# 4/5 DFS

최백준 choi@startlink.io

# 비트마스크

# 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.
- 어떤 수가 홀수 인지 판별하는 if (N % 2 == 1) 은 if (N & 1)로 줄여 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



- $\{1, 3, 4, 5, 9\} = 570$
- 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
  - $570 \ \& \ 2^3 = 570 \ \& \ (1<<3) = 8$

# 비트마스크

Bitmask



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

# 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
  - $562 \ \& \sim 2^4 = 562 \ \& \sim(1<<4) = 546 \ (1000101010_2)$

# 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)`

# 집합



https://www.acmicpc.net/problem/11723

- 비트마스크를 연습해보는 문제

# 집합



https://www.acmicpc.net/problem/11723

- C++: https://gist.github.com/Baekjoon/3503aaa55c03cdde9df51b1bd5155486
- Java: https://gist.github.com/Baekjoon/2da9289baa79449207eed6c2013f8c41

# 비트마스크

Bitmask



- 물론 배열을 사용하는 것이 더욱 편리하지만, 비트마스크를 사용하는 이유는
- 집합을 배열의 인덱스로 표현할 수 있기 때문이다.
- 상태 다이나믹을 할 때 자주 사용하게 된다.

# bitset

bitset



- 비트마스크는 STL의 bitset을 이용해서 더 쉽게 나타낼 수 있다.

# 순열

# 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

- C++: https://gist.github.com/Baekjoon/d51fbc6f75332cfc6ab9
- C++ (next_permutation 구현): https://gist.github.com/Baekjoon/f8d9765ccde7262744b5
- Java: https://gist.github.com/Baekjoon/c307fc69373a74a730c0

# 이전 순열



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

- 이전 순열을 구하는 문제

# 이전 순열



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

- C++: https://gist.github.com/Baekjoon/2db36900d5b1f37b2397
- C++ (prev_permutation 구현): https://gist.github.com/Baekjoon/c3b6e4a24b3841575dc9
- Java: https://gist.github.com/Baekjoon/37eda7e437c1d14092aa

# 모든 순열



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

- 모든 순열을 구하는 문제

# 모든 순열



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

- C++: https://gist.github.com/Baekjoon/8c1c89872b713e45d45d
- Java: https://gist.github.com/Baekjoon/bb4679d85dd726fd3456

# 문제

# 외판원 순회 2



https://www.acmicpc.net/problem/10971

- 영어로 Travelling Salesman Problem (TSP)
- 1번부터 N번까지 번호가 매겨져있는 도시가 있다
- 한 도시에서 시작해 N개의 모든 도시를 거쳐 다시 원래 도시로 돌아오려고 한다 (한 번 갔던 도시로는 다시 갈 수 없다)
- 이 때, 가장 적은 비용을 구하는 문제

- W[i][j] = i -> j 비용

# 외판원 순회 2



https://www.acmicpc.net/problem/10971

- $2 \leq N \leq 10$
- N! = 10! = 3628800
- 모든 경우를 다해봐도 시간 안에 나온다

# 외판원 순회 2



https://www.acmicpc.net/problem/10971

- $2 \leq N \leq 10$
- N! = 10! = 3628800
- 모든 경우를 다해봐도 시간 안에 나온다
- 모든 경우 = N!
  - 비용 계산 = N
- 시간복잡도: O(N*N!)

# 외판원 순회 2



https://www.acmicpc.net/problem/10971

```
do {
    bool ok = true;
    int sum = 0;
    for (int i=0; i<n-1; i++) {
        if (w[d[i]][d[i+1]] == 0) ok = false;
        else sum += w[d[i]][d[i+1]];
    }
    if (ok && w[d[n-1]][d[0]] != 0) {
        sum += w[d[n-1]][d[0]];
        if (ans > sum) ans = sum;
    }
} while (next_permutation(d.begin(), d.end()));
```

# 외판원 순회 2



https://www.acmicpc.net/problem/10971

- O(N*N!)
- C++: https://gist.github.com/Baekjoon/a62f0b1263752c8d1a75
- Java: https://gist.github.com/Baekjoon/a5450f44bc19da72f9ac

