2024-1
기초 스터디

6. 브루트 포스 & 시간 복잡도

목차

브루트 포스

- **Brute Force : Brute (무식한) + Force(힘)**

- 컴퓨터의 빠른 연산 속도(힘)를 사용하여 무식하게? 문제를 푸는 기법
- 문제의 정답으로 가능한 범위가 A ~ B 이고, 이 안에 답이 존재할 때, A ~ B 에 속하는 모든 값을 전부 시도하면서 정답을 찾는 기법

- 다른 이름으로는 **'완전 탐색'** 이라고도 합니다.

브루트 포스

- **Brute Force : Brute (무식한) + Force(힘)**
- 지금까지 배운 문법을 활용하여 모든 해답을 시도해보자

브루트 포스 – 연습문제 1

- https://www.acmicpc.net/problem/2798

블랙잭 성공 다국어

N장의 카드에 써져 있는 숫자가 주어졌을 때, M을 넘지 않으면서

M에 최대한 가까운 카드 3장의 합을 구해 출력하시오.

브루트 포스 – 연습문제 1

- 이 문제를 브루트 포스로 풀려면 어떻게 해야 할까?

브루트 포스 – 연습문제 1

- 모든 경우의 수를 탐색해봅시다.

  → N장의 카드 중에서 3장을 뽑는 모든 경우의 수

브루트 포스 – 연습문제 1

1. N장의 카드 중에서 임의로 3장을 뽑는다.
2. 3장에 적힌 숫자 합이 M 이하라면 정답 후보이다.
3. 기존 정답과 비교해서 M에 더 가깝다면 새로운 정답으로 갱신한다.
4. 3장을 뽑는 모든 경우의 수에 대해 1 ~ 3 을 반복한다.

브루트 포스 – 연습문제 1

- 같이 풀어봅시다.

브루트 포스 – 연습문제 1

- 정답 코드

```python
N, M = map(int, input().split())
nums = list(map(int, input().split()))

ans = 0
for i in range(N):
    for j in range(i+1, N):
        for k in range(j+1, N):
            s = sum([nums[i], nums[j], nums[k]])
            if s <= M:
                ans = max(ans, s)
print(ans)
```

브루트 포스 – 연습문제 2

- https://www.acmicpc.net/problem/1436

종말의 수란 어떤 수에 6이 적어도 3개 이상 연속으로 들어가는 수를 말한다. 제일 작은 종말의 수는 666이고, 그 다음으로 큰 수는 1666, 2666, 3666, ... 이다. 따라서, 숌은 첫 번째 영화의 제목은 "세상의 종말 666", 두 번째 영화의 제목은 "세상의 종말 1666"와 같이 이름을 지을 것이다. 일반화해서 생각하면, N번째 영화의 제목은 세상의 종말 (N번째로 작은 종말의 수) 와 같다.

숌이 만든 N번째 영화의 제목에 들어간 수를 출력하는 프로그램을 작성하시오. 숌은 이 시리즈를 항상 차례대로 만들고, 다른 영화는 만들지 않는다.

브루트 포스 – 연습문제 2

- 어떻게 문제를 풀 수 있을까?

- 규칙 찾기?
- 666, 1666, 2666, 3666, ..., 10666, 11666
- 단순히 증가하는 숫자 뒤에 666붙이기?
- 그렇다면 6661 은 어떻게 처리할까?

브루트 포스 – 연습문제 2

- 복잡하게 생각하지 말고, 무식하게 종말의 수를 세어보자.

1. 666부터 시작해서 숫자를 1씩 증가시킨다.
2. 해당 숫자를 문자열로 바꾼 뒤, ‘666’ 이 들어있는지 확인한다.
3. ‘666’이 들어있다면 종말의 수 이므로 카운트 증가
4. 종말의 수 카운트가 N이라면 N번째 종말의 수를 출력

브루트 포스 – 연습문제 2

- 이렇게 풀어도 되나요?
- 숫자가 100만, 1000만까지 가면 시간이 오래 걸리지 않을까요?

