

브루트 포스

최백준 choi@startlink.io

브루트 포스

브루트 포스

Brute Force

3

- 브루트 포스는 모든 경우의 수를 다 해보는 것이다.

브루트 포스

Brute Force

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

브루트 포스

Brute Force

- 예를 들어, 비밀번호가 4자리이고, 숫자로만 이루어져 있다고 한다면
- 0000부터 9999까지 다 입력해보면 된다.
- 경우의 수가 10,000가지 이다.
- 사람이 직접 비밀번호를 입력하는데 1초가 걸린다면 $10,000\text{초} = 2.7\text{시간}$ 정도 걸린다.

브루트 포스

Brute Force

6

- 예를 들어, 비밀번호가 12자리이고, 숫자로만 이루어져 있다고 한다면
- 000000000000부터 999999999999까지 다 입력해보면 된다.
- 경우의 수가 1,000,000,000,000가지 이다.

브루트 포스

Brute Force

7

- 예를 들어, 비밀번호가 12자리이고, 숫자로만 이루어져 있다고 한다면
- 000000000000부터 999999999999까지 다 입력해보면 된다.
- 경우의 수가 1,000,000,000,000가지 이다.
- 사람이 직접 비밀번호를 입력하는데 1초가 걸린다면 1,000,000,000,000초 = 약 31688년이 걸린다.

브루트 포스

Brute Force

- 브루트 포스는 모든 경우의 수를 다 해보는 것이다.
- 이 때, 경우의 수를 다 해보는데 걸리는 시간이 문제의 시간 제한을 넘지 않아야 한다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.
1. 문제의 가능한 경우의 수를 계산해본다.
 2. 가능한 모든 방법을 다 만들어본다.
 3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.
 1. 문제의 가능한 경우의 수를 계산해본다.
 - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
 2. 가능한 모든 방법을 다 만들어본다.
 3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.
 1. 문제의 가능한 경우의 수를 계산해본다.
 - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
 2. 가능한 모든 방법을 다 만들어본다.
 - 하나도 빠짐 없이 만들어야 한다.
 - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
 3. 각각의 방법을 이용해 답을 구해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.
 1. 문제의 가능한 경우의 수를 계산해본다.
 - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
 2. 가능한 모든 방법을 다 만들어본다.
 - 하나도 빠짐 없이 만들어야 한다.
 - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
 3. 각각의 방법을 이용해 답을 구해본다.
 - 이 단계는 보통은 어렵지 않다. 문제에 나와있는 대로 답을 계산해본다.

브루트 포스

Brute Force

- 브루트 포스로 문제를 풀기 위해서는 다음과 같은 3가지 단계를 생각해볼 수 있다.
 1. 문제의 가능한 경우의 수를 계산해본다.
 - 직접 계산을 통해서 구한다. 대부분 손으로 계산해볼 수 있다.
 2. 가능한 모든 방법을 다 만들어본다.
 - 하나도 빠짐 없이 만들어야 한다.
 - 대표적으로 그냥 다 해보는 방법, for문 사용, 순열 사용, 재귀 호출 사용, 비트마스크 사용이 있다.
 3. 각각의 방법을 이용해 답을 구해본다.
 - 이 단계는 보통은 어렵지 않다. 문제에 나와있는 대로 답을 계산해본다.
- 브루트 포스 문제의 시간 복잡도는 대부분 $O(\text{경우의 수} * \text{방법 1개를 시도해보는데 걸리는 시간 복잡도})$ 가 걸린다.

그냥 다 해보기

일곱 난쟁이

<https://www.acmicpc.net/problem/2309>

- 아홉 명의 난쟁이 중 일곱 명의 난쟁이를 찾는 문제
- 일곱 난쟁이의 키의 합은 100이다.

일곱 난쟁이

<https://www.acmicpc.net/problem/2309>

- 아홉 명 중에 일곱 명을 고르는 것은
- 아홉 명 중에 두 명을 고르는 것과 같다.

