07-1 소수 구하기

37 - 소수 구하기(실버3)

39 - 소수 & 팰린드롬 수 중에서 최솟값 찾기(골드5)

40 - 제곱이 아닌 수 찾기 (골드1)

박송이

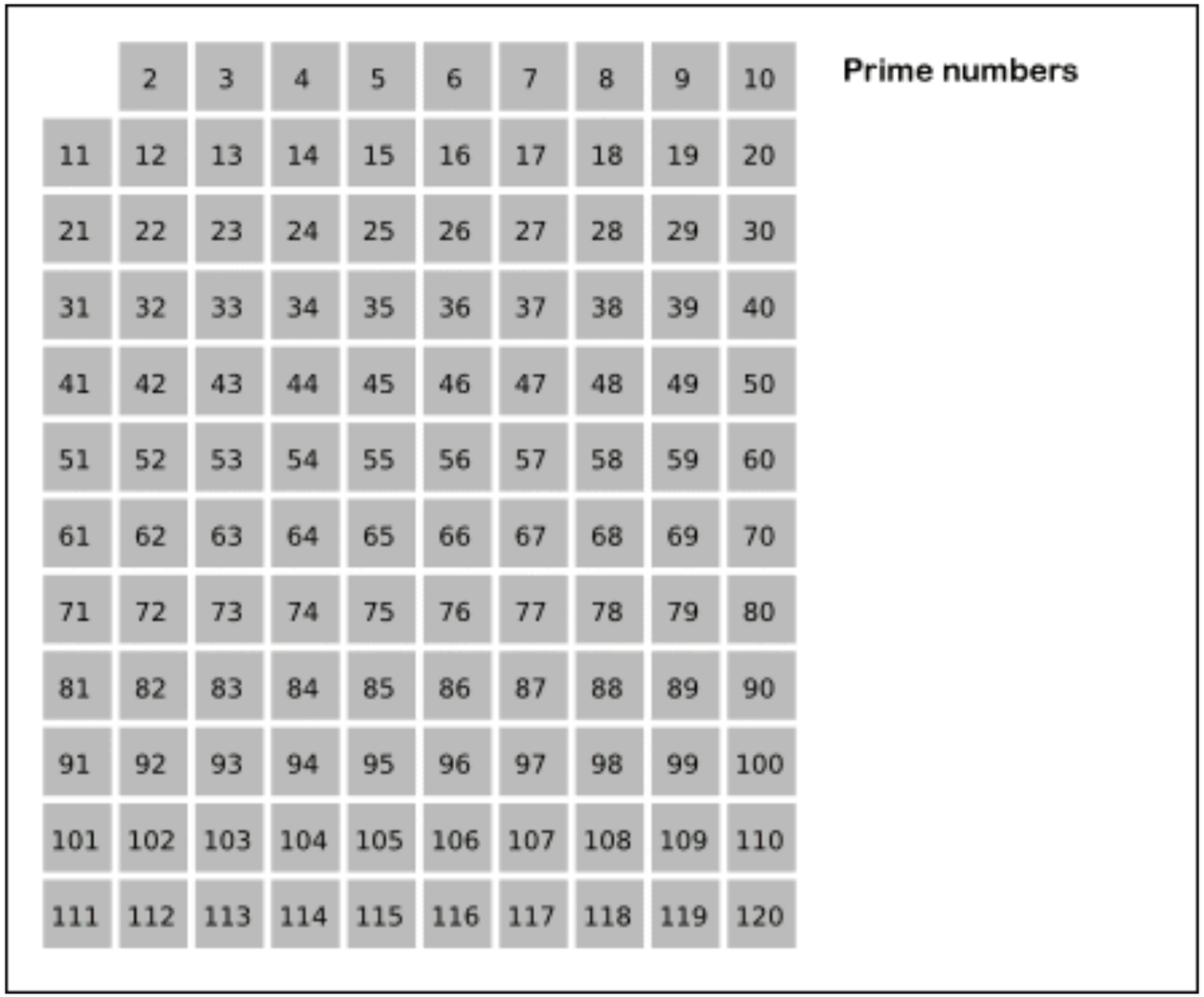
기본 개념

에라토스테네스의 체 원리

- 소수 : 1과 자기 자신 외에 약수가 존재하지 않는 수
- 코딩테스트에서는 소수를 판별하는 방식을 묻는 소수 구하기 문제가 종종 출제됨.
- **에라토스테네스의 체 원리**
  - 1) 구하고자 하는 소수의 범위만큼 1차원 리스트를 생성한다.
  - 2) 2부터 시작하고 현재 숫자가 지워진 상태가 아닌 경우 현재 선택된 숫자의 배수에 해당하는 수를 리스트에서 끝까지 탐색하면서 지운다. 이때 처음으로 선택된 숫자는 지우지 않는다.
  - 3) 리스트의 끝까지 2)를 반복한 후, 리스트에서 남아 있는 모든 수를 출력한다.

