컴퓨팅 사고와 SW 코딩

05

완전 탐색

경북대학교 배준현 교수
(joonion@knu.ac.kr)

05. 완전 탐색

- **완전 탐색:** *exhaustive* search
  - 모든 가능한 경우의 수를 전부 다 탐색해 보는 방법
    - 단순무식하게 푸는 방법: *Brute-Force*
  - 장점:
    - 답이 있는 모든 문제를 해결할 수 있는 일반적인 문제해결 방법
  - 단점:
    - 시간이 너무 많이 걸리므로 상당수의 문제들은 해결이 불가능 (시간 초과)

05. 완전 탐색

BOJ 1929: 소수 구하기

- 문제:
  - M이상 N이하의 소수를 모두 출력하는 프로그램을 작성하시오.
- 입력:
  - 첫째 줄에 자연수 M과 N이 빈 칸을 사이에 두고 주어진다.
    - $(1 \le M \le N \le 1{,}000{,}000)$
  - M이상 N이하의 소수가 하나 이상 있는 입력만 주어진다.
- 출력:
  - 한 줄에 하나씩, 증가하는 순서대로 소수를 출력한다.

05. 완전 탐색

예제 입력

```
3  16
```

예제 출력

```
3
5
7
11
13
```

05. 완전 탐색

```python
def is_prime(n):
    if n < 2:
        return False
    else:
        for k in range(2, int(n ** 0.5) + 1):
            if n % k == 0:
                return False
        return True

def solve(m, n):
    for k in range(m, n + 1):
        if is_prime(k):
            print(k)

M, N = map(int, input().split())
solve(M, N)
```

05. 완전 탐색

BOJ 1259: 팰린드롬 수

- 문제:
  - 어떤 수를 뒤에서부터 읽어도 똑같다면 그 수를 팰린드롬 수라고 한다.
  - 121, 12421 등은 팰린드롬수다.
  - 123, 1231은 뒤에서부터 읽으면 다르므로 팰린드롬수가 아니다.
  - 이번 문제에서는 무의미한 0이 앞에 올 수 없다고 하자.
- 입력:
  - 입력은 여러 개의 테스트 케이스로 이루어져 있으며, 각 줄마다 1 이상 99999 이하의 정수가 주어진다.
  - 입력의 마지막 줄에는 0이 주어지며, 이 줄은 문제에 포함되지 않는다.
- 출력:
  - 각 줄마다 주어진 수가 팰린드롬 수이면 'yes', 아니면 'no'를 출력한다.

05. 완전 탐색

예제 입력

```
121
1231
12421
0
```

예제 출력

```
yes
no
yes
```

05. 완전 탐색

```python
def is_palindrome(n):
    s = str(n)
    return s == s[::-1]

while True:
    n = int(input())
    if n == 0:
        break
    print("yes" if is_palindrome(n) else "no")
```

05. 완전 탐색

BOJ 1747: 소수인 팰린드롬

- 문제:
  - 어떤 수와 그 수의 숫자 순서를 뒤집은 수가 일치하는 수를 팰린드롬이라 부른다.
    - 예를 들어 79,197과 324,423 등이 팰린드롬 수이다.
  - 어떤 수 N ($1 \le N \le 1{,}000{,}000$)이 주어졌을 때,
    - N보다 크거나 같고, 소수이면서 팰린드롬인 수 중에서,
    - 가장 작은 수를 구하는 프로그램을 작성하시오.
- 입력:
  - 첫째 줄에 N이 주어진다.
- 출력:
  - 첫째 줄에 조건을 만족하는 수를 출력한다.

05. 완전 탐색

예제 입력	예제 출력
31	101

05. 완전 탐색

- 어느 쪽을 먼저 검사하는 것이 더 효율적일까?