일곱 난쟁이

<https://www.acmicpc.net/problem/2309>

- 아홉 명 중에 일곱 명을 고르는 것은
- 아홉 명 중에 두 명을 고르는 것과 같다.
- 난쟁이의 수를 N 이라고 했을 때
- 두 명을 고르는 경우의 수: N^2 라고 할 수 있다.
- 나머지 난쟁이의 키의 합을 고르는 시간 복잡도: $O(N)$

일곱 난쟁이

<https://www.acmicpc.net/problem/2309>

- 아홉 명 중에 일곱 명을 고르는 것은
- 아홉 명 중에 두 명을 고르는 것과 같다.
- 난쟁이의 수를 N 이라고 했을 때
- 두 명을 고르는 경우의 수: N^2 라고 할 수 있다.
- 나머지 난쟁이의 키의 합을 고르는 시간 복잡도: $O(N)$
- 따라서, 이 문제는 $O(N^3)$ 으로 해결할 수 있다.

일곱 난쟁이

<https://www.acmicpc.net/problem/2309>

- 소스: <http://codeplus.codes/f7443f083b8949d7a9cdcdbb411034aa>

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 준규가 사는 나라는 E S M이라는 연도를 사용한다.
- $1 \leq E \leq 15, 1 \leq S \leq 28, 1 \leq M \leq 19$
- 1년 = 1 1 1 • 17년: 2 17 17
- 2년 = 2 2 2 • 18년: 3 18 18
- ... • 19년: 4 19 19
- 15년 = 15 15 15 • 20년: 5 20 1
- 16년 = 1 16 16 • 21년: 6 21 2
- E S M이 주어졌을 때, 이게 몇 년인지 구하는 문제

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 가능한 경우의 수
- $15 \times 28 \times 19 = 7,980$
- 모든 경우를 다 해보면 된다

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 소스: <http://codeplus.codes/07de283fab5f42f99ebbecfa4bb6b88e>

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 모든 E, S, M에서 1을 빼면, 이 문제는 다음을 만족하는 가장 작은 자연수 year를 찾는 문제이다.
- $\text{year} \bmod 15 == E$
- $\text{year} \bmod 28 == S$
- $\text{year} \bmod 19 == M$
- 이런식으로 year를 0부터 증가시키면서 위의 식을 검사해 구현하는 방법도 가능하다.

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 소스: <http://codeplus.codes/30bb5b836af34fd6bd5b5ad074e914c8>

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 모든 E, S, M에서 1을 빼면, 이 문제는 다음을 만족하는 가장 작은 자연수 year를 찾는 문제이다.
- $\text{year} \bmod 15 == E$
- $\text{year} \bmod 28 == S$
- $\text{year} \bmod 19 == M$
- 이 문제는 중국인의 나머지 정리로도 풀 수 있다.
- 중국인의 나머지 정리는 이 챕터에서 중요한 내용이 아니기 때문에, 소스 코드만 첨부한다.

