완전탐색(Brute Force) & 시뮬레이션

▼ 1. 완전탐색(Brute Force)

가능한 모든 경우의 수를 탐색하면서 답을 찾는 알고리즘 방법

▼ 🔎 1.1. Brute Force의 장단점

✔️ 장점

- 모든 경우를 다 고려하기 때문에 확실한 정답을 찾을 수 있다.
- 복잡한 알고리즘 없이 빠르게 구현 가능

✔️ 단점

- 모든 경우를 다 고려하기 때문에 효율적이지 못한 경우가 많다.
- 알고리즘 실행시간이 오래 걸린다.

> 💡 **Keypoint!**
> 완전탐색은 **데이터의 크기가 작은 경우**(약 100만 이하)에 사용하는 것이 적절하다.

▼ 🔎 1.2. Brute Force 예시

> **"10개의 정수 원소로 이루어진 수열이 있다.**
> **이 수열에서 두 원소를 선택해서 구한 합의 최댓값을 구하시오."**

10개의 원소에서 두 원소를 고르는 **모든 경우는 45가지** 밖에 되지 않으므로 Brute Force로도 충분히 빠르게 해결할 수 있다.

하지만, 원소의 개수가 **10만이라면 경우의 수가 50억**이 되므로 시간초과가 될 가능성이 높다.

따라서 **완전탐색로도 충분히 풀 수 있는 문제인지 파악하는 것이 가장 중요**하다.

> 💡 **Keypoint!**
> 완전 탐색은 기본적으로 데이터의 크기가 작다는 것을 전제로 하기 때문에 **시간복잡도가 지수승이나 팩토리얼 꼴**로 나온다.

▼ 🔎 1.3. Brute Force에 사용되는 기법

완전 탐색은 특정 알고리즘에 국한되지 않고 다양하게 구현될 수 있는데, 가장 많이 사용되는 기법들은 아래와 같다.

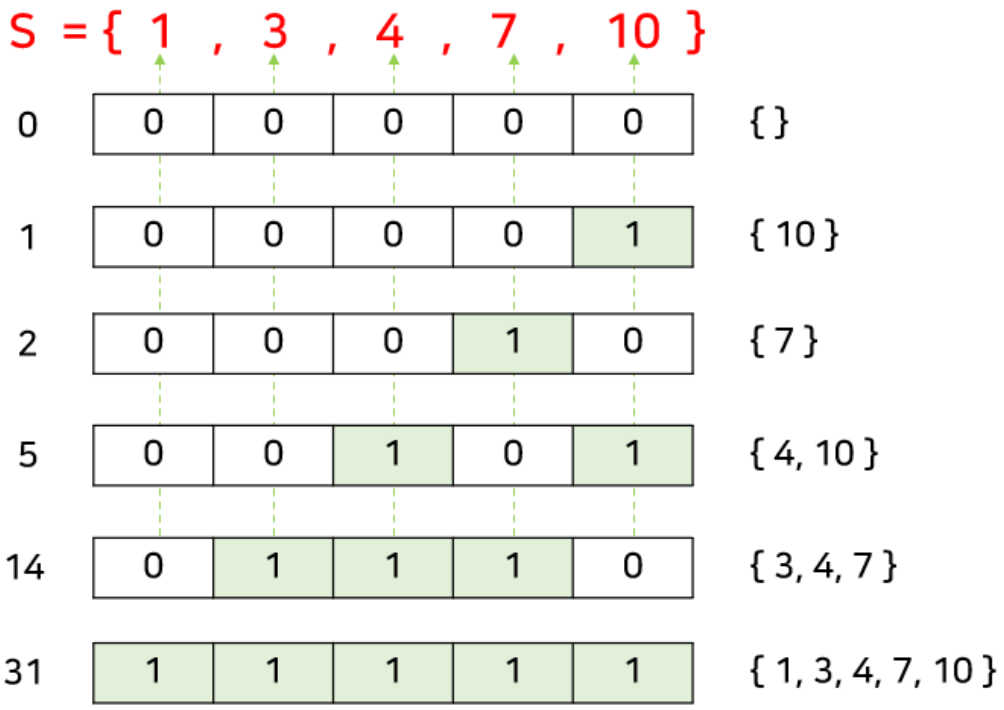
1. 단순 Brute-Force

어떤 특정 기법을 사용하지 않고 단순히 for문 또는 if문 등으로 모든 case을 고려하여 답을 구하는 방법

2. 비트마스크(Bitmask)

