슬라이드

최백준 choi@startlink.io

부르트 포스

부르트 포스

Brute Force

- 부르트 포스는 모든 경우의 수를 다 해보는 것이다.

부르트 포스

Brute Force

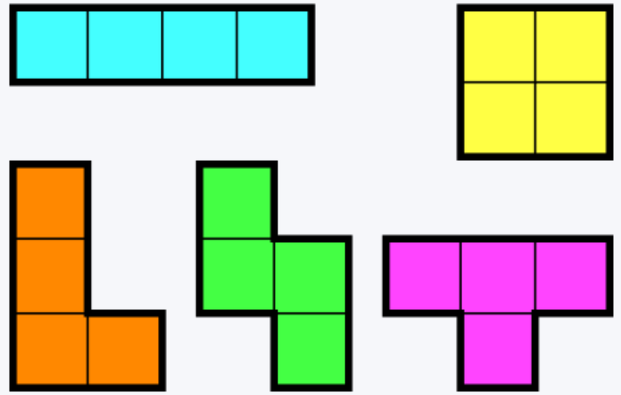
- 예를 들어, 비밀번호가 4자리이고, 숫자로만 이루어져 있다고 한다면
- 0000부터 9999까지 다 입력해보면 된다.
- 경우의 수가 10,000가지 이다.

그냥 다 해보기

테트로미노

https://www.acmicpc.net/problem/14500

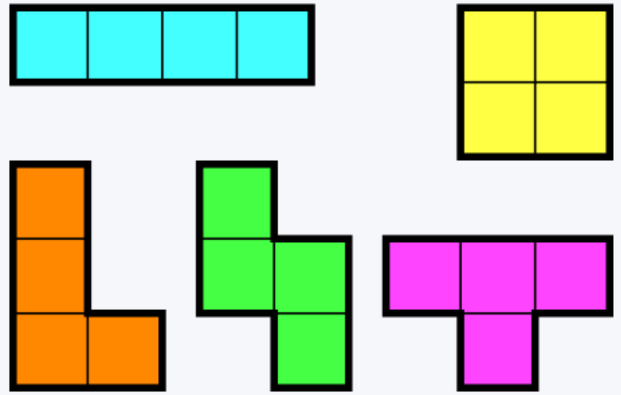
- 폴리오미노는 크기가 1 ×1인 정사각형을 여러 개 이어 붙여서 만든 도형이다.
- 정사각형 4개를 이어 붙인 폴리오미노는 테트로미노라고 하며, 총 5가지가 있다.
- N×M 크기의 종이 위에 테트로미노를 하나 놓아서
- 놓인 칸에 쓰여 있는 수의 합을 최대로 하는 문제
- $4 \leq N, M \leq 500$



테트로미노

https://www.acmicpc.net/problem/14500

- 테트로미노는 총 19가지가 있고
- 하나의 테트로미노당 놓을 수 있는 방법의 개수는 약, O(NM) 가지 이다
- 경우의 수가 많지 않기 때문에
- 각각의 테트로미노에 대해서 모든 칸에 놓아본다



테트로미노

https://www.acmicpc.net/problem/14500

- 소스: http://boj.kr/8dbe85516ab848b1afa385741c02c729

순열 사용하기

순열

Permutation

- 1 ~ N 까지로 이루어진 수열
- 1 2 3
- 4 1 3 2
- 5 4 2 3 1
- 6 5 1 2 3 4
- 크기는 항상 N이 되어야 하고, 겹치는 숫자가 존재하지 않음

순열

Permutation

- 크기가 N인 순열은 총 N!개가 존재한다
- 순열을 사전순으로 나열했을 때
- N = 3인 경우에 사전순은 다음과 같다
- 1 2 3
- 1 3 2
- 2 1 3
- 2 3 1
- 3 1 2
- 3 2 1

다음 순열

Next Permutation

- 순열을 사전순으로 나열했을 때, 사전순으로 다음에 오는 순열과 이전에 오는 순열을 찾는 방법
- C++ STL의 algorithm에는 이미 next_permutation과 prev_permutation이 존재하기 때문에 사용하면 된다