날짜 계산

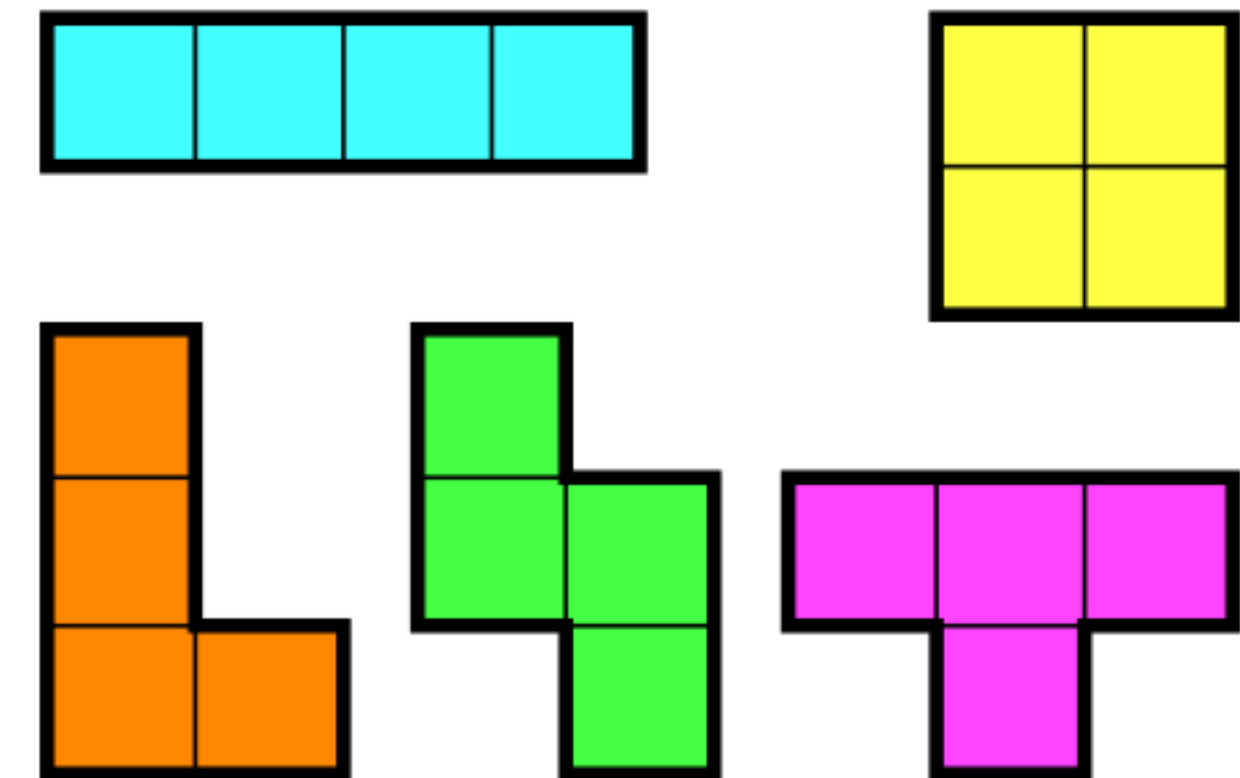
<https://www.acmicpc.net/problem/1476>

- 소스: <http://codeplus.codes/c3bab714265d4f16af30dee5312bde9c>

테트로미노

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

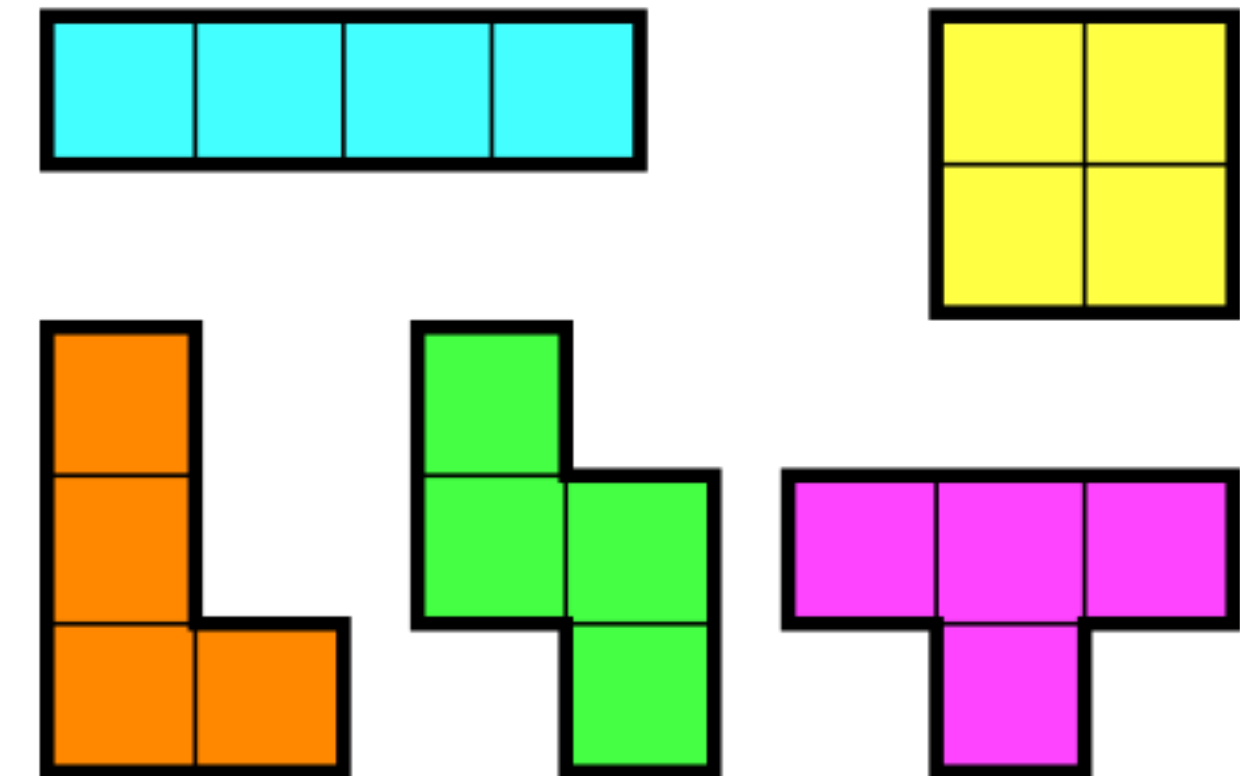
- 폴리오미노는 크기가 1×1 인 정사각형을 여러 개 이어 붙여서 만든 도형이다.
- 정사각형 4개를 이어 붙인 폴리오미노는 테트로미노라고 하며, 총 5가지가 있다.
- $N \times M$ 크기의 종이 위에 테트로미노를 하나 놓아서
- 놓인 칸에 쓰여 있는 수의 합을 최대로 하는 문제
- $4 \leq N, M \leq 500$



