SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

알고리즘

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Day6. 백트래킹

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

순열과 조합

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 경우의 수 출력

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 111
  - 112
  - 113
  - 121
  - …
  - 444

```
4 3
1 2 3 4
111
112
113
114
121
122
123
124
131
```

```
423
424
431
432
433
434
441
442
443
444
```

지금까지 우리가 학습한 내용의 모든 것

```
void func(int now) {
    if (now == k) {
        for (int i = 0; i < k; i++) {
            cout << DAT[i];
        }
        cout << '\n';
        return;
    }

    for (int i = 0; i < n; i++) {
        DAT[now] = arr[i];
        func(now + 1);
        DAT[now] = 0;
    }
}
```

https://gist.github.com/hoconoco/09fba3ab76676093d6d2b62484b79149

1, 2, 3, 4 를 나열하는 방법은 어떤 게 있을까?

- **순서를 고려할 지?**
- **중복을 고려할 지?**

[참고] 순열과 조합

순열 : 중복을 허용하지 않고 순서대로 숫자를 나열한다.

- 순서가 다르면 다른 순열이다. ex) 1 2 3 / 3 2 1 (서로 다른 순열 O)
- 중복된 숫자를 뽑지 않는다. ex) 1 1 2 (허용 x)

조합 : 중복을 허용하지 않고 선택할 수 있는 숫자의 조합

- 순서를 고려하지 않고 숫자를 선택한 경우이다. ex) 1 2 3 / 3 2 1 (같은 조합 O, 한 가지 경우만 존재)
- 중복된 숫자를 뽑지 않는다. ex) 1 1 2 (허용 x)

중복 순열 : 중복을 허용하며 순서대로 숫자를 나열한다.

- 순서가 다르면 다른 순열이다. ex) 1 2 3 / 3 2 1 (서로 다른 순열 O)
- 중복된 숫자를 허용한다. ex) 1 1 2 (허용 O)

중복 조합 : 중복을 허용하며 선택할 수 있는 숫자의 조합

- 순서를 고려하지 않고 숫자를 선택한 경우이다. ex) 1 2 3 / 3 2 1 (같은 조합 O, 한 가지 경우만 존재)
- 중복된 숫자를 허용한다. ex) 1 1 2 (허용 O)

그동안 재귀에서 다룬 숫자 뽑기

우리가 그동안 했던 경우의 수들은 중복 순열이었다.

- **중복 순열 : 중복을 허용하며 순서대로 숫자를 나열한다.**

1 2 3 4 중 3개의 숫자를 뽑는 모든 경우의 수 = 중복 순열

- **중복 O (1 1 1)**
- **순열 (순서대로 나열, 1 1 2 , 1 2 1 모두 뽑음)**

- **총 64개이다.**
- **DAT 대신 path 사용**
- **visited[] 로 경로 체크**

```
for (int i = 0; i < n; i++) {
    path[level] = number[i];
    func(level + 1);
    path[level] = 0;
}
```

https://gist.github.com/hoconoco/a4ba86a5390beaa90aadeae2d3cac4e6

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 순열 출력

개수도 출력한다

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 1 2 3
  - …
  - 4 3 2
  - 24 개

```
4 3
1 2 3 4
1 2 3
1 2 4
1 3 2
1 3 4
1 4 2
1 4 3
2 1 3
2 1 4
2 3 1
2 3 4
2 4 1
2 4 3
3 1 2
3 1 4
3 2 1
3 2 4
3 4 1
3 4 2
4 1 2
4 1 3
4 2 1
4 2 3
4 3 1
4 3 2
24
```