브루트 포스 – 연습문제 2

- 컴퓨터는 우리 생각보다 아주 빠르다!
- 알고리즘 문제를 풀 때, **컴퓨터는 1초에 1억 번 연산**을 할 수 있다고 가정한다.

```c
#include <stdio.h>

int main(void) {
    int i = 0;
    for (;i<100000000;i++);
    printf("%d", i);
    return 0;
}

```

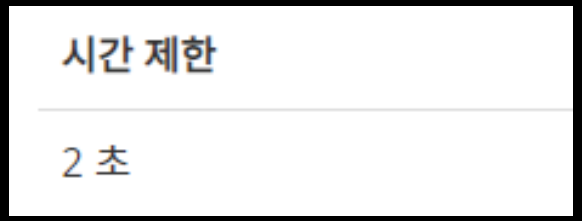
input Output

Success #stdin #stdout 0s 5308KB

100000000

브루트 포스 – 연습문제 2

- Q. 문제의 제한 시간이 2초 던데, 종말의 수가 2억을 넘어가면요?



브루트 포스 – 연습문제 2

- 한번 계산해봅시다.

브루트 포스 – 연습문제 2

- 맨 처음 추측했던 규칙처럼,
  종말의 수를 어떤 숫자 뒤에 666을 붙인 수라고 하자.
- N번째 종말의 수는 “N” + “666” 이다.

브루트 포스 – 연습문제 2

- N = 10000 일 때, N번째 종말의 수는 **“10000666”**
- 이 수는 1억보다 작으므로,
  1부터 10000666까지의 수를 1초 안에 확인할 수 있다.

- 게다가 10000번째 종말의 수는 실제로 이보다 더 작은 범위에 있다.
  6661, 6662 와 같이 666이 중간에 들어있는 경우도 있기 때문이다.

브루트 포스 – 연습문제 2

- Q. 2초 안에 10000666번 돌 수 있는 건 알겠어요.
  그런데, 1번 돌 때 얼마나 실행될지 어떻게 아나요?
  1번 돌 때, 20번만 연산해도 2억 번이 넘는데요?

브루트 포스 – 연습문제 2

- A. 정확한 연산량을 계산하는 것은 힘들기 때문에,
  시간 복잡도를 이용합니다.
  시간 복잡도를 이용하면 우리가 생각한 알고리즘을 구현한 프로그램이 연산을 몇 번 하는지 **예측**할 수 있습니다.

브루트 포스 – 시간 복잡도

- 우리가 작성한 프로그램이 답을 구하기까지 **최대** **몇 번의 연산**을 하는지 예측할 때 **‘시간복잡도’**라는 개념을 사용한다.

- 시간복잡도는 N이라는 입력으로부터 원하는 결과를 얻는데 걸리는 **최악**의 시간을 **O(g(N))**로 표기한다.
  (이를 ‘빅-O 표기법’ 이라고 합니다. 영어로는 ‘Big-O notation’)

브루트 포스 – 시간 복잡도

- n이라는 입력이 주어졌을 때, 이를 활용하여 어떤 문제를 푸는데 필요한 코드 실행 횟수를 **f(n)** 이라고 하자.
- 이때 어떤 특정 n_1 이후의 모든 n에 대해, (즉, $n_1 \leq n$) $f(n) \leq c*g(n)$ 을 만족하도록 하는 (n_1, c) 쌍이 존재하면 (c는 양의 상수)
- 이때의 g(n)에 대해 **O(g(n))** 이라고 쓸 수 있다.

이게 무슨 말이죠..?

- n개 데이터로 구성된 문제를 푸는 연산량이 **f(n)** 일 때, 일반적으로 n이 증가하면 연산량 **f(n)**도 같이 증가한다.
- 이때 어떤 시점 이후로는 n이 아무리 증가해도 f(n) 이 c*g(n) 은 안 넘는다고 표현할 수 있다면 **f(n)은 O(g(n))의 시간 복잡도를 갖는다**고 말한다.