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://codeplus.codes/5b8ac59173604f2fb0b1dafb337b9665>
- 소스 2: <http://codeplus.codes/3693a7da43b44afd9adc8424c8a7a2d6>
- 소스 3: <http://codeplus.codes/c0053046043447e18ba0cbe8e47848b5>
- 소스 3과 같이 구현할 수도 있지만, 이 문제에서는 적절하지 않다.

N중 for문

N중 for문

for

- N개 중에 일부를 선택해야 하는 경우에 많이 사용한다
- 재귀 호출이나 비트마스크를 사용하면 더 간결하고 보기 쉬운 코드를 작성할 수 있기 때문에, 사용할 일이 거의 없다.

1, 2, 3 더하기

32

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

- 정수 n 을 1, 2, 3의 합으로 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>

- N이 10보다 작거나 같기 때문에
- 최대 10개 이하로 표현 가능
- $1+1+1+1+1+1+1+1+1+1$
- 10중 for문!

1, 2, 3 더하기

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

```
for (int l1=1; l1<=3; l1++) {  
    if (l1 == n) ans += 1;  
    for (int l2=1; l2<=3; l2++) {  
        if (l1+l2 == n) ans += 1;  
        ... 생략  
        for (int l0=1; l0<=3; l0++) {  
            if (l1+l2+l3+l4+l5+l6+l7+l8+l9+l0 == n) {  
                ans += 1;  
            }  
        }  
    }  
}
```

1, 2, 3 더하기

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

- 소스: <http://codeplus.codes/006055b814014702aa7abeded348e307>

순열 사용하기

순열

Permutation

37

- 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

38

- 크기가 N 인 순열은 총 $N!$ 개가 존재한다
- 순열을 사전순으로 나열했을 때
- $N = 3$ 인 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 `algorithm`에는 이미 `next_permutation`과 `prev_permutation`이 존재하기 때문에 사용하면 된다


다음 순열

Next Permutation

1. $A[i-1] < A[i]$ 를 만족하는 가장 큰 i 를 찾는다
2. $j \geq i$ 이면서 $A[j] > A[i-1]$ 를 만족하는 가장 큰 j 를 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 i 를 찾는다
- 즉, 순열의 마지막 수에서 끝나는 가장 긴 감소수열을 찾아야 한다
- 순열: 7 2 3 6 5 4 1

다음 순열

Next Permutation

42

- 순열: 7 2 3 6 5 4 1
- $j \geq i$ 이면서 $A[j] > A[i-1]$ 를 만족하는 가장 큰 j 를 찾는다
- 순열: 7 2 3 6 5 4 1

다음 순열

Next Permutation

43

- 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

44

- 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

C++ #include <algorithm>
next_permutation STL로 존재!

```
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;
}
```

반대로 하면
이전 순열

다음 순열

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

- 다음 순열을 구하는 문제

다음 순열

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

- 소스: <http://codeplus.codes/6c197e71ebf1455d956525dc6eafb95c>

이전 순열

48

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

- 이전 순열을 구하는 문제

이전 순열

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

- 소스: <http://codeplus.codes/09bed4156dbb45eeaac458b645e37e6d>

모든 순열

50

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

- 모든 순열을 구하는 문제

모든 순열

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

- 소스: <http://codeplus.codes/d77eca079b224dbaac5abda1445dfcd5>

팩토리얼

Factorial

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

차이를 최대로

<https://www.acmicpc.net/problem/10819>

- 수 N 개가 주어졌을 때 ($3 \leq N \leq 8$)
- $|A[0] - A[1]| + |A[1] - A[2]| + \dots + |A[N-2] - A[N-1]|$
- 를 최대로 하는 문제