나올 수 있는 모든 경우의 수 중에 각각의 원소가 **포함되는지, 포함되지 않는지 두 가지만의 선택으로 구성되는 경우**에 유용하게 사용한다.

예를 들어서, 원소가 5개인 집합의 모든 부분집합을 구해야 하는 경우, 부분집합의 원소에 포함되는지 안 되는지의 경우만 존재하기 때문에 아래와 같이 표현할 수 있다.



이처럼 포함여부를 이진수로 표현하여 체크하는 기법이다.

3. 백트래킹

현재 상태에서 가능한 후보군으로 **가지를 치며** 탐색하는 방법

4. 순열 (Permutation)

완전 탐색의 가장 대표적인 유형

순열을 이용한 완전탐색은 기본적으로 시간복잡도가 N제곱이므로 완전탐색에 적절한 문제인지 잘 확인할 것

5. BFS / DFS

너비 우선 탐색(**BFS**) : 정점과 같은 레벨에 있는 형제 노드들을 탐색

깊이 우선 탐색(**DFS**) : 정점의 자식 노드들을 탐색

▼ 🔎 1.4. Brute Force 잘 활용하기

1. 입력으로 주어지는 데이터(N)의 크기가 매우 작다.

보통 프로그래밍 문제들을 풀게 되면 N = 10만, 20만 같은 크기를 주는 경우가 많다. 하지만 대부분의 경우 **완전 탐색 문제는 N의 크기가 매우 작다.**

2. 답의 범위가 작고, 임의의 답을 하나 선택했을 때 문제 조건을 만족하는지 역추적 할 수 있다.

답의 범위가 아주 제한적인 경우에는, 임의로 답을 고정시켜놓고 주어진 조건들이 답에 적합한지 역으로 확인해보는 방법을 이용해볼 수 있다. 가능한 답을 모두 확인하는 과정에서 완전 탐색이 이용되는 것이다.

즉, N의 크기가 작은 게 아니라 답의 범위가 작을 경우, 답을 유추하는 게 아니라 답을 정해놓고 역으로 조건에 적합한지 체크하는 완전탐색을 이용하는 방법이다.

3. 여러 문제 조건 중 한 조건을 고정시키면 문제 풀이가 간단해진다.

어떤 문제에서 주어진 조건 중 **하나를 고정시켜서 문제가 단순해질 수 있는 경우**에, 완전 탐색을 적용하여 문제를 해결할 수도 있다. (완전 탐색 + 그리디)

알아두는 것만으로도 접근법에 도움이 될 것 같다

▼ 2. 시뮬레이션

문제에서 제시한 알고리즘을 **한 단계씩 차례대로 직접 수행**해야 하는 문제

특정 알고리즘에 종속되는 것이 아니라 **주어진 문제에 맞게 구현**하면 되는데, 구현이 빡세게 필요한 것들을 통틀어서 시뮬레이션 유형이라고 한다.

그냥 빡구현의 유형이다. 노가다의 유형 부르기 뭣해서 시뮬레이션이라는 명칭을 붙인 것 같다. 아무튼 **하드코딩이 상당히 요구되는 유형**.... 눈물

🔎 2.1. 종류

- 알고리즘은 간단한데 **코드가 지나칠 만큼 길어지**는 문제
- **실수 연산**을 다루고 **특정 소수점 자리**까지 출력해야 하는 문제

- 문자열을 **특정한 기준에 따라서 끊어 처리**해야 하는 문제
- **적절한 라이브러리**를 찾아서 사용해야 하는 문제

= 정말 귀찮은 문제