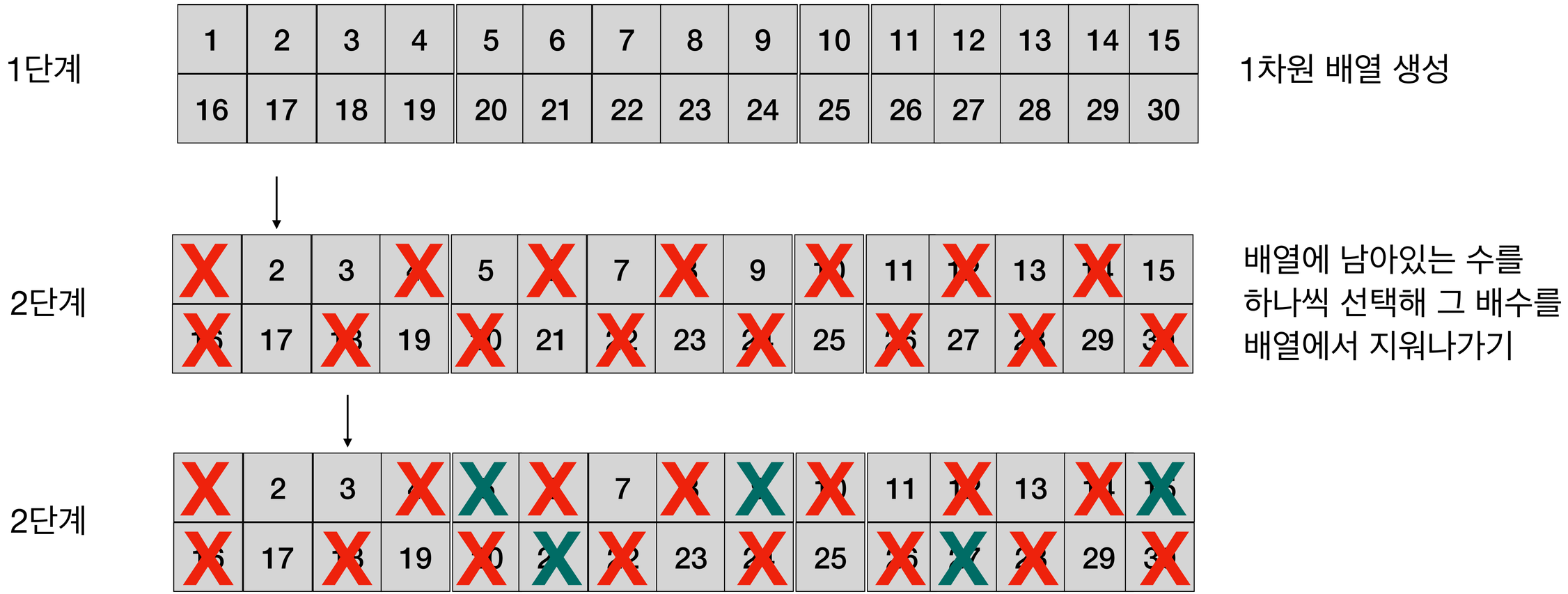
기본 개념

에라토스테네스의 체 원리



에라토스테네스의 체 원리(1)