차이를 최대로

<https://www.acmicpc.net/problem/10819>

- $N! = 8! = 40320$
- 모든 경우를 다해봐도 된다.
- 다음 순열을 이용해 모든 경우를 다 해본다

차이를 최대로

55

<https://www.acmicpc.net/problem/10819>

```
do {  
    int temp = calculate(a);  
    ans = max(ans, temp);  
} while(next_permutation(a.begin(), a.end()));
```

차이를 최대로

56

<https://www.acmicpc.net/problem/10819>

- 소스: <http://codeplus.codes/83d377dfd8b142dfb49eab0826c39b46>

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
- $W[i][j] = i \rightarrow j$ 비용

외판원 순회 2

58

<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

61

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

- $O(N \cdot N!)$
- 소스: <http://codeplus.codes/305bda020b3e4cd7810b244ed87aecaa>

외판원 순회 2

62

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

- 1 2 3 4
- 2 3 4 1
- 3 4 1 2
- 4 1 2 3
- 위의 4가지는 모두 같은 경우이다

외판원 순회 2

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

- 1 2 3 4
- 2 3 4 1
- 3 4 1 2
- 4 1 2 3
- 위의 4가지는 모두 같은 경우이다
- 즉, 시작점을 1로 고정해도 된다

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()+1, d.end()));
```


외판원 순회 2

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

```
do {  
    if (d[0] != 1) break;  
    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>

- $O(N!)$
- 소스: <http://codeplus.codes/7f92e70383994b888a29ba77ccbde922>

<https://www.acmicpc.net/problem/6603>

- 배열에 1, 1, 2, 2, 2를 넣고 next_permutation을 수행하면 어떻게 될까?

로또

<https://www.acmicpc.net/problem/6603>

- 1 1 2 2 2
- 1 2 1 2 2
- 1 2 2 1 2
- 1 2 2 2 1
- 2 1 1 2 2
- 2 1 2 1 2
- 2 1 2 2 1
- 2 2 1 1 2
- 2 2 1 2 1
- 2 2 2 1 1

로또

<https://www.acmicpc.net/problem/6603>

- 입력으로 주어진 K개의 수 중에서 6개의 수를 고르는 문제

<https://www.acmicpc.net/problem/6603>

- 0을 K-6개, 1을 6개를 넣은 다음에 next_permutation 를 수행하면 조합 모든 조합을 구할 수 있다

로또

<https://www.acmicpc.net/problem/6603>

- 소스: <http://codeplus.codes/600817f8dc28479e8697fd9b2fade90b>

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, \times 1개, \div 1개인 경우
- 60가지가 가능하다
- $1+2+3-4\times 5\div 6 = 1$
- $1\div 2+3+4-5\times 6 = 12$
- $1+2\div 3\times 4-5+6 = 5$
- $1\div 2\times 3-4+5+6 = 7$

연산자 끼워넣기

74

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

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

연산자 끼워넣기

75

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

- 소스: <http://codeplus.codes/52c1c673993e44618535a9fa2b415562>

재귀 함수 사용하기

재귀 함수 사용하기

Recursion

77

- 재귀 함수를 잘 설계해야 한다

1, 2, 3 더하기

78

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

- 정수 n 을 1, 2, 3의 합으로 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>

- $n \leq 10$ 이기 때문에
- 총 경우의 수는 3^n 인 것을 알 수 있다.

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 `count`개로 합 `sum`을 만드는 경우의 수
- 불가능한 경우
 - `sum > goal`
- 정답을 찾은 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 (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>

- 다 만들고 나서 보니 count는 별로 의미가 없다.

1, 2, 3 더하기

85

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

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

1, 2, 3 더하기

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

- 소스: <http://codeplus.codes/b746db30ac8541aeb75622b88911d54a>

암호 만들기

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

- 암호는 서로 다른 L 개의 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 C 가지
- 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
- 정답을 찾은 경우 (문제의 조건에 맞는지 확인 과정은 여기서 필요함)
 - `n == password.length()`
- 불가능한 경우
 - `i >= alpha.size()`

