- 따라서 그냥 다 놓아보면 된다

14500 – 테트로미노


1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

14500 - 테트로미노

- 진짜 생짜구현

```
// 3 * 2 8종
memset(cnt_a, 0, 8);
tbt three_by_two[8] = {{{0, 0}, {1, 0}, {2, 0}, {2, 1}},
                       {{0, 1}, {1, 1}, {2, 0}, {2, 1}},
                       {{0, 0}, {0, 1}, {1, 0}, {2, 0}},
                       {{0, 0}, {0, 1}, {1, 1}, {2, 1}},
                       {{0, 0}, {1, 0}, {1, 1}, {2, 1}},
                       {{0, 1}, {1, 0}, {1, 1}, {2, 0}},
                       {{0, 0}, {1, 0}, {1, 1}, {2, 0}},
                       {{0, 1}, {1, 0}, {1, 1}, {2, 1}}
};
```

```
// 2 * 3 8종
memset(cnt_a, 0, 8);
tbt two_by_three[8] = {{{0, 2}, {1, 0}, {1, 1}, {1, 2}},
                       {{0, 0}, {1, 0}, {1, 1}, {1, 2}},
                       {{0, 0}, {0, 1}, {0, 2}, {1, 2}},
                       {{0, 0}, {0, 1}, {0, 2}, {1, 0}},
                       {{0, 1}, {0, 2}, {1, 0}, {1, 1}},
                       {{0, 0}, {0, 1}, {1, 1}, {1, 2}},
                       {{0, 1}, {1, 0}, {1, 1}, {1, 2}},
                       {{0, 0}, {0, 1}, {0, 2}, {1, 1}}
};
```

- http://boj.kr/64ce0010d33c45beb9a4086dba6a99b8 (비추)
- DFS를 통해 영역당 최솟값만 빼줘도 가능 (T미노 따로 구현)

끝