다음 순열

Next Permutation

1. A[i-1] < A[i] 를 만족하는 가장 큰 i를 찾는다
2. j ≥ i 이면서 A[j] > A[i-1] 를 만족하는 가장 큰 j를 찾는다
3. A[i-1]과 A[j]를 swap 한다
4. A[i]부터 순열을 뒤집는다

다음 순열

Next Permutation

- 순열: 7 2 3 6 5 4 1

- A[i-1] < A[i] 를 만족하는 가장 큰 i를 찾는다
- 즉, 순열의 마지막 수에서 끝나는 가장 긴 감소수열을 찾아야 한다

- 순열: 7 2 3 6 5 4 1

다음 순열

Next Permutation

- 순열: 7 2 3 6 5 4 1

- j ≥ i 이면서 A[j] > A[i-1] 를 만족하는 가장 큰 j를 찾는다

- 순열: 7 2 3 6 5 4 1

다음 순열

Next Permutation

- 순열: 7 2 3 6 5 4 1

- A[i-1]과 A[j]를 swap 한다

- 순열: 7 2 4 6 5 3 1

다음 순열

Next Permutation

- 순열: 7 2 4 6 5 3 1

- A[i]부터 순열을 뒤집는다

- 순열: 7 2 4 1 3 5 6

다음 순열

Next Permutation


```
bool next_permutation(int *a, int n) {
    int i = n-1;
    while (i > 0 && a[i-1] >= a[i]) i -= 1;
    if (i <= 0) return false; // 마지막 순열
    int j = n-1;
    while (a[j] <= a[i-1]) j -= 1;
    swap(a[i-1], a[j]);
    j = n-1;
    while (i < j) {
        swap(a[i], a[j]);
        i += 1; j -= 1;
    }
    return true;
}
```

다음 순열

https://www.acmicpc.net/problem/10972

- 다음 순열을 구하는 문제

다음 순열

https://www.acmicpc.net/problem/10972

- 소스: http://boj.kr/40e85bf7267e48f79070b5f138a70f53

이전 순열

https://www.acmicpc.net/problem/10973

- 이전 순열을 구하는 문제

이전 순열

https://www.acmicpc.net/problem/10973

- 소스: http://boj.kr/b3d907df6aac46cea5006d944b1b5ca3

모든 순열

https://www.acmicpc.net/problem/10974

- 모든 순열을 구하는 문제

모든 순열

https://www.acmicpc.net/problem/10974

- 소스: http://boj.kr/8aa0322cf54c4c40a4dcfee6f89ba4d1

팩토리얼

Factorial

- 3! = 6
- 4! = 24
- 5! = 120
- 6! = 720
- 7! = 5,040
- 8! = 40,320
- 9! = 362,880
- 10! = 3,628,800
- 11! = 39,916,800
- 12! = 479,001,600
- 13! = 6,227,020,800

연산자 끼워넣기

https://www.acmicpc.net/problem/14888

- N개의 수로 이루어진 수열과 N-1개의 연산자가 있다. ($2 \leq N \leq 11$)
- 이 때, 수와 수 사이에 연산자를 하나씩 끼워넣어서 만들 수 있는 수식 결과의 최대값과 최소값을 구하는 문제
- 수식의 계산은 연산자 우선순위를 무시하고 앞에서부터 진행한다
- 수의 순서는 바꿀 수 없다

연산자 끼워넣기

https://www.acmicpc.net/problem/14888

- 수열 = [1, 2, 3, 4, 5, 6], 연산자 = +2개, -1개, ×1개, ÷1개인 경우
- 60가지가 가능하다
- 1+2+3-4×5÷6 = 1
- 1÷2+3+4-5×6 = 12
- 1+2÷3×4-5+6 = 5
- 1÷2×3-4+5+6 = 7

연산자 끼워넣기

https://www.acmicpc.net/problem/14888

- N ≤ 11이고, 연산자는 최대 10개이기 때문에, N!가지로 모든 경우의 수를 순회해본다.