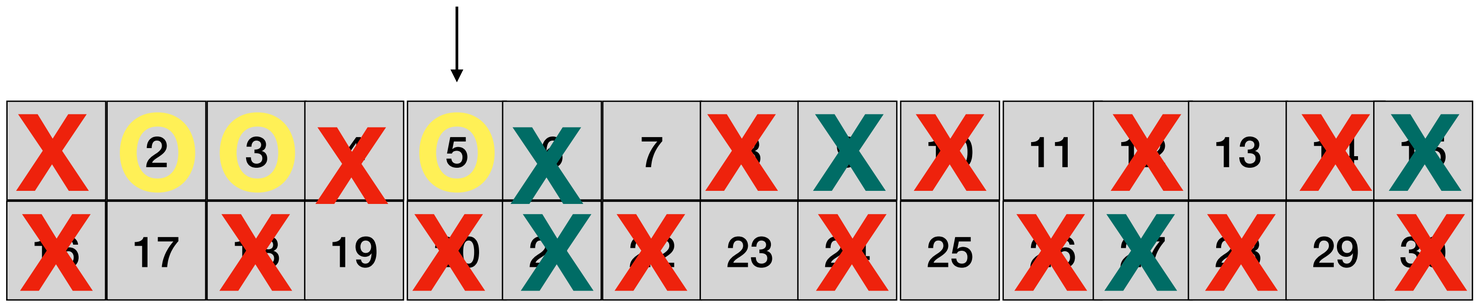
1부터 30까지의 수 중 소수를 구하는 예시



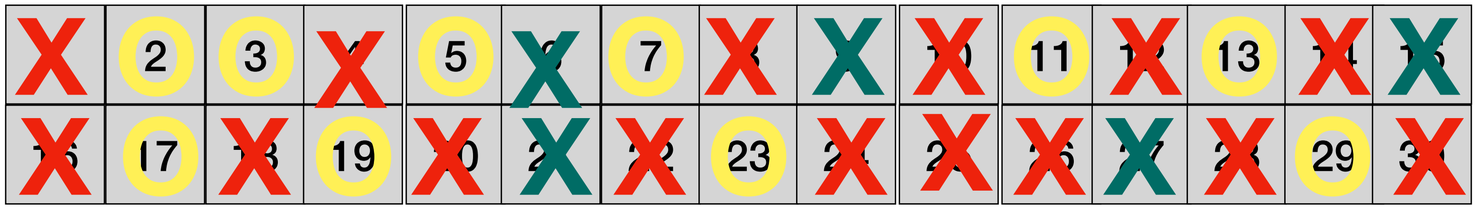
에라토스테네스의 체 원리 (2)

1부터 30까지의 수 중 소수를 구하는 예시

2단계



3단계



2단계 모두 반복 후
리스트에 남아있는 요소 출력

출력되는 수 : 2,3,5,7,11,13,17,19,23,29

문제 - 37번

M 이상 N 이하의 소수를 모두 출력하는 프로그램을 작성하기

예제 입력 1
3 16

예제 출력
3 5 7 11 13

[37] 소수 구하기


[1]
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
크기가 N+1인 리스트를 선언한다.
[2]
X 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
1인 소수가 아니므로 삭제한다.
[3]
X 2 3 X 5 X 7 X 9 X 11 X 13 X 15 X
2부터 N의 제곱근까지 탐색한다.
여기서는 16의 제곱근 4까지 탐색한다.
X 2 3 X 5 X 7 X X X 11 X 13 X X X
[4]
X 2 3 X 5 X 7 X X X 11 X 13 X X X
4까지 탐색 후 남아있는 수를 모두
출력한다.

[37] 소수 구하기

- **N의 제곱근까지만 탐색하는 이유**

N의 제곱근이 n일 때
N = a * b를 만족하는 a와 b가 모두 n보다 클 수 없다. (이때 a,b는 N의 약수가 된다)

ex) N=16 = 1*16 , 2*8, 4*4 (n=4)

a가 n보다 크다면 b는 n보다 작아야 한다.

즉 N보다 작은 수 가운데 소수가 아닌 수는 항상 n보다 작은 약수를 가진다.

따라서 에라토스테네스의 체로 n이하의 수의 배수를 모두 제거하면 1부터 N사이의 소수를 구할 수 있다.
(n이하의 수의 배수 => 어떤 수의 배수라는 것은 곧 그 수는 소수가 아니라는 뜻)

[37] 소수 구하기 ②

- 코드구현

```
import math
M,N = map(int,input().split())
A = [0] * (N+1)

for i in range(2,N+1):
    A[i] = i


for i in range(2,int(math.sqrt(N))+1):
    if A[i]==0:
        continue
    for j in range(i+i,N+1,i):
        A[j] = 0

for i in range(M,N+1):
    if A[i]!=0:
        print(A[i])
```

M은 시작값, N은 종료값

어차피 소수는 2부터 시작이므로 2부터 N+1까지의 수를 배열 A에 넣어준다.