시간 복잡도 계산 예시

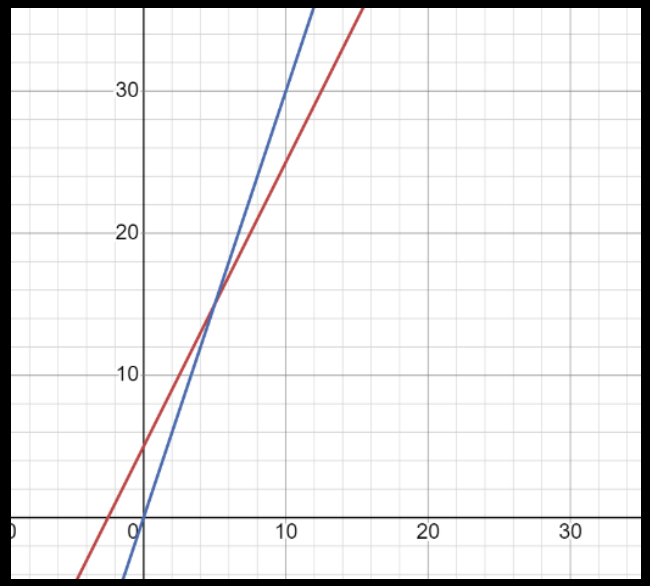
- n이라는 입력이 주어진 어떤 문제를 풀고자 한다. 이때 필요한 연산량이 2n + 5 라고 하자.

```
N = int(input())
for i in range(2*N+5):
    print(i)
```

예를 들면, 위 코드의 실행 횟수는 2N + 5 이다.

시간 복잡도 계산 예시

- 이 코드의 실행 횟수 (2n+5)는 n이 증가함에 따라 어느 순간 (n = 5)부터는 3n보다 항상 작거나 같다.

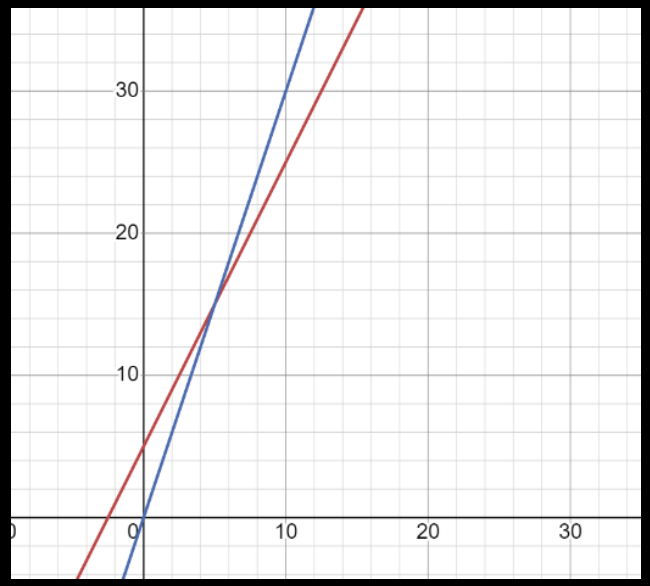


$2n + 5$

$3n$

시간 복잡도 계산 예시

- 따라서 $n_1 = 5$ 일 때, $n_1 \leq n$ 인 모든 n에 대해 **$f(n) = 2n + 5 \leq 3n$** 을 만족한다.



$2n + 5$

$3n$

시간 복잡도 계산 예시

- 어떤 특정 n_1 이후의 모든 n에 대해, ($n_1 \le n$)
  $f(n) \le c*g(n)$ 을 만족하는 (n_1, c) 쌍이 존재한다!
- $n_1 = 5$
- c = 3
- 이때의 g(n) = **n** → **O(g(n)) = O(n)**

시간 복잡도 계산 예시

- 따라서 2n + 5 의 연산량을 갖는 프로그램의 시간복잡도는 **O(n)** 으로 표현할 수 있습니다.

시간 복잡도 계산 예시

- Q. n_1 = 4, 3, 2, 1, ... 일 때는 2n+5 가 3n 보다 큰데요?
- A. Big-O 표기법은 **'최악의 경우'의 연산량**을 따지기 때문에 **5 ≤ n** 이든 **1억 ≤ n** 이든 **어떤 값 이후** 모든 n에 대해서 f(n) ≤ c * g(n) 을 만족하기만 하면 쓸 수 있어요.

그래도 모르겠어요..