- O(N!)
- C++: https://gist.github.com/Baekjoon/3eeee9003b22cffb2a76
- C++ 2: https://gist.github.com/Baekjoon/45c47a211c3be61e054a
- Java: https://gist.github.com/Baekjoon/88bfb6c2e54bb399beb2

# 1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- 정수 n을 1, 2, 3의 조합으로 나타내는 방법의 수를 구하는 문제
- n = 4
- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

# 1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수

# 1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수
- 불가능한 경우
  - count > 10
  - sum > goal
- 가능한 경우
  - sum == goal

# 1, 2, 3 더하기



https://www.acmicpc.net/problem/9095

- go(count, sum, goal)
- 숫자 count개로 합 sum을 만드는 경우의 수
- 다음 경우
  - 1을 사용하는 경우
    - go(count+1, sum+1, goal)
  - 2를 사용하는 경우
    - go(count+1, sum+2, goal)
  - 3을 사용하는 경우
    - go(count+1, sum+3, goal)

# 1, 2, 3 더하기



https://www.acmicpc.net/problem/9095

```
int go(int count, int sum, int goal) {
    if (count > 10) return 0;
    if (sum > goal) return 0;
    if (sum == goal) return 1;
    int now = 0;
    for (int i=1; i<=3; i++) {
        now += go(count+1, sum+i, goal);
    }
    return now;
}
```

# 1, 2, 3 더하기



https://www.acmicpc.net/problem/9095

- C++: https://gist.github.com/Baekjoon/3235f76fe44c1ad17648
- Java: https://gist.github.com/Baekjoon/bdeba307e9e6d1e80fc7

# 암호 만들기



https://www.acmicpc.net/problem/1759

- 암호는 서로 다른 L개의 알파벳 소문자들로 구성되며 최소 한 개의 모음과 최소 두 개의 자음으로 구성되어 있다
- 암호를 이루는 알파벳이 암호에서 증가하는 순서로 배열되었어야 한다
- 암호로 사용할 수 있는 문자의 종류는 C가지
- 가능성 있는 암호를 모두 구하는 문제

# 암호 만들기



https://www.acmicpc.net/problem/1759

- L = 4, C = 6
- 사용 가능한 알파벳: a t c i s w
- 가능한 암호
- acis
- acit
- aciw
- acst
- acsw
- actw
- aist
- aisw
- aitw
- astw
- cist
- cisw
- citw
- istw

# 암호 만들기



https://www.acmicpc.net/problem/1759

- go(n, alpha, password, i)
  - n: 만들어야 하는 암호의 길이
  - alpha: 사용할 수 있는 알파벳
  - password: 현재까지 만든 암호
  - i: 사용할지 말지 결정해야 하는 알파벳의 인덱스

# 암호 만들기



https://www.acmicpc.net/problem/1759

- go(n, alpha, password, i)
  - n: 만들어야 하는 암호의 길이
  - alpha: 사용할 수 있는 알파벳
  - password: 현재까지 만든 암호
  - i: 사용할지 말지 결정해야 하는 알파벳의 인덱스
- 언제 답인지 아닌지 확인해야 하나?
  - n == password.length()
- 다음
  - i번째 알파벳을 사용하는 경우
  - i번째 알파벳을 사용하지 않는 경우

# 암호 만들기



https://www.acmicpc.net/problem/1759

- 다음
  - i번째 알파벳을 사용하는 경우
    - go(n, alpha, password+alpha[i], i+1)
  - i번째 알파벳을 사용하지 않는 경우
    - go(n, alpha, password, i+1)

# 암호 만들기



https://www.acmicpc.net/problem/1759

```
void go(int n, vector<char> &alpha, string password, int i) {
    if (password.length() == n) {
        if (check(password)) {
            cout << password << '\n';
        }
        return;
    }
    if (i >= alpha.size()) return;
    go(n, alpha, password+alpha[i], i+1);
    go(n, alpha, password, i+1);
}
```