QUIZ 2 정답

순열은 중복 허용 X

순서 다르면 허용 O

- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
for (int i = 0; i < n; i++) {
    if (visited[i] == 0) {
        visited[i] = 1;
        path[level] = number[i];
        func(level + 1);
        path[level] = 0;
        visited[i] = 0;
    }
}
```

https://gist.github.com/hoconoco/0e71602231b114f0cc34b3af77ce8bf4

N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 중복 조합 출력

- **입력 예시**
  - 4 3
  - 1 2 3 4
- **출력 예시**
  - 1 1 1
  - 1 1 2
  - 1 1 3
  - 1 1 4
  - 1 2 2
  - 1 2 3
  - 1 2 4
  - 1 3 3
  - 1 3 4
  - 1 4 4
  - 2 2 2
  - 2 2 3
  - 2 2 4
  - 2 3 3
  - 2 3 4
  - 2 4 4
  - 3 3 3
  - 3 3 4
  - 3 4 4
  - 4 4 4
  - 20 개

```
4 3
1 2 3 4
1 1 1
1 1 2
1 1 3
1 1 4
1 2 2
1 2 3
1 2 4
1 3 3
1 3 4
1 4 4
2 2 2
2 2 3
2 2 4
2 3 3
2 3 4
2 4 4
3 3 3
3 3 4
3 4 4
4 4 4
20
```

QUIZ 3 정답

중복 조합은 순서 다르면 허용 X

- 이전 값보다 작은 선택을 할 수 없게 한다.
- 코드를 이해하고 직접 짤 수 있어야 한다.

```
for (int i = 0; i < n; i++) {
    //이전 값보다 작은 선택은 하지 않는다.
    if (level > 0 && path[level - 1] > number[i]) continue;
    path[level] = number[i];
    func(level + 1);
    path[level] = 0;
}
```

https://gist.github.com/hoconoco/2e849ec47832e208ffdd52b9e77abea4

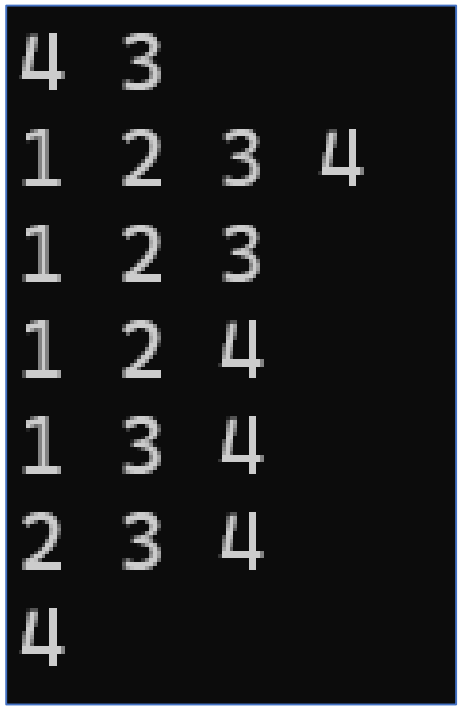
N,K를 입력 받고

N개의 정수를 입력 받아서 K개를 뽑는 조합 출력

개수도 출력한다

- **입력 예시**
  - 4 3
  - 1 2 3 4

- **출력 예시**
  - 1 2 3
  - 1 2 4
  - 1 3 4
  - 2 3 4
  - 4개



조합은 중복 허용 X

순서 다르면 허용 X

- **코드를 이해하고 직접 짤 수 있어야 한다.**

```
for (int i = 0; i < n; i++) {
    //중복 허용 X
    if (visited[i] == 0) {
        //이전 값보다 작은 선택은 하지 않는다.
        if (level > 0 && path[level - 1] > number[i]) continue;
        visited[i] = 1;
        path[level] = number[i];
        func(level + 1);
        path[level] = 0;
        visited[i] = 0;
    }
}
```

https://gist.github.com/hoconoco/08ee36d86f40d1b36c5a249de70e4a03

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

재귀 중 연산

재귀를 하면서 연산을 할 수 있다.

다음과 같이 처리할 수 있다.

1. 기저에서 모아서 하는 경우
2. 전역 변수를 만들어서 처리
3. 매개변수로 넘겨서 처리

하나씩 진행해 본다.

누적합을 구하자

- **입력 예시**
  - 4 3
  - 1 2 3 4
- **출력 예시**
  - 111 3
  - 112 4
  - 113
  - 121
  - …
  - 444 12

