고 급 문 제 해 결

Chapter 4 & 5

발표 자료

2018117543 한재성

Contents

- **Chapter 4: Primitive types**
  - ✓문제 4.8 – 숫자 뒤집기
  - ✓문제 4.9 – 회문 확인하기
- **Chapter 5: Arrays**
  - ✓문제 5.3 – 주식 사고팔기

4.8 숫자 뒤집기

5.8 Reverse digits

Write a program which takes an integer and returns the integer corresponding to the digits of the input written in reverse order. For example, the reverse of 42 is 24, and the reverse of −314 is −413.

4.8 숫자 뒤집기(1/2)

- 간단하게 생각하기
  - ✓ Integer → String 변환
  - ✓ String.reverse()
  - ✓ String → Integer 변환

- 시간 복잡도?
  - ✓ $O(n)$

- 공간 복잡도?
  - ✓ $O(n)$

```
# Time Complexity: O(n)? O(logn)
# Space Somplexity: O(n)
def reverseNumber1(n: int) -> int:
    sign = 1
    out = str(n)  # 35 -> "35"
    # -35 -> "-35"

    out = out[::-1]  # "35" -> "53"
    # "-35" -> "53-"

    if n < 0:
        out = out[:-1]  # "53-" -> "53"
        sign = -1

    return sign * int(out)
```

4.8 숫자 뒤집기(2/2)

- 덜 간단하게 생각하기
  - ✓ N = 123456
  - ✓ S = 6
  - ✓ S = 60 + 5
  - ✓ S = 650 + 4
  - ✓ ….Shift 하는 느낌

- 시간 복잡도?
  - ✓ $O(n)$

- 공간 복잡도?
  - ✓ $O(1)$

```
# Time Complexity: O(n)
# Space Complexity: O(1)
def reverseNumber2(n: int) -> int:
    # 3212345 -> 5432123
    sign = 1
    if n < 0:
        sign = -1
        n = -n

    s = n % 10
    n //= 10
    while n > 0:
        s = s * 10 + (n % 10)
        n //= 10

    return sign * s
```

4.9 회문 확인하기

5.9 CHECK IF A DECIMAL INTEGER IS A PALINDROME

A palindromic string is one which reads the same forwards and backwards, e.g., "redivider". In this problem, you are to write a program which determines if the decimal representation of an integer is a palindromic string. For example, your program should return true for the inputs 0, 1, 7, 11, 121, 333, and 2147447412, and false for the inputs −1, 12, 100, and 2147483647.

Write a program that takes an integer and determines if that integer's representation as a decimal string is a palindrome.

4.9 회문 확인하기(1/2)

- 간단하게 생각하기
  - ✓ Integer → String 변환
  - ✓ isPalindrome(s) ?

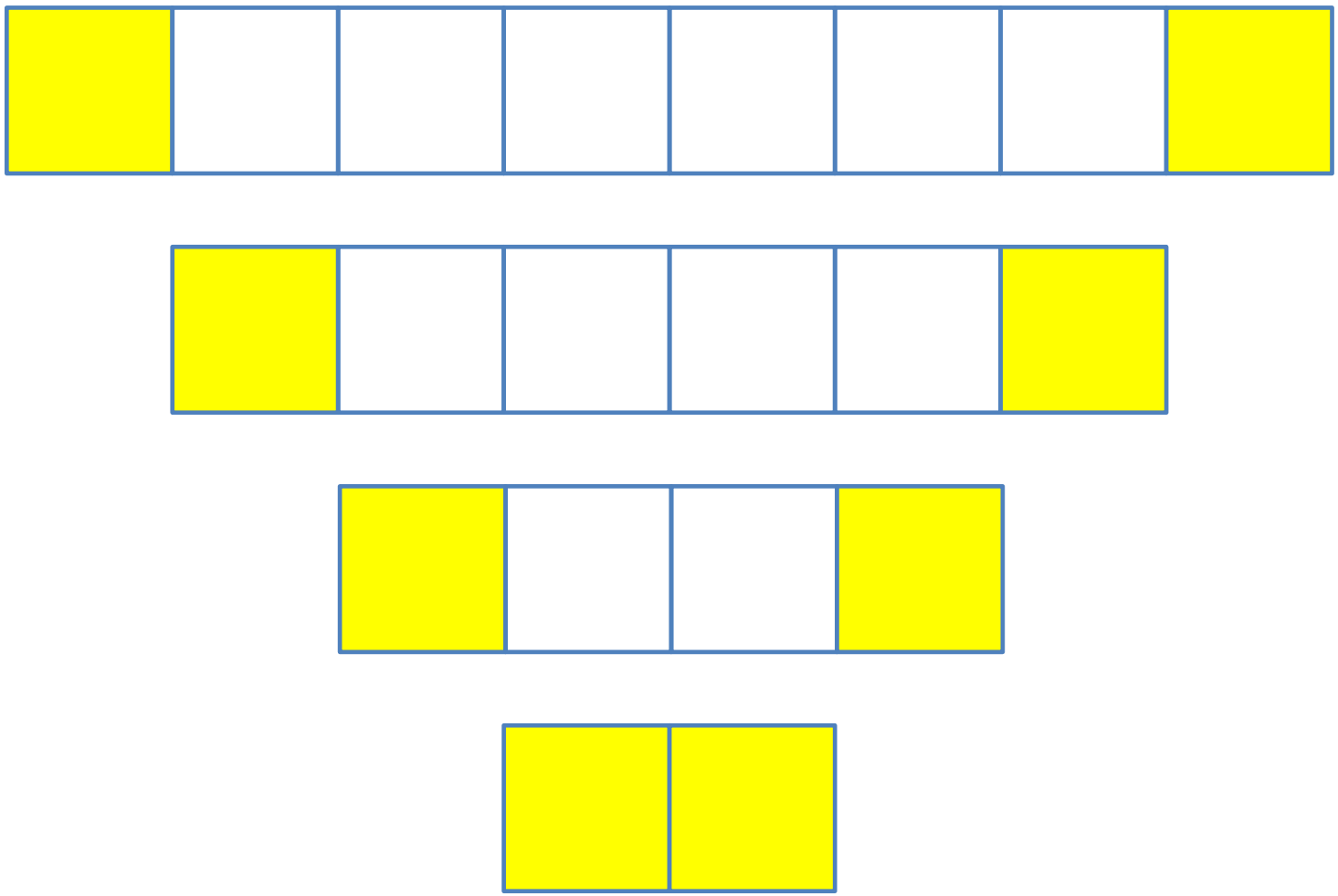
```python
# O(n)