# 암호 만들기



https://www.acmicpc.net/problem/1759

```
bool check(string &password) {
    int ja = 0;
    int mo = 0;
    for (char x : password) {
        if (x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u') {
            mo += 1;
        } else {
            ja += 1;
        }
    }
    return ja >= 2 && mo >= 1;
}
```

# 암호 만들기



https://www.acmicpc.net/problem/1759

- C++: https://gist.github.com/Baekjoon/dff42ddf0ae028f6b7f1
- Java: https://gist.github.com/Baekjoon/e92cfec2c020cd62b8ef

# N-Queen

https://www.acmicpc.net/problem/9663

- N*N크기의 체스판 위에 Queen을 N개 놓는 방법의 수를 구하는 문제

# N-Queen



https://www.acmicpc.net/problem/9663

- calc(row): row 행에 퀸을 어디에 놓을지 결정해야 함

# N-Queen



https://www.acmicpc.net/problem/9663

- calc(row): row 행에 퀸을 어디에 놓을지 결정해야 함

```
void calc(int row) {
    if (row == n) {
        ans += 1;
    }
    for (int col=0; col<n; col++) {
        a[row][col] = true;
        if (check(row, col)) {
            calc(row+1);
        }
        a[row][col] = false;
    }
}
```

# N-Queen



https://www.acmicpc.net/problem/9663

- C++: https://gist.github.com/Baekjoon/1945a35cb532d5d294768d89822fbbfe
- Java: https://gist.github.com/Baekjoon/3f8faef559a0a37cc2fd42f28b2bb184

# N-Queen

https://www.acmicpc.net/problem/9663

- check_col[i] = i번 열에 퀸이 놓여져 있으면 true

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

# N-Queen

https://www.acmicpc.net/problem/9663

- check_dig[i] = / 대각선에 퀸이 있으면

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 5 | 6 | 7 | 8 | 9 | 10 |

# N-Queen

https://www.acmicpc.net/problem/9663

- check_dig2[i] = \ 대각선에 퀸이 있으면

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | 6 | 5 | 4 | 3 | 2 | 1 |
| 2 | 7 | 6 | 5 | 4 | 3 | 2 |
| 3 | 8 | 7 | 6 | 5 | 4 | 3 |
| 4 | 9 | 8 | 7 | 6 | 5 | 4 |
| 5 | 10 | 9 | 8 | 7 | 6 | 5 |

# N-Queen

https://www.acmicpc.net/problem/9663

- Check 부분을 배열을 이용하면 O(1)만에 해결 할 수 있다.
- C++: https://gist.github.com/Baekjoon/7ce9963ec6d292ad9cfd2aeb3face717
- Java: https://gist.github.com/Baekjoon/4d69907aad8adc3bf166c5ec8d8ad3b6

# 알파벳



https://www.acmicpc.net/problem/1987

- 세로 R칸, 가로 C칸으로 된 표 모양의 보드가 있다
- 보드의 각 칸에는 대문자 알파벳이 하나씩 적혀 있고, 좌측 상단 칸 (1행 1열) 에는 말이 놓여 있다
- 말은 상하좌우로 인접한 네 칸 중의 한 칸으로 이동할 수 있다
- 같은 알파벳이 적힌 칸을 두 번 지날 수 없다
- 좌측 상단에서 시작해서, 말이 최대한 몇 칸을 지날 수 있는지를 구하는 문제

# 알파벳



https://www.acmicpc.net/problem/1987

- go(board, check, x, y, cnt)
  - board: 보드
  - check: 방문한 알파벳
  - x, y: 현재 위치
  - cnt: 방문한 칸의 수

# 알파벳



https://www.acmicpc.net/problem/1987

- go(board, check, x, y, cnt)
  - board: 보드
  - check: 방문한 알파벳
  - x, y: 현재 위치
  - cnt: 방문한 칸의 수
- 새로운 칸 nx, ny로 이동할 수 있는 경우
  - go(board, check, nx, ny, cnt+1)
    - 이 때, check는 변경해 줘야함