암호 만들기

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

- 다음 경우
 - 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>

- 소스: <http://codeplus.codes/97ae5cd477ec4157bfc54c969e6be5a>

로또

<https://www.acmicpc.net/problem/6603>

- 로또의 모든 조합을 출력해보는 문제

로또

<https://www.acmicpc.net/problem/6603>

- go(a, index, cnt)
 - a: 입력으로 주어진 수
 - index: 선택할지 말지 결정해야 하는 인덱스
 - cnt: 현재까지 포함한 수의 개수

<https://www.acmicpc.net/problem/6603>

- `go(a, index, cnt)`
 - `a`: 입력으로 주어진 수
 - `index`: 선택할지 말지 결정해야 하는 인덱스
 - `cnt`: 현재까지 포함한 수의 개수
- 정답을 찾은 경우
 - `cnt == 6`
- 불가능한 경우
 - `index == a.size()`
- 다음 경우 (선택하는 것은 다른 배열을 사용)
 - `index`번째를 선택: `go(a, index+1, cnt+1)`
 - `index`번째를 선택하지 않음: `go(a, index, cnt)`

로또

<https://www.acmicpc.net/problem/6603>

- 소스: <http://codeplus.codes/771c18460eaf43aab2df5a600c76482a>

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

100

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

- go(index, sum)
 - index: 부분집합에 포함할지 말지 결정해야 하는 인덱스
 - sum: 현재까지 부분집합의 합

부분집합의 합

101

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

- `go(index, sum)`
 - `index`: 부분집합에 포함할지 말지 결정해야 하는 인덱스
 - `sum`: 현재까지 부분집합의 합
- 정답을 찾은 경우
 - `index == n && sum == m`
- 불가능한 경우
 - `index == n && sum != m`
- 다음 경우
 - `index`번째를 부분집합에 추가: `go(index+1, sum+a[i])`
 - `index`번째를 부분집합에 추가하지 않음: `go(index+1, sum)`

부분집합의 합

102

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

- 소스: <http://codeplus.codes/38bc43d45b774d529673ee4d02022382>

퇴사

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

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

퇴사

104

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

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

퇴사

105

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

- `go(day, sum)`
 - `day`일이 되었다. `day`일에 있는 상담을 할지 말지 결정해야 한다.
 - 지금까지 얻은 수익은 `sum`이다
- 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

106

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

- 소스: <http://codeplus.codes/26bcee6c5b5245f09782110a6f21cb1f>

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

108

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

- 수열 = [1, 2, 3, 4, 5, 6], 연산자 = +2개, -1개, \times 1개, \div 1개인 경우
- 60가지가 가능하다
- $1+2+3-4\times 5\div 6 = 1$
- $1\div 2+3+4-5\times 6 = 12$
- $1+2\div 3\times 4-5+6 = 5$
- $1\div 2\times 3-4+5+6 = 7$

연산자 끼워넣기

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

- go(a, index, cur, plus, minus, mul, div)
 - a: 입력으로 주어진 수열
 - index: 현재 계산해야 하는 인덱스
 - cur: index-1번째까지 계산한 결과
 - plus, minus, mul, div: 사용할 수 있는 연산자의 개수

연산자 끼워넣기

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

- `go(a, index, cur, plus, minus, mul, div)`
 - `a`: 입력으로 주어진 수열
 - `index`: 현재 계산해야 하는 인덱스
 - `cur`: `index-1`번째까지 계산한 결과
 - `plus, minus, mul, div`: 사용할 수 있는 연산자의 개수
- 정답을 찾은 경우
 - `index == n`
- 다음 경우
 - `+` 사용: `go(a, index+1, cur+a[index], plus-1, minus, mul, div)`
 - `-` 사용: `go(a, index+1, cur-a[index], plus, minus-1, mul, div)`
 - `×` 사용: `go(a, index+1, cur*a[index], plus, minus, mul-1, div)`
 - `/` 사용: `go(a, index+1, cur/a[index], plus, minus, mul, div-1)`

연산자 끼워넣기

111

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

- 소스: <http://codeplus.codes/4144c096fb4e47fb8ae1999bfbbbc8a2>

연산자 끼워넣기 (2)

<https://www.acmicpc.net/problem/15658>

- N개의 수로 이루어진 수열과 **N-1개 이상**의 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 (2)

<https://www.acmicpc.net/problem/15658>

- 수열 = [1, 2, 3, 4, 5, 6], 연산자 = +3개, -2개, \times 1개, \div 1개인 경우
- 72가지가 가능하다
- $1+2+3-4\times 5\div 6 = 1$
- $1\div 2+3+4-5\times 6 = 12$
- $1+2\div 3\times 4-5+6 = 5$
- $1\div 2\times 3-4+5+6 = 7$
- $1+2+3+4-5-6 = -1$
- $1+2+3-4-5\times 6 = -18$

