

특강 3일차 - 완전 탐색

Contents


01	—————	완전 탐색
02	—————	순열
03	—————	조합
04	—————	부분 집합
05	—————	완전 탐색 실습

완전 탐색

완전 탐색

- Exhaustive search
- Brute force
- 문제를 해결하기 위해 가능한 모든 경우의 수를 시도하는 알고리즘 기법
- 모든 경우의 수를 탐색하기 때문에 답을 놓칠 염려가 없지만, 경우의 수가 많아지면 비효율적일 수 있음
- 간단하고 직관적이기 때문에 상대적으로 빠른 시간 안에 알고리즘 설계 가능
- 일반적으로 경우의 수가 작을 때 유용
- 코딩테스트에서 문제를 풀 때, 우선 완전 검색으로 접근하여 해답을 도출한 후, 성능 개선을 위해 다른 알고리즘을 사용하고
해답을 확인하는 것이 바람직함

exhaustive

미국·영국 [ɪgˈzɔːstrɪv]  영국식 

형용사

(하나도 빠뜨리는 것 없이) 철저한[완전한]

완전 탐색

- 많은 종류의 문제들이 특정 조건을 만족하는 경우나 요소를 찾는 것이다.
- 또한, 이들은 전형적으로 순열(permutation), 조합(combination), 그리고 부분집합(subset)과 같은 조합적 문제들 (Combinatorial Problems) 과 연관된다.
- 외판원 순회(TSP)
- 배낭 채우기
- 연구소
 - <https://www.acmicpc.net/problem/14502>
- 게리맨더링
 - <https://www.acmicpc.net/problem/17471>

순열

순열

- 서로 다른 것들 중 몇 개를 뽑아서 순서적으로 나열하는 것
- $nPr = n * (n-1) * (n-2) * \dots * (n-r+1)$
- $nPn = n * (n-1) * (n-2) * \dots * 2 * 1$
- 다수의 알고리즘 문제들은 순서화된 요소들의 집합에서 최선의 방법을 찾는 것과 관련

N	순열의 수
10	3628800
11	39916800
12	479001600
13	6227020800
14	87178291200

순열 생성-1

- {1,2,3} 중 3개를 뽑아 순서적으로 나열

```
for(int i=1; i<=3; ++i) {  
    for(int j=1; j<=3; ++j) {  
        if(j==i) continue;  
        for(int k=1; k<=3; ++k) {  
            if(k==i || k==j) continue;  
            System.out.println(i+" "+j+" "+k);  
        }  
    }  
}
```


순열 생성-2

- {1,2,3} 중 3개를 뽑아 순서적으로 나열

```
public static void permutation(int cnt) {  
    if(cnt==3) {  
        totalCount++;  
        System.out.println(Arrays.toString(numbers));  
        return;  
    }  
    for(int i=1; i<=3; ++i) {  
        if(isSelected[i]) continue;  
        numbers[cnt] = i;  
        isSelected[i] = true;  
        permutation(cnt+1);  
        isSelected[i] = false;  
    }  
}
```

순열 응용

- 주사위를 3번 던져서 나올 수 있는 모든 경우 출력

```
1 1 1
1 1 2
...
1 2 1
...
6 6 5
6 6 6
```

- 주사위를 3번 던져서 서로 다른 수를 순서적으로 나열 가능한

모든 경우 출력

```
1 2 3
1 2 4
...
1 3 2
...
3 2 1
...
6 5 3
6 5 4
```

조합

조합

- 서로 다른 n 개의 원소 중 r 개를 순서 없이 골라낸 것
- 예를 들어, $\{A, B, C\}$ 에서 두 개를 선택하는 조합은 $\{A, B\}$, $\{A, C\}$, $\{B, C\}$ 로 총 3가지
- 순열과 달리, $\{A, B\}$ 와 $\{B, A\}$ 는 동일한 조합으로 취급
- $nCr = nCn-r$

$${}_nC_r = \frac{{}_nP_r}{r!} = \frac{n!}{(n-r)!r!}$$

조합 생성-1

- {1,2,3} 중 2개를 순서 무관하게 선택

```
for(int i=1; i<=3; ++i) {  
    for(int j=i+1; j<=3; ++j) {  
        System.out.println(i+" "+j);  
    }  
}
```

조합 생성-2

- {1,2,3} 중 2개를 순서 무관하게 선택

```
private static void combination(int cnt,int start) {  
    if(cnt == 2) {  
        totalCount++;  
        System.out.println(Arrays.toString(numbers));  
        return;  
    }  
    for(int i=start; i<=3; ++i) {  
        numbers[cnt] = i;  
        combination(cnt+1,i+1);  
    }  
}
```

조합 응용

- 주사위를 3번 던져서 중복이 되는 경우를 제외하고

나올 수 있는 모든 경우

- 1 1 2 와 중복 : 1 2 1, 2 1 1

1 1 1
1 1 2
...
1 1 6
1 2 2
...
5 6 6
6 6 6

- 주사위를 3번 던져서 모두 다른 수가 나올 수 있는 모든 경우

- 1 2 3 과 중복 : 1 3 2, 2 1 3, 2 3 1, 3 1 2

1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
...
4 5 6

부분 집합

부분 집합

- 원래 집합의 모든 가능한 하위 집합
- 공집합과 자신도 부분 집합에 포함
- 예시: A, B의 부분 집합= $\{\}, \{A\}, \{B\}, \{A, B\}$
- 부분 집합은 원소를 선택하거나 선택하지 않는 방법으로 만들어지며,
집합의 크기가 n 일 때 공집합을 포함한 부분 집합의 개수는 2^n
 - 각 원소가 부분 집합에 포함, 포함되지 않는 두 가지 경우의 수

1

2

3

부분 집합 생성

- {1,2,3} 집합으로 만들 수 있는 모든 부분집합 생성

```
private static void makePowerset(int no) {  
    if(no>3) {  
        ++totalCount;  
        for(int i=1; i<=3; ++i) {  
            System.out.print((isSelected[i]?i:"X")+" ");  
        }  
        System.out.println();  
        return;  
    }  
    isSelected[no] = true;  
    makePowerset(no+1);  
    isSelected[no] = false;  
    makePowerset(no+1);  
}
```

완전 탐색 실습

완전 탐색 실습

- 백설공주와 일곱난쟁이
 - <https://www.acmicpc.net/problem/3040>
- 캠프 준비
 - <https://www.acmicpc.net/problem/16938>
- 영재의 시험
 - <https://www.acmicpc.net/problem/19949>

Thank you!