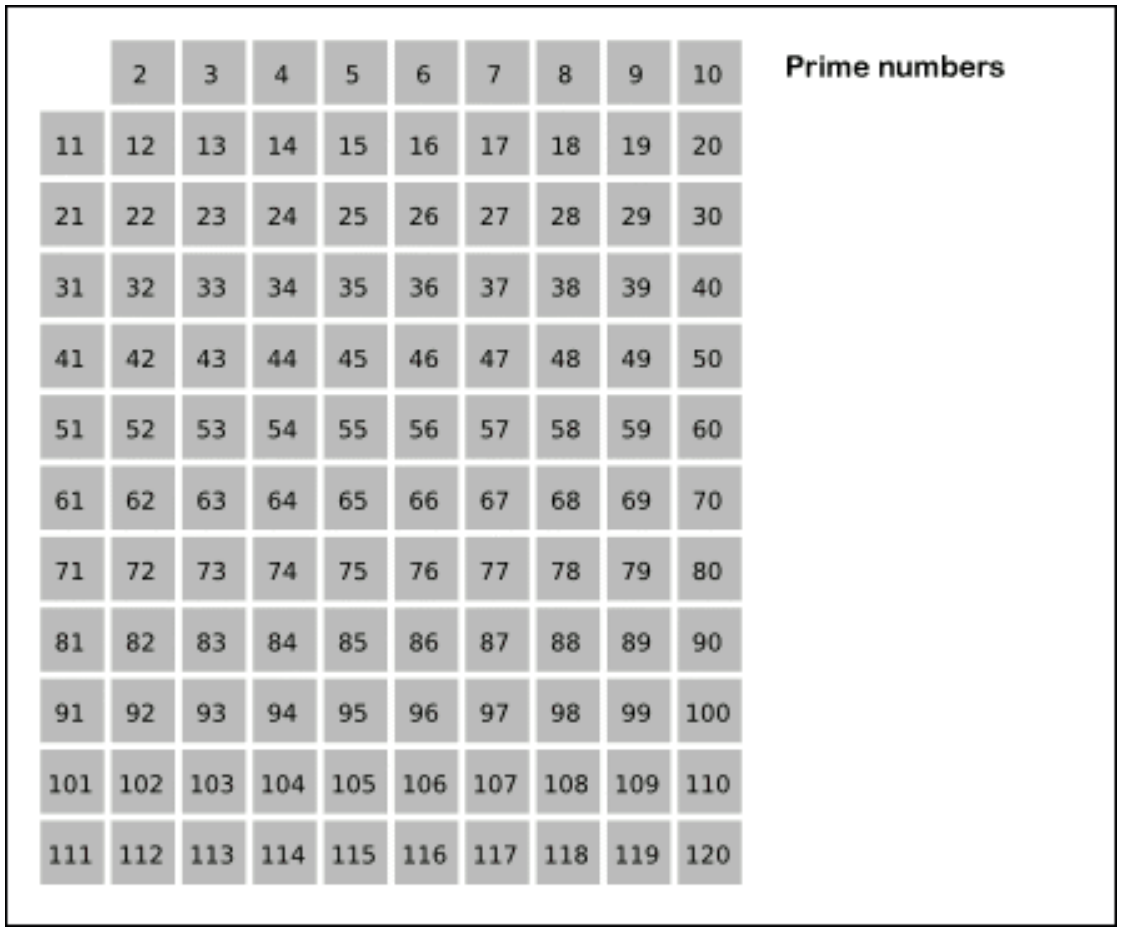
```
def solve(n):
    while True:
        if is_prime(n) and is_palindrome(n):
            return n
        n += 1

N = int(input())
print(solve(N))
```

05. 완전 탐색

- **에라토스테네스의 체: *Sieve* of *Eratosthenes***
  - N보다 작거나 같은 모든 소수를 효율적으로 찾는 알고리즘
    1. 2부터 N까지 모든 정수를 적는다.
    2. 아직 지우지 않은 수 중 가장 작은 수를 찾는다.
       - 이 수를 P라고 하고, 이 수는 소수이다.
    3. P를 지우고, 아직 지우지 않은 P의 배수를 크기 순서대로 지운다.
    4. 아직 모든 수를 지우지 않았다면, 다시 2번 단계로 간다.