연산자 끼워넣기 (2)

<https://www.acmicpc.net/problem/15658>

- go(a, index, cur, plus, minus, mul, div)
 - a: 입력으로 주어진 수열
 - index: 현재 계산해야 하는 인덱스
 - cur: index-1번째까지 계산한 결과
 - plus, minus, mul, div: 사용할 수 있는 연산자의 개수

연산자 끼워넣기 (2)

<https://www.acmicpc.net/problem/15658>

- `go(a, index, cur, plus, minus, mul, div)`
 - `a`: 입력으로 주어진 수열
 - `index`: 현재 계산해야 하는 인덱스
 - `cur`: `index-1`번째까지 계산한 결과
 - `plus, minus, mul, div`: 사용할 수 있는 연산자의 개수
- 정답을 찾은 경우
 - `index == n`
- 다음 경우
 - `+` 사용: `go(a, index+1, cur+a[index], plus-1, minus, mul, div)`
 - `-` 사용: `go(a, index+1, cur-a[index], plus, minus-1, mul, div)`
 - `×` 사용: `go(a, index+1, cur*a[index], plus, minus, mul-1, div)`
 - `/` 사용: `go(a, index+1, cur/a[index], plus, minus, mul, div-1)`

연산자 끼워넣기 (2)

116

<https://www.acmicpc.net/problem/15658>

- 소스: <http://codeplus.codes/fbec8490acdb46dd879e7dc6c1972b6e>
- 연산자 끼워넣기와 같은 소스로 해결할 수 있다

비트마스크 사용하기

비트마스크

Bitmask

118

- 비트(bit) 연산을 사용해서 부분 집합을 표현할 수 있다.

비트 연산

Bitwise operation

119

- $\&$ (and), $|$ (or), \sim (not), \wedge (xor)

A	B	$\sim A$	$A \& B$	$A B$	$A \wedge B$
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.
- $A = 27, B = 83$ 인 경우
- $A = 11011_2, B = 1010011_2$
- $A \& B = 19, A \mid B = 91, A \wedge B = 72$

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.
- $A = 83 = 1010011_2$
- $\sim A = 10101100_2$ (8비트 자료형인 경우)
- $\sim A = 11111111\ 11111111\ 11111111\ 10101100_2$ (32비트 자료형인 경우)
- 또, unsigned, signed에 따라서 보여지는 값은 다르다.

비트 연산

Bitwise operation

- shift left (<<) 와 shift right (>>) 연산이 있다.
- $A \ll B$ (A를 왼쪽으로 B비트만큼 민다.)
- $1 \ll 0 = 1$
- $1 \ll 1 = 2 \ (10_2)$
- $1 \ll 2 = 4 \ (100_2)$
- $1 \ll 3 = 8 \ (1000_2)$
- $1 \ll 4 = 16 \ (10000_2)$
- $3 \ll 3 = 24 \ (11000_2)$
- $5 \ll 10 = 5120 \ (101000000000000_2)$

비트 연산

123

Bitwise operation

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

비트 연산

Bitwise operation

124

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

비트마스크

Bitmask

125

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

비트마스크

Bitmask

126

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

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

비트마스크

Bitmask

127

- 보통 0부터 $N-1$ 까지 정수로 이루어진 집합을 나타낼 때 사용한다.
- 1부터 N 까지 정수로 이루어진 집합을 사용하는 건 공간이 2배 더 필요하다.
- 또, 각종 연산을 조금 변형해서 사용해야 한다.
- 따라서, 0부터 $N-1$ 까지로 변형해서 사용하는 것이 더 좋다.