- 쉽게 정리하자면
  어떤 프로그램의 실행 횟수를 입력 데이터 크기 n에 대해서 표현했을 때, 그 식의 **최고차항**을 g(n)으로 보면 됩니다.
- $2n + 50000$ → $O(n)$
- $2n^2 + n + 100$ → $O(n^2)$
- $2 * \log(n) + 10$ → $O(\log n)$

브루트 포스 – 시간 복잡도 예시

만약 N개의 데이터가 주어졌을 때,
원하는 결과를 얻기까지 필요한 연산의 횟수가

- N값에 상관없이 **몇 번**만으로 된다 → O(1)
- **N에 비례**한다. → O(N)
- **N^2에 비례**한다. → $O(N^2)$

브루트 포스 – 시간 복잡도 예시

N값에 상관없이 **몇 번**만으로 된다 → **O(1)**

https://www.acmicpc.net/problem/24262

MenOfPassion 알고리즘은 다음과 같다.

```
MenOfPassion(A[], n) {
    i = ⌊n / 2⌋;
    return A[i]; # 코드1
}
```

브루트 포스 – 시간 복잡도 예시

- **N에 비례**한다. → **O(N)**
- https://www.acmicpc.net/problem/24263

MenOfPassion 알고리즘은 다음과 같다.

```
MenOfPassion(A[], n) {
    sum <- 0;
    for i <- 1 to n
        sum <- sum + A[i]; # 코드1
    return sum;
}
```

- sum을 구하는 반복문이 **N번 실행**된다.
  따라서 시간 복잡도는 **O(N)**

브루트 포스 – 시간 복잡도

- **시간복잡도에 가능한 N의 최댓값을 대입**하면
  최대 연산 횟수를 대략적으로 예측할 수 있습니다.

- 만약 $O(N^2)$ 시간복잡도를 가지는데,
  주어지는 N이 최대 10000 이라면,
  최대 10000^2 = 1억 번의 연산을 한다고 생각합니다.

브루트 포스 – 시간 복잡도

- Q. 프로그램 연산량이 **n + 1억** 이면요?
  n이 최대 1억이면 최대 2억 번 연산하는 건데,
  O(n) 이라서 1억으로 예측하면 안되는 것 아닌가요?

- A. 맞습니다! 하지만 그렇게 상수가 너무 커서
  시간 복잡도로 예측한 연산량을 크게 벗어나는
  경우는 거의 없습니다. (저도 그런 건 아직 안 풀어봤어요)

브루트 포스 – 연습문제 2

- 다시 연습 문제로 돌아와서 N번째 종말의 수를 구하는 알고리즘의 시간 복잡도를 계산해봅시다.

브루트 포스 – 연습문제 2

1. 666에서 시작한다.
2. 현재 숫자가 종말의 수인지 확인한다.
3. 종말의 숫자라면 count 를 늘린다.
4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.
5. 아니라면 현재 숫자를 1 증가시키고 2로 돌아간다.

브루트 포스 – 연습문제 2

알고리즘을 보면 찾고자 하는 N번째 수에 따라 2~4를 최대 **10,000,000번 반복**한다.

(666 ~ 10,000,666 범위를 탐색한다고 가정하고 있기 때문)

브루트 포스 – 연습문제 2

이제 한번 반복할 때의 시간 복잡도를 구해보자.

2. 현재 숫자가 종말의 수인지 확인한다.
3. 종말의 숫자라면 count 를 늘린다.
4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.

브루트 포스 – 연습문제 2

2. 현재 숫자가 종말의 수인지 확인한다.

3. 종말의 숫자라면 count 를 늘린다.

4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.

```
1  if is_end_number:
2      count += 1
3
4      if count == N:
5          break
```

3, 4번 과정은 **O(1)** 에 할 수 있다.

브루트 포스 – 연습문제 2

2. 현재 숫자가 종말의 수인지 확인한다.
3. 종말의 숫자라면 count 를 늘린다.
4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.

현재 숫자가 종말의 숫자인지 어떻게 확인할까?

브루트 포스 – 연습문제 2

1. 현재 숫자를 문자열로 바꾼다.
2. 문자열 안에 ‘666’ 이 들어있는지 확인한다.

문자열을 확인하는 과정은 문자열의 길이와 관련이 있다.

문자열은 최대 ‘10000666’ **8글자** 이다.

브루트 포스 – 연습문제 2

8글자 문자열 안에 '666' 이 들어있는지 확인하는 것은

```
for i in range(8-2):
    if string[i+0] == '6':
    if string[i+1] == '6':
    if string[i+2] == '6':
```

대충 계산했을 때 6 * 3 = **18번**의 코드를 실행한다.

