브루트 포스 - 순열

최백준 choi@startlink.io

순열 사용하기

순열

Permutation

- 임의의 수열을 다른 순서로 섞는 연산
- A = [1, 5, 6]인 경우 [1, 5, 6], [1, 6, 5], [5, 1, 6], [5, 6, 1], [6, 1, 5], [6, 1, 6]이 순열이다.

순열

Permutation

- 크기가 N인 수열의 서로 다른 순열은 총 N!개가 있다.
- 모든 순열을 사전순으로 나열했을 때
- A = [1, 2, 3]인 경우 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

- 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

Next Permutation

- 다음 순열을 구하는 시간 복잡도는 O(N)이다.
- 전체 순열을 모두 구하는 시간 복잡도는 O(N!×N) 이다.

다음 순열

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

- 다음 순열을 구하는 문제

다음 순열

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

- 소스: http://codeplus.codes/cbbd695b7ba34e14b36ea43f11047a88

이전 순열

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

- 이전 순열을 구하는 문제

이전 순열

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

- 소스: http://codeplus.codes/40905ecd5b94444e968fa4eb2428cc0c

모든 순열

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

- 모든 순열을 구하는 문제

모든 순열

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

- 소스: http://codeplus.codes/343de5d49ae74b6daac615f4424f92a5

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 N개가 주어졌을 때 ($3 \leq N \leq 8$)
- $|A[0] - A[1]| + |A[1] - A[2]| + \ldots + |A[N-2] - A[N-1]|$
- 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

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

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

차이를 최대로

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

- 소스: http://codeplus.codes/d4962757773d4c9294bfabf69229aed5

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, 0인 경우는 갈 수 없음

외판원 순회 2

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

- $2 \leq N \leq 10$
- N! = 10! = 3628800
- 모든 경우를 다해봐도 된다
- 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!)
- 소스: http://codeplus.codes/fcc33a5f5df94b60a26893be88b3aa2e

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.
- 다시 시작한 도시로 돌아가야 하기 때문
- 따라서 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/3fd7e661e2be4186a75a7b27450b35a2

로또

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

- 같은 수가 있어도 순열을 만들 수 있다.
- 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

로또

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/ad4f81f0566c443895c02be939d0b09d

끝

코드 플러스

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.