# 알파벳



https://www.acmicpc.net/problem/1987

```
void go(vector<string> &board, vector<bool> &check, int x, int y, int
cnt) {
    if (cnt > ans) ans = cnt;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k];
        int ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                go(board, check, nx, ny, cnt+1);
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
}
```

# 알파벳



https://www.acmicpc.net/problem/1987

- go(board, check, x, y)
- board: 보드
- check: 방문한 알파벳
- x, y: 현재 위치
- 리턴 값: 방문할 수 있는 칸의 최대 개수
- 의미: (x, y)에서 이동을 시작하고, 방문한 알파벳이 check일 때, 방문할 수 있는 칸의 최대 개수

# 알파벳



https://www.acmicpc.net/problem/1987

```
int go(vector<string> &board, vector<bool> &check, int x, int y) {
    int ans = 0;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k], ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                int next = go(board, check, nx, ny);
                if (ans < next) ans = next;
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
    return ans + 1;
}
```

# 알파벳



https://www.acmicpc.net/problem/1987

- C++: https://gist.github.com/Baekjoon/f412bcc16f3b3f0cbffd
- Java: https://gist.github.com/Baekjoon/411767759d38830b5911

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- C++: https://gist.github.com/Baekjoon/5d90f9d1582559c619ad2821b126ac16
- Java: https://gist.github.com/Baekjoon/923eddd3d8d3bef43372433c83afb6cf

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 모든 집합의 개수 = 2^N
- 모든 집합을 구해보면 된다!

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1

```
for (int i=0; i<(1<<n); i++) {

}
```

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1
- 공집합은 제외해야 한다

```
for (int i=1; i<(1<<n); i++) {

}
```

# 부분집합의 합



https://www.acmicpc.net/problem/1182

- 전체 집합 = (1<<N) – 1
- 공집합은 제외해야 한다
- 집합에 무엇이 포함되어 있는지 확인하기

```
for (int i=1; i<(1<<n); i++) {
    for (int k=0; k<n; k++) {
        if (i&(1<<k)) {
        }
    }
}
```

# 부분집합의 합



https://www.acmicpc.net/problem/1182

```
for (int i=1; i<(1<<n); i++) {
    int sum = 0;
    for (int k=0; k<n; k++) {
        if (i&(1<<k)) {
            sum += a[k];
        }
    }
    if (sum == s) {
        ans += 1;
    }
}
```

# 부분집합의 합



https://www.acmicpc.net/problem/1182

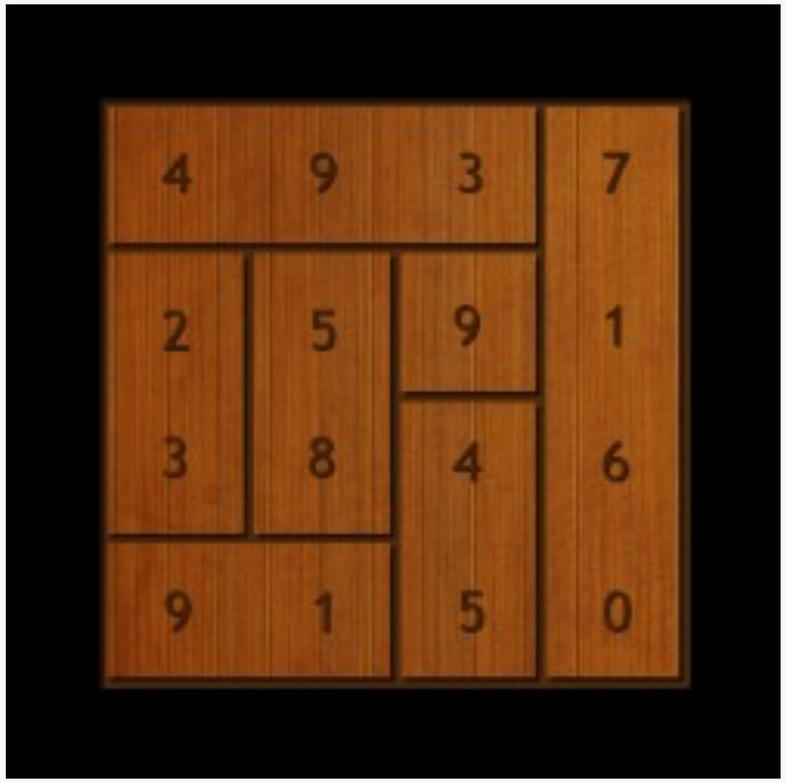
- C++: https://gist.github.com/Baekjoon/f4154089addcd1adacc5
- Java: https://gist.github.com/Baekjoon/bddda372acf45d698817