비트마스크

Bitmask

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

```

      1000111010
    & 0000000100
    -----
      0000000000

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


비트마스크

Bitmask

• {1, 3, 4, 5, 9} = 570

• 1 추가하기

• $570 \mid 2^1 = 570 \mid (1 \ll 1) = 570 \text{ (1000111010}_2\text{)}$

$$\begin{array}{r} 1000111010 \\ | 0000000100 \\ \hline \end{array}$$

• 2 추가하기

• $570 \mid 2^2 = 570 \mid (1 \ll 2) = 574 \text{ (1000111110}_2\text{)}$

$$\begin{array}{r} 1000111010 \\ | 0000001000 \\ \hline \end{array}$$

• 3 추가하기

• $574 \mid 2^3 = 570 \mid (1 \ll 3) = 570 \text{ (1000111010}_2\text{)}$

$$\begin{array}{r} 1000111110 \\ | 0000001000 \\ \hline \end{array}$$

• 4 추가하기

• $574 \mid 2^4 = 570 \mid (1 \ll 4) = 570 \text{ (1000111010}_2\text{)}$

$$\begin{array}{r} 1000111010 \\ | 0000001000 \\ \hline \end{array}$$

비트마스크

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$

- 1 제거하기

- $570 \ \& \sim 2^1 = 570 \ \& \sim (1 \ll 1) = 568 \ (1000111000_2)$

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

- 2 제거하기

- $570 \ \& \sim 2^2 = 570 \ \& \sim (1 \ll 2) = 570 \ (1000111010_2)$

```
1000110010
```

- 3 제거하기

- $562 \ \& \sim 2^3 = 562 \ \& \sim (1 \ll 3) = 562 \ (1000110010_2)$

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

- 4 제거하기

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

```
1000111010
```

비트마스크

Bitmask

- $\{1, 3, 4, 5, 9\} = 570$

- 1 토글하기

- $570 \wedge 2^1 = 570 \wedge (1 \ll 1) = 568 \text{ (1000111000}_2\text{)}$

$$\begin{array}{r} 1000111010 \\ \wedge 0000000100 \\ \hline \end{array}$$

- 2 토글하기

- $570 \wedge 2^2 = 570 \wedge (1 \ll 2) = 574 \text{ (1000111110}_2\text{)}$

$$\begin{array}{r} 1000111010 \\ \wedge 0000001000 \\ \hline \end{array}$$

- 3 토글하기

- $574 \wedge 2^3 = 570 \wedge (1 \ll 3) = 562 \text{ (1000110010}_2\text{)}$

$$\begin{array}{r} 1000111110 \\ \wedge 0000010000 \\ \hline \end{array}$$

- 4 추가하기

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

$$\begin{array}{r} 1000110010 \\ \wedge 0000010000 \\ \hline \end{array}$$

비트마스크

Bitmask

132

- 전체 집합
 - $(1 \ll N) - 1$
- 공집합
 - 0

비트마스크

133

Bitmask

- 현재 집합이 S일때
- i를 추가
 - $S \mid (1 \ll i)$
- i를 검사
 - $S \& (1 \ll i)$
- i를 제거
 - $S \& \sim(1 \ll i)$
- i를 토글 (0을 1로, 1을 0으로)
 - $S \wedge (1 \ll i)$

비트 연산

Bitwise operation

- 비트 연산을 사용할 때는 연산자 우선 순위를 생각해야 한다.
- $1 \ll N - 1$ 은 $(1 \ll N) - 1$ 일까? $1 \ll (N - 1)$ 일까?

비트 연산

135

Bitwise operation

- 비트 연산을 사용할 때는 연산자 우선 순위를 생각해야 한다.
- $1 \ll N - 1$ 은 $(1 \ll N) - 1$ 일까? $1 \ll (N - 1)$ 일까?
- 정답은 $1 \ll (N - 1)$

집합

136

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

- 비트마스크를 연습해보는 문제

집합

137

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

- 소스: <http://codeplus.codes/5a41773c77ab4cb9920ba778e0a3acee>

비트마스크

Bitmask

138

- 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

- C++ 기준으로 int는 32비트, long long는 64비트이다.
- 64비트를 넘는 비트는 정수로 나타낼 수 없다.
- 이런 경우에는 C++은 `bitset`을 이용하면 된다.

부분집합의 합

140

<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

141

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

- 모든 집합의 개수 = 2^N
- 모든 집합을 구해보면 된다!

부분집합의 합

142

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

- 전체 집합 = $(1 \ll N) - 1$

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

부분집합의 합

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

- 전체 집합 = $(1 \ll N) - 1$
- 공집합은 제외해야 한다

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

부분집합의 합

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

- 전체 집합 = $(1 \ll N) - 1$
- 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

145

<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

146

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

- 소스: <http://codeplus.codes/6d279b66225c4a7b8e54478d4f03293b>

끝

코드 플러스

148

<https://code.plus>

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.