연산자 끼워넣기

https://www.acmicpc.net/problem/14888

- 소스: http://boj.kr/1507d6523f924a02b2df86e5cbfeacba

스타트와 링크

https://www.acmicpc.net/problem/14889

- N명을 N/2명씩 두 팀으로 나누려고 한다. ($4 \le N \le 20$, N은 짝수)
- 두 팀의 능력치를 구한 다음, 차이의 최소값을 구하는 문제
- S[i][j] = i번 사람과 j번 사람이 같은 팀에 속했을 때, 팀에 더해지는 능력치
- 팀의 능력치: 팀에 속한 모든 쌍의 S[i][j]의 합

스타트와 링크

https://www.acmicpc.net/problem/14889

- N명을 N/2명씩 두 팀으로 나누려고 한다. ($4 \le N \le 20$, N은 짝수)
- 이것은, 0을 N/2개, 1을 N/2개 넣어서 모든 순열을 구하는 것으로 다 해볼 수 있다.

스타트와 링크

https://www.acmicpc.net/problem/14889

- 소스: http://boj.kr/28dafbcab6514eba8175af2f93bff8b7

재귀호출 사용하기

재귀함수 사용하기

Recursion

- 재귀함수를 잘 설계해야 한다

퇴사

https://www.acmicpc.net/problem/14501

- N+1일이 되는 날 퇴사를 하려고 한다 ($1 \le N \le 15$)
- 남은 N일 동안 최대한 많은 상담을 하려고 한다
- 하루에 하나의 상담을 할 수 있고
- i일에 상담을 하면, T[i]일이 걸리고 P[i]원을 번다

퇴사

https://www.acmicpc.net/problem/14501

- go(day, sum)
  - day일이 되었다. day일에 있는 상담을 할지 말지 결정해야 한다.
  - 지금까지 얻은 수익은 sum이다

퇴사

https://www.acmicpc.net/problem/14501

- go(day, sum)
  - day일이 되었다. day일에 있는 상담을 할지 말지 결정해야 한다.
  - 지금까지 얻은 수익은 sum이다
- 정답을 찾은 경우
  - day == n
- 불가능한 경우
  - day > n
- 다음 경우
  - 상담을 한다: go(day+t[day], sum+p[day])
  - 상담을 하지 않는다: go(day+1, sum)

퇴사

https://www.acmicpc.net/problem/14501

- 소스: http://boj.kr/72def5e2dc514cd0be6210b4796d31e8

비트마스크 사용하기

비트마스크

Bitmask

- 비트(bit) 연산을 사용해서 부분 집합을 표현할 수 있다.

비트 연산

bitwise operation

- & (and), | (or), ~ (not), ^ (xor)

A	B	~A	A & B	A | B	A ^ B
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

비트 연산

bitwise operation

- 두 수 A와 B를 비트 연산 하는 경우에는 가장 뒤의 자리부터 하나씩 연산을 수행하면 된다.
- A = 27, B = 83인 경우
- $A = 11011_2$, $B = 1010011_2$
- A & B = 19, A | B = 91, A ^ B = 73

```
  0 0 1 1 0 1 1      0 0 1 1 0 1 1      0 0 1 1 0 1 1
& 1 0 1 0 0 1 1    | 1 0 1 0 0 1 1    ^ 1 0 1 0 0 1 1
---------------    ---------------    ---------------
  0 0 1 0 0 1 1      1 0 1 1 0 1 1      1 0 0 1 0 0 0
```

비트 연산

bitwise operation

- not 연산의 경우에는 자료형에 따라 결과가 달라진다.
- A = 83 = 1010011_2
- ~A = 10101100_2 (8비트 자료형인 경우)
- ~A = $11111111\ 11111111\ 11111111\ 10101100_2$ (32비트 자료형인 경우)
- 또, unsigned, signed에 따라서 보여지는 값은 다르다.