# 종이 조각



https://www.acmicpc.net/problem/14391

- 종이를 조각으로 잘라서 합의 최대값을 구하는 문제

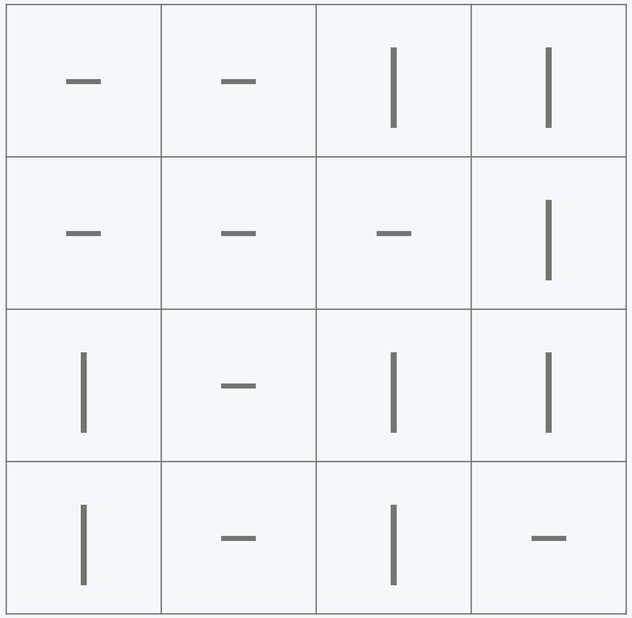


# 종이 조각



https://www.acmicpc.net/problem/14391

- 각각의 칸에 대해서, 가로(-)인지 세로(|)인지 정하면 된다.

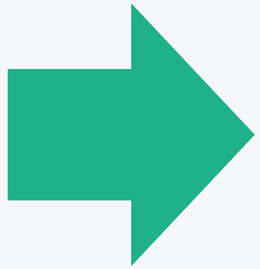


# 종이 조각

https://www.acmicpc.net/problem/14391

- 각각의 칸에 대해서, 가로(-)인지 세로(|)인지 정하면 된다.

| | | | |
|---|---|---|---|
| – | – | \| | \| |
| – | – | – | \| |
| \| | – | \| | \| |
| \| | – | \| | – |



| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |

# 종이 조각

https://www.acmicpc.net/problem/14391

- 각각의 칸에 대해서, 가로(-)인지 세로(|)인지 정하면 된다.

| | | | |
|---|---|---|---|
| – | – | \| | \| |
| – | – | – | \| |
| \| | – | \| | \| |
| \| | – | \| | – |

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |

# 종이 조각



https://www.acmicpc.net/problem/14391

- 2^(NM) 으로 상태를 만들고 나누어보면 된다

# 종이 조각



https://www.acmicpc.net/problem/14391

- C++: https://gist.github.com/Baekjoon/39614927a2cae287cfc304615de1d7f5
- Java: https://gist.github.com/Baekjoon/85e9bdc01ae9e741a331875a23726c7f

# 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

- C++: https://gist.github.com/Baekjoon/c9dbf1e5eae35c2f4501f410482c1469
- Java: https://gist.github.com/Baekjoon/d462fd8f86659be5c7244d67113c5ff6

# 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

- C++: https://gist.github.com/Baekjoon/522d138cf5ff54abc22e49374a7aeedc
- Java: https://gist.github.com/Baekjoon/4a8748c9675eb1562d744b7f87a6b1da