앞에서 설명했듯이 N까지의 모든 수를 다 확인하는 것이 아니라
N의 제곱근까지만 확인한다.

i부터 시작해서 그 값을 i만큼 더해간다 => i의 배수를 확인해서
그 위치의 값을 0으로 해준다.

위에서 루프를 돌면서 어떤 수의 배수는 0으로 만들어 놓았고
시작위치 ~ 끝위치까지 확인하면서 값이 0이 아니라면 출력하지 않는다.

[37] 소수 구하기

```python
import math
M,N = map(int,input().split())
A = [0] * (N+1)

for i in range(2,N+1):
    A[i] = i
```

```python
for i in range(2,int(math.sqrt(N))+1):
    if A[i]==0:
        continue
    for j in range(i+i,N+1,i):
        A[j] = 0
```

M = 3, N=16

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

M = 3, N=16
for i in range(2,5)

i=2인 경우, for j in range(4,17,2)

0	0	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

0	0	2	3	0	5	0	7	0	9	0	11	0	13	0	15	0

i=3인 경우, for j in range(6,17,3)

0	0	2	3	0	5	0	7	0	9	0	11	0	13	0	15	0

0	0	2	3	0	5	0	7	0	0	0	11	0	13	0	0	0

문제 - 39번

어떤 수 N이 주어졌을 때 N보다 크거나 같고 소수이면서 팰린드롬인 수 중 가장 작은 수를 구하라

어떤 수 N(1 <=N <=1,000,000)인 수
즉 우리는 N보다 같거나 크면서 1,000,000 보다는 작은 수 중에서 제일 작은 팰린드롬 수를 찾아야 한다.

예제 입력 1
31

예제 출력
101

[39] 소수&팰린드롬 수 중에서 최솟값 찾기

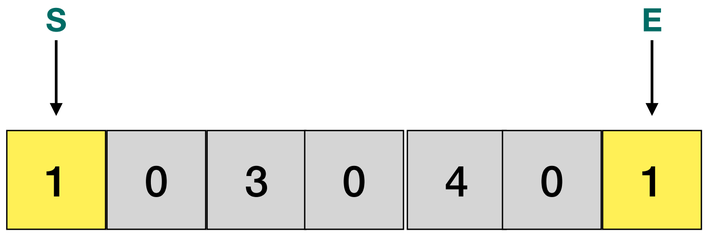
문제 풀이
[1단계] 최대 범위에 해당하는 모든 소수를 구하기
[2단계] 이 소수들의 집합에서 N보다 크거나 같으면서 팰린드롬인 수를 찾기

1단계 - 최대 범위에 해당하는 모든 소수를 구하기

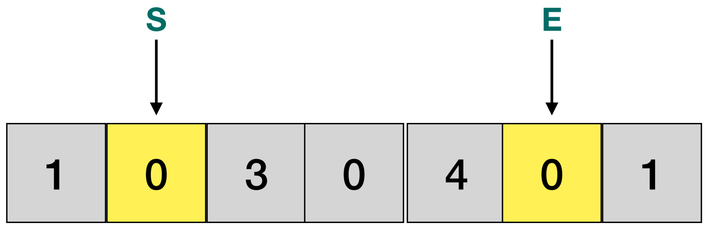
소수는 2부터 입력의 최대값인 1,000,000 사이에서 구한다.

[39] 소수&팰린드롬 수 중에서 최솟값 찾기

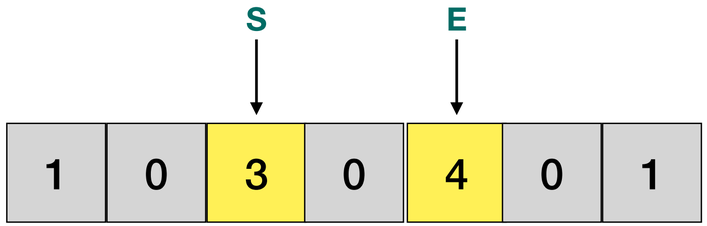
2단계 - 이 소수들의 집합에서 N보다 크거나 같으면서 팰린드롬인 수를 찾기