05. 완전 탐색



https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes#/media/File:Sieve_of_Eratosthenes_animation.gif

05. 완전 탐색

```
def find_primes(n):
    sieve = [0] * (n + 1)
    for i in range(2, int(n ** 0.5) + 1):
        if sieve[i] == 0:
            for j in range(i * i, n + 1, i):
                sieve[j] = 1
    return [i for i in range(2, n + 1) if sieve[i] == 0]

print(find_primes(10))
print(find_primes(100))
```

05. 완전 탐색

BOJ 4948: 베르트랑 공준

- 문제:
  - 임의의 자연수 n에 대하여, n보다 크고, 2n보다 작거나 같은 소수는 적어도 하나 존재한다.
    - 10보다 크고, 20보다 작거나 같은 소수는 4개가 있다. (11, 13, 17, 19)
    - 14보다 크고, 28보다 작거나 같은 소수는 3개가 있다. (17,19, 23)
  - n보다 크고, 2n보다 작거나 같은 소수의 개수를 구하는 프로그램을 작성하시오.
- 입력:
  - 입력은 여러 개의 테스트 케이스로 이루어져 있다.
  - 각 케이스는 n을 포함하는 한 줄로 이루어져 있다.
  - 입력의 마지막에는 0이 주어진다.
- 출력:
  - 각 테스트 케이스에 대해서, n보다 크고, 2n보다 작거나 같은 소수의 개수를 출력한다.

05. 완전 탐색

예제 입력

```
1
10
13
100
1000
10000
100000
0
```

예제 출력

```
1
4
3
21
135
1033
8392
```

05. 완전 탐색

```
def find_primes(m):
    n = 2*m
    sieve = [0] * (n + 1)
    for i in range(2, int(n ** 0.5) + 1):
        if sieve[i] == 0:
            for j in range(i * i, n + 1, i):
                sieve[j] = 1
    return [i for i in range(m + 1, n + 1) if sieve[i] == 0]

while True:
    n = int(input())
    if n == 0: break
    print(len(find_primes(n)))
```

05. 완전 탐색

- BOJ 1969: DNA
  - 문제:
    - DNA란 어떤 유전물질을 구성하는 분자이다. DNA는 서로 다른 4가지의 뉴클레오티드로 이루어져 있다
      - Adenine, Thymine, Guanine, Cytosine.
    - 어떤 DNA의 물질을 표현할 때, 이 DNA를 이루는 뉴클레오티드의 첫글자를 따서 표현한다.
      - Thymine-Adenine-Adenine-Cytosine-Thymine-Guanine-Cytosine-Cytosine-Guanine-Adenine-Thymine 로 이루어진 DNA를
      - “TAACTGCCGAT”로 표현

05. 완전 탐색

- Hamming Distance란 길이가 같은 두 DNA가 있을 때,
  - 각 위치의 뉴클오티드 문자가 다른 것의 개수이다.
  - "AGCAT" 와 "GGAAT"는 첫 번째 글자와 세 번째 글자가 다르므로 Hamming Distance는 2이다.
- 우리가 할 일은 다음과 같다.
  - N개의 길이 M인 DNA $s_1, s_2, \ldots, s_n$가 주어져 있을 때
  - Hamming Distance의 합이 가장 작은 DNA s를 구하는 것이다.
  - 즉, s와 s_1의 Hamming Distance + s와 s_2의 Hamming Distance + s와 s_3의 Hamming Distance ... 의 합이 최소가 된다는 의미이다.

05. 완전 탐색

05. 완전 탐색

05. 완전 탐색

05. 완전 탐색

- **순열과 조합**
  - **순열**: permutations
  - **조합**: combinations

05. 완전 탐색

05. 완전 탐색

05. 완전 탐색

05. 완전 탐색

- **더 풀어볼 문제:**
  - BOJ 2581: 소수
  - BOJ 14561: 회문
  - BOJ 1990: 소수인 팰린드롬
  - BOJ 6588: 골드바흐의 추측
  - BOJ 2960: 에라토스테네스의 체
  - BOJ 2529: 부등호
  - BOJ 2503: 숫자 야구

Any Questions?