브루트 포스 – 연습문제 2

M글자 문자열 안에 '666' 이 들어있는지 확인하는 것은

```
1  M = len(string)
2  for i in range(M-2):
3      if string[i+0] == '6':
4      if string[i+1] == '6':
5      if string[i+2] == '6':
```

대충 계산했을 때 **3*(M-2) 번**의 코드를 실행한다.

따라서 **O(M)** 의 시간복잡도를 갖는다고 볼 수 있다.

브루트 포스 – 연습문제 2

2. 현재 숫자가 종말의 수인지 확인한다.
3. 종말의 숫자라면 count 를 늘린다.
4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.

2, 3, 4번 과정은 O(M) + O(1) = **O(M)** 에 할 수 있다.

브루트 포스 – 연습문제 2

1. 666에서 시작한다.
2. 현재 숫자가 종말의 수인지 확인한다.
3. 종말의 숫자라면 count 를 늘린다.
4. count가 N이면 (N번째 종말의 숫자라면) 멈춘다.
5. 아니라면 현재 숫자를 1 증가시키고 2로 돌아간다.

따라서 이 과정은 최대 **10,000,000 * O(M)** 안에 할 수 있다.

M ≤ 8 이므로, 연산 횟수는 최대 **8천만**

브루트 포스 – 연습문제 2

- 시간 복잡도를 계산하여, 이 알고리즘을 사용하면 문제를 2초 내에 풀 수 있음을 예측했습니다.
- 이제 같이 풀어봅시다.

브루트 포스 – 연습문제 2

- 정답 코드

```
1 n = int(input())
2 count = 1
3 number = 666
4 while count < n:
5     number += 1
6     if '666' in str(number):
7         count += 1
8 print(number)
9
```

추가 연습 문제

- https://www.acmicpc.net/step/53
- 시간복잡도 단계에 있는 모든 문제를 푸는 것을 권장드립니다.

- 추가 연습문제는 24265, 24267 만 넣었습니다!

추가 연습 문제 - 24265

- i = 1 일 때, j = 2부터 N까지 **N-1번** 반복하며 **코드1**을 실행합니다.
- i = 2 일 때, j = 3부터 N까지 **N-2번** 반복하며 **코드1**을 실행합니다.
- 따라서 **코드1**은 N-1, N-2, N-3, ..., 2, 1 번 실행됩니다. 등차수열의 합 공식을 이용하면 **N*(N-1)/2** 로 계산할 수 있습니다.

추가 연습 문제 - 24265

- 정답 코드

```python
n = int(input())
print(n*(n-1)//2)
print(2)
```

- N이 최대 50만이라, 주어진 함수를 직접 코드로 짜서 돌리면 시간초과가 발생합니다.

추가 연습 문제 - 24267

- 24265 문제를 풀면서 세웠던 식의 **n**에 1부터 n-1 수를 모두 대입하고 더한 것과 같습니다.

- N^2 의 식이 N번 반복되므로,
  코드 실행 횟수의 최고차항 계수는 3입니다.
  (물론 N에 대해 실행 횟수를 수식으로 나타낼 수도 있습니다.)

추가 연습 문제 - 24267

- 정답 코드

```python
n = int(input())
s = 0
for i in range(n):
    s += int(i*(i-1)/2)
print(s)
print(3)
```

```python
n = int(input())
print(n*(n-1)*(n-2)//6)
print(3)
```

- N이 최대 50만이라 반복문을 한번 돌려도 되고, 수식을 계산해서 풀어도 됩니다.

브루트 포스 – 이번주 연습문제

- 1018 체스판 다시 칠하기
- 19532 수학은 비대면강의입니다
- 2231 분해합
- 11478 서로 다른 부분 문자열의 개수
- 1065 한수
- 3100 국기 인식
- 2503 숫자 야구
- **2422 한윤정이 이탈리아에 가서 아이스크림을 사먹는데**

브루트 포스 – 이번주 연습문제

- 2422 힌트
- 지금 보고 있는 아이스크림 조합이 가능한 조합인지 확인할 때 보통 이 숫자가 어떤 숫자 데이터 그룹에 포함되는지 확인합니다.
- 혹시 시간초과가 난다면, **‘포함 여부를 빠르게 확인할 수 있는 자료구조’** 가 무엇인지 고민해보세요. (지난 주 강의록 참고!)
- 아이스크림 개수가 200개로 적기 때문에 2차원 리스트를 사용해도 빠르게 확인할 수 있어요