소수를 리스트 형태로 변환하고 양 끝의 투 포인터 지정하기



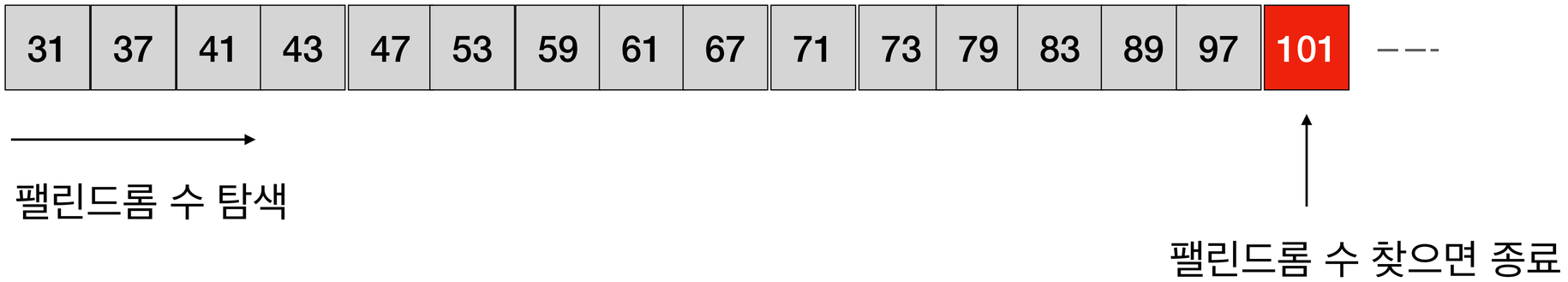
투 포인터가 가리키는 값이 같다면 S++, E- - 연산으로 포인터 이동시키기



S < E 를 만족할때까지 반복해서 모든 값이 같으면 팰린드롬 수로 판별한다.

[39] 소수&팰린드롬 수 중에서 최솟값 찾기

3단계 - 오름차순으로 과정 2를 실행하다가 최초로 팰린드롬 수가 나오면 종료



[39] 소수&팰린드롬 수 중에서 최솟값 찾기

```python
import math

N  = int(input())
A = [0] * (10000001)

for i in range(2,len(A)):
    A[i] = i

for i in range(2,int(math.sqrt(len(A)))+1):
    if A[i]==0:
        continue
    for j in range(i+i,len(A),i):
        A[j] = 0
```

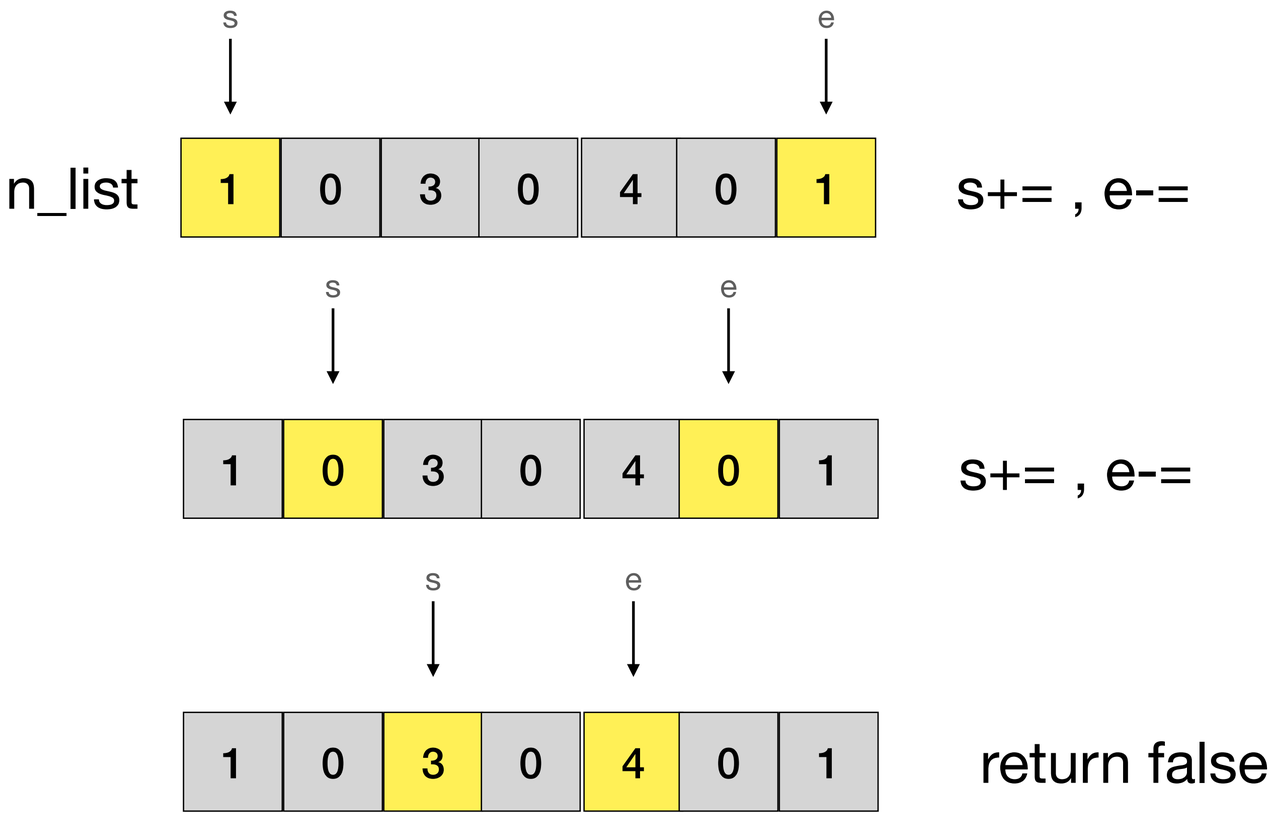
```python
def isPalindrome(target):
    n_list = list(map(int, str(target)))
    s = 0
    e = len(n_list)-1
    while s < e:
        if n_list[s]!=n_list[e]:
            return False
        s+=1
        e-=1
    return True
```

```python
while True:
    if A[i]!=0:
        result = A[i]
        if (isPalindrome(result)):
            print(result)
            break
    i +=1
```