비트 연산

bitwise operation

- shift left (<<) 와 shift right (>>) 연산이 있다.
- A << B (A를 왼쪽으로 B비트만큼 민다.)
- `1 << 0 = 1`
- `1 << 1 = 2` (10_2)
- `1 << 2 = 4` (100_2)
- `1 << 3 = 8` (1000_2)
- `1 << 4 = 16` (10000_2)
- `3 << 3 = 24` (11000_2)
- `5 << 10 = 5120` (1010000000000_2)

비트 연산

bitwise operation

- shift left (<<) 와 shift right (>>) 연산이 있다.
- A >> B (A를 오른쪽으로 B비트만큼 민다.)
- `1 >> 0 = 1`
- `1 >> 1 = 0` (0_2)
- `10 >> 1 = 5` (101_2)
- `10 >> 2 = 2` (10_2)
- `10 >> 3 = 1` (1_2)
- `30 >> 1 = 15` (1111_2)
- `1024 >> 10 = 1` (1_2)

비트 연산

bitwise operation

- A << B는 $A \times 2^B$와 같다.
- A >> B는 $A / 2^B$와 같다.
- (A + B) / 2는 (A+B) >> 1로 쓸 수 있다.

비트마스크

Bitmask

- 정수로 집합을 나타낼 수 있다.
- $\{1, 3, 4, 5, 9\} = 570 = 2^1 + 2^3 + 2^4 + 2^5 + 2^9$

비트마스크

Bitmask

- 보통 0부터 N-1까지 정수로 이루어진 집합을 나타낼 때 사용한다.

비트마스크

Bitmask

- {1, 3, 4, 5, 9} = 570
- 0이 포함되어 있는지 검사
  - `570 & 2^0 = 570 & (1<<0) = 0`
- 1이 포함되어 있는지 검사
  - `570 & 2^1 = 570 & (1<<1) = 2`
- 2이 포함되어 있는지 검사
  - `570 & 2^2 = 570 & (1<<2) = 0`
- 3이 포함되어 있는지 검사
  - `570 & 2^3 = 570 & (1<<3) = 8`

```
  1000111010
& 0000000100
--------------
  0000000000

  1000111010
& 0000001000
--------------
  0000001000
```

비트마스크

Bitmask

- {1, 3, 4, 5, 9} = 570
- 1 추가하기
  - `570 | 2^1 = 570 | (1<<1) = 570 (1000111010_2)`
- 2 추가하기
  - `570 | 2^2 = 570 | (1<<2) = 574 (1000111110_2)`
- 3 추가하기
  - `574 | 2^3 = 570 | (1<<3) = 570 (1000111010_2)`
- 4 추가하기
  - `574 | 2^4 = 570 | (1<<4) = 570 (1000111010_2)`

```
  1000111010
| 0000000100
--------------
  1000111110

  1000111010
| 0000001000
--------------
  1000111010
```

비트마스크

Bitmask

- {1, 3, 4, 5, 9} = 570
- 1 제거하기
  - $570 \ \& \sim 2^1 = 570 \ \& \sim(1<<1) = 568\ (1000111000_2)$
- 2 제거하기
  - $570 \ \& \sim 2^2 = 570 \ \& \sim(1<<2) = 570\ (1000111010_2)$
- 3 제거하기
  - $562 \ \& \sim 2^3 = 562 \ \& \sim(1<<3) = 562\ (1000110010_2)$
- 4 제거하기
  - $562 \ \& \sim 2^4 = 562 \ \& \sim(1<<4) = 546\ (1000101010_2)$

```
  1000111010
& 1111110111
--------------
  1000110010

  1000111010
| 1111111011
--------------
  1000111010
```

비트마스크

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$
- 1 토글하기
  - $570 \text{ ^ } 2^1 = 570 \text{ ^ } (1<<1) = 568\ (1000111000_2)$

```
  1000111010
^ 0000000100
--------------
  1000111110
```

