브루트포스

물론입니다! 아래는 **브루트포스**(Brute Force) 알고리즘에 대한 간략한 요약과, **C, Python, C++**에서 기본적인 예제를 제시한 내용입니다.

🔍 브루트포스 알고리즘 (Brute Force) 요약

📌 정의

브루트포스는 가능한 모든 경우를 시도하여 문제를 해결하는 **직접적인 접근 방식**입니다.
시간 복잡도가 일반적으로 높을 수 있지만, 문제의 규모가 작거나 해답의 범위가 제한적이면 유용합니다.

✅ 장점

- 구현이 간단하고 직관적
- 모든 경우를 탐색하여 **정확한 해를 보장**할 수 있음
- 복잡한 문제에 대한 **기초적인 해결 방안**

❌ 단점

- 시간/공간 복잡도가 매우 높음 (예: O(n!) → n이 커질수록 비효율적)
- 실무에서 실용성이 제한적 (대신 더 효율적인 알고리즘이 존재할 수 있음)

📚 대표적인 문제 예시

문제	설명
암호 해독 (Password cracking)	가능한 모든 문자 조합을 시도하여 비밀번호 찾기
조합 문제 (Subset Sum)	배열에서 합이 특정 값인 부분 집합 찾기
두 수의 곱이 특정 값인 경우 찾기	두 수의 곱이 100인 경우 (1~100 범위)
행복한 수 (Happy Number)	반복해서 제곱합 계산하여 1이 되는지 확인 (단, 이 경우는 브루트포스가 아님)

💡 예제: 두 수의 곱이 100인 경우 찾기 (브루트포스)

✅ 문제

1~100 범위에서 두 수의 곱이 100인 경우를 모두 찾아보세요.

🐍 Python 예제

```python
def find_pairs_product(target=100, max_val=100):
    pairs = []
    for i in range(1, max_val + 1):
        for j in range(1, max_val + 1):
            if i * j == target:
                pairs.append((i, j))
    return pairs

# 실행
result = find_pairs_product(100, 100)
print("두 수의 곱이 100인 경우:", result)
```

출력:

```
두 수의 곱이 100인 경우: [(1, 100), (2, 50), (4, 25), (5, 20), (10, 10), (20, 5), (25, 4), (50, 2), (100, 1)]
```

> ⚠️ 주의: 이 경우는 중복을 제거하지 않음. 원하는 경우에 따라 정렬/중복 제거 가능

🚀 C++ 예제

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

vector<pair<int, int>> find_pairs_product(int target, int max_val) {
    vector<pair<int, int>> pairs;
    for (int i = 1; i <= max_val; ++i) {
        for (int j = 1; j <= max_val; ++j) {
            if (i * j == target) {
                pairs.push_back({i, j});
            }
        }
    }
    return pairs;
}
```

```cpp
int main() {
    int target = 100;
    int max_val = 100;

    vector<pair<int, int>> result = find_pairs_product(target, max_val);

    cout << "두 수의 곱이 " << target << "인 경우:" << endl;
    for (auto& p : result) {
        cout << "(" << p.first << ", " << p.second << ")" << endl;
    }

    return 0;
}
```

> 💡 컴파일: `g++ -o brute_force brute_force.cpp`
> 실행: `./brute_force`

⚙️ C 예제 (간단한 버전)

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int target = 100;
    int max_val = 100;

    printf("두 수의 곱이 %d인 경우:\n", target);
    for (int i = 1; i <= max_val; i++) {
        for (int j = 1; j <= max_val; j++) {
            if (i * j == target) {
                printf("(%d, %d)\n", i, j);
            }
        }
    }

    return 0;
}
```

> 컴파일: `gcc -o brute_force.c`
> 실행: `./brute_force`

📌 사용 팁

- **작은 데이터 범위**에서만 유용 (예: 1~100)
- **대규모 문제**에서는 **브루트포스는 실용적이지 않음** → 다른 알고리즘 (예: 분할 정복, 동적 프로그래밍) 사용
- **암호 해독, 키 찾기, 조합 최적화** 등에 자주 사용됨

📚 관련 태그 (Obsidian에 추가할 경우)

- `#브루트포스`
- `#알고리즘`
- `#C`
- `#Python`
- `#C++`
- `#기초 알고리즘`

필요하시면 다음과 같은 추가 예제도 도와드릴 수 있습니다:

- 🔐 암호 키 찾기 (예: 4자리 패스워드)
- 🧩 부분 집합 합 (Subset Sum)
- 🔄 순열 조합 (Permutation)

원하시는 문제를 알려주시면 바로 예제를 만들어 드릴게요! 😊