[39] 소수&팰린드롬 수 중에서 최솟값 찾기

```python
def isPalindrome(target):
    n_list = list(map(int, str(target)))
    s = 0
    e = len(n_list)-1
    while s < e:
        if n_list[s]!=n_list[e]:
            return False
        s+=1
        e-=1
    return True
```

target = 1030401



[39] 소수&팰린드롬 수 중에서 최솟값 찾기

```
while True:
    if A[i]!=0:
        result = A[i]
        if (isPalindrome(result)):
            print(result)
            break
    i +=1
```

팰린드롬 수를 찾는 함수에
이전에 찾았던 소수들을 넣어서 확인하기

만약 팰린드롬 수를 찾는다면 그 결과를 출력하고
while문 종료하기

만약 A[i]값이 0이라면 i의 값을 하나 증가시켜주기

문제 - 40번

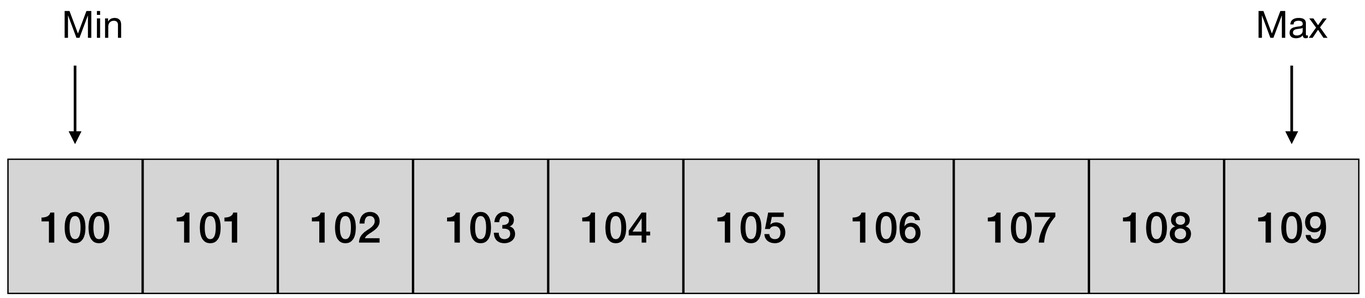
- 어떤 수 X가 1보다 큰 제곱수로 나누어떨어지지 않을 때 이 수를 ‘제곱이 아닌 수’라고 한다.
- min과 max가 주어졌을 때 이 사이에서 ‘제곱이 아닌 수’가 몇 개 있는지 출력하라

문제 풀이

[1단계] 최대 범위에 해당하는 모든 소수를 구하기

[2단계] 이 소수들의 집합에서 N보다 크거나 같으면서 팰린드롬인 수를 찾기

[40] 제곱이 아닌 수 찾기



2의 제곱수 4 -> 4, 8
3의 제곱수 9 -> 9
4의 제곱수 16
5의 제곱수 25로
나누어 떨어지지 않을 때 그 수를 ‘제곱이 아닌 수’라고 한다.