```
4 3
1 2 3 4
111 sum : 3
112 sum : 4
113 sum : 5
114 sum : 6
121 sum : 4
122 sum : 5
123 sum : 6
124 sum : 7
131 sum : 5
132 sum : 6
```

```
442 sum : 10
443 sum : 11
444 sum : 12
```

QUIZ 6 정답

매우 쉬웠다 하하

```
void func(int now) {
    if (now == k) {
        int sum = 0;
        for (int i = 0; i < k; i++) {
            sum += DAT[i];
            cout << DAT[i];
        }
        cout << " sum : " << sum << '\n';
        return;
    }

    for (int i = 0; i < n; i++) {
        DAT[now] = arr[i];
        func(now + 1);
        DAT[now] = 0;
    }
}
```

https://gist.github.com/hoconoco/a058a9a47e5a0006b009664ff6da3a72

현재 방식

그동안에 했던 선택했던 값들을 모두 모아서 합계를 구했다

- 경로를 다 마치고 나서 다시 연산 작업이 들어가니 불편하다

만약, 경로를 지나갈 때마다 (선택을 하면서 / 재귀를 호출할 때마다)
누적 합을 구한다면 속도가 빨라지지 않을까?

해결 방법

1. 매개변수로 기록
2. 전역 변수로 기록

방법1. 매개변수로 기록한다

- 매개변수로 처리하는 것은
  전역변수보다 좀 더 고민이 많이 필요하다

```
int sum = 0;
void func(int now, int sum) {
    if (now >= k) {
        for (int i = 0; i < k; i++) {
            cout << DAT[i];
        }
        cout << " sum : " << sum << '\n';
        return;
    }

    for (int i = 0; i < n; i++) {
        DAT[now] = arr[i];
        func(now + 1, sum+arr[i]);
        DAT[now] = 0;
    }
}
```

https://gist.github.com/hoconoco/71bd0317ee5404802fe59ba6cbe76b22

방법2. 전역 변수 사용하기

- 복구하는 코드 필수

```
int sum = 0;
void func(int now) {
    if (now >= k) {
        for (int i = 0; i < k; i++) {
            cout << DAT[i];
        }
        cout << " sum : " << sum << '\n';
        return;
    }

    for (int i = 0; i < n; i++) {
        DAT[now] = arr[i];
        sum += arr[i];
        func(now + 1);
        DAT[now] = 0;
        sum -= arr[i];
    }
}
```

https://gist.github.com/hoconoco/7763ab9a461620e9ac91e8c9b84919dd

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

백트래킹

Backtracking 이란

재귀를 진행하면서 애초에 가망성이 없는 경로를 선택하지 않도록 특정 조건을 처리하는 작업

- **연산 속도 UP**
- **최적화!**

이런 유형이 있다.

1~4 까지 숫자 중 3개를 뽑는다.

다만, 인접하는 수로 1차이가 나면 뽑지 않는다.

- 123(x)
- 213(x)
- 135(O)

1을 선택한 다음 2 를 선택하는 것은 매우 불필요한 작업이다.

이럴 경우 조건을 넣어서 2를 선택하지 않도록 해야 한다.

백트래킹은 특별하게 어떤 알고리즘이 아니다.

- **그저 재귀에서 불필요한 선택을 하지 않기 위한 방법 일뿐**

```
for (int i = 1; i <= 4; i++) {
    //판별 : 인접하는 수가 1차이면 뽑지 않는다.

    //하나는 뽑아야 비교가 된다.
    if (now > 0) {
        //이전 값과 지금 뽑을 값의 차이가 1이면 넘겨! ( 절대값 )
        if ( abs(path[now - 1] - i) == 1 ) continue;
    }

    //앞으로~
    path[now] = i;
    func(now + 1);
    path[now] = 0;
}
```

https://gist.github.com/hoconoco/b790b623f6966da88a6ad7ff5f789d79

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미