- 2 토글하기
  - $570 \text{ ^ } 2^2 = 570 \text{ ^ } (1<<2) = 574\ (1000111110_2)$
- 3 토글하기
  - $574 \text{ ^ } 2^3 = 570 \text{ ^ } (1<<3) = 562\ (1000110010_2)$

```
  1000111010
^ 0000001000
--------------
  1000110010
```

- 4 추가하기
  - $574 \text{ ^ } 2^4 = 570 \text{ ^ } (1<<4) = 554\ (1000101010_2)$

비트마스크

Bitmask

- 전체 집합
  - `(1 << N) - 1`
- 공집합
  - `0`

비트마스크

Bitmask

- 현재 집합이 S일때
- i를 추가
  - `S | (1 << i)`
- i를 검사
  - `S & (1 << i)`
- i를 제거
  - `S & ~(1 << i)`
- i를 토글 (0을 1로, 1을 0으로)
  - `S ^ (1 << i)`

째로탈출 2

https://www.acmicpc.net/problem/13460

- 보드의 상태가 주어졌을 때, 최소 몇 번 만에 빨간 구슬을 구멍을 통해 빼낼 수 있는지 구하는 문제
- 만약, 10번 이내에 움직여서 빨간 구슬을 구멍을 통해 빼낼 수 없으면 -1을 출력

째로탈출 2

https://www.acmicpc.net/problem/13460

- 보드의 상태가 주어졌을 때, 최소 몇 번 만에 빨간 구슬을 구멍을 통해 빼낼 수 있는지 구하는 문제
- 만약, **10번 이내**에 움직여서 빨간 구슬을 구멍을 통해 빼낼 수 없으면 -1을 출력

째로탈출 2

https://www.acmicpc.net/problem/13460

- 이동할 수 있는 방향이 4방향
- 최대 10번 이내로 움직여야 한다
- 가능한 이동 방법의 수: 4^10 = 1,048,576

째로탈출 2

https://www.acmicpc.net/problem/13460

- 같은 방향으로 연속해서 두 번 이상 이동하는건 의미가 없다
- 가능한 이동 방법의 수: 4 * 3^9 = 78,732가지

째로탈출 2

https://www.acmicpc.net/problem/13460

- 같은 방향으로 연속해서 두 번 이상 이동하는건 의미가 없다
- 한 방향으로 이동한 다음, 반대 방향으로 바로 이동하는 것도 의미가 없다
- 가능한 이동 방법의 수: 4 * 2^9 = 2,048가지

째로탈출 2

https://www.acmicpc.net/problem/13460

- 먼저, 이동 가능한 방법을 비트마스크를 이용해서 4^10가지를 만든 다음
- 앞 페이지에 나온 두 가지 경우를 모두 제외시킨다
- 4^10을 만들기 위해 0부터 2^20까지 수를 모두 만들고
- 4진법으로 변환해서 경우의 수를 모두 만든다

째로탈출 2

https://www.acmicpc.net/problem/13460

- 그 다음, 문제에 나와있는 대로 시뮬레이션 해본다.
- 동시에 두 개의 공을 이동시키는 것은 어렵기 때문에
- 공을 하나씩 움직여서 더 이상 두 공이 움직이지 않을 때 까지 이동시켜본다

째로탈출 2

https://www.acmicpc.net/problem/13460

- 소스: http://boj.kr/56f8db0fd448461c8670d33c90543b08

2048 (Easy)

https://www.acmicpc.net/problem/12100

- 2048 게임에서 최대 5번 이동시켜서 얻을 수 있는 가장 큰 블럭을 출력하는 문제

2048 (Easy)

https://www.acmicpc.net/problem/12100

- 이동 횟수가 5번이기 때문에, 4^5 = 1024번 이동을 시켜보면 된다.

2048 (Easy)

https://www.acmicpc.net/problem/12100

- 소스: http://boj.kr/255a8fed524b4d3abf4a90b71aa0fd83