[40] 제곱이 아닌 수 찾기

```python
import math

Min ,Max = map(int,input().split())
Check = [False] * (Max - Min+1)

for i in range(2,int(math.sqrt(Max))+1):
    pow = i * i #제곱수
    start_index = int(Min/pow)
    if Min % pow !=0:
        start_index +=1
    for j in range(start_index, int(Max/pow)+1):
        Check[int((j*pow)-Min)] = True

count = 0

for i in range(0,Max-Min+1):
    if not Check[i]:
        count +=1

print(count)
```

기본적으로는 배열에 모든 요소를 False로 만들어준다.

전체 루프는 기존의 소수 찾기에서와 마찬가지로 Max의 제곱근+1까지만 찾음.

만약 그 수가 제곱수라면 값을 True로 바꿔준다.

최종적으로는 False인 요소만 count를 해서 출력한다.

[40] 제곱이 아닌 수 찾기 - 2

```
    pow = i * i #제곱수
    start_index = int(Min/pow)
    if Min % pow !=0:
        start_index +=1
    for j in range(start_index, int(Max/pow)+1):
        Check[int((j*pow)-Min)] = True
```

Min = 100
pow = 4
start_index = 25

for j in range(start_index=25, 28)

index	0	1	2	3	4	5	6	7	8	9
m ~ n	100	101	102	103	104	105	106	107	108	109
Check	false	false	false	false	false	false	false	false	false	false

j = 25 25 * 4 - 100 = 0 0번째 인덱스는 True

---

j = 26 26 * 4 - 100 = 4 4번째 인덱스는 True

---

j = 27 25 * 4 - 100 = 8 8번째 인덱스는 True

---

[40] 제곱이 아닌 수 찾기 - 3

```
    pow = i * i #제곱수
    start_index = int(Min/pow)
    if Min % pow !=0:
        start_index +=1
    for j in range(start_index, int(Max/pow)+1):
        Check[int((j*pow)-Min)] = True
```

Min = 100
pow = 9
start_index = 12

for j in range(start_index=12, 13)

index	0	1	2	3	4	5	6	7	8	9
m ~ n	100	101	102	103	104	105	106	107	108	109
Check	True	false	false	false	True	false	false	false	True	false

j = 12 12 * 9 - 100 = 8 8번째 인덱스는 True

[40] 제곱이 아닌 수 찾기 - 4

```
    pow = i * i #제곱수
    start_index = int(Min/pow)
    if Min % pow !=0:
        start_index +=1
    for j in range(start_index, int(Max/pow)+1):
        Check[int((j*pow)-Min)] = True
```

Min = 100
pow = 16
start_index = 7

for j in range(start_index=7, 7)

index	0	1	2	3	4	5	6	7	8	9
m ~ n	100	101	102	103	104	105	106	107	108	109
Check	True	false	false	false	True	false	false	false	True	false

실행되지 않음.

[40] 제곱이 아닌 수 찾기 - 5

```
    pow = i * i #제곱수
    start_index = int(Min/pow)
    if Min % pow !=0:
        start_index +=1
    for j in range(start_index, int(Max/pow)+1):
        Check[int((j*pow)-Min)] = True
```

Min = 100
pow = 25
start_index = 4

for j in range(start_index=4, 5)

index	0	1	2	3	4	5	6	7	8	9
m ~ n	100	101	102	103	104	105	106	107	108	109
Check	True	false	false	false	True	false	false	false	True	false

j = 4 4 * 25 - 100 = 0 0번째 인덱스는 True

[40] 제곱이 아닌 수 찾기

```
for i in range(0,Max-Min+1):
    if not Check[i]:
        count +=1

print(count)
```

index	0	1	2	3	4	5	6	7	8	9
m ~ n	100	101	102	103	104	105	106	107	108	109
Check	True	false	false	false	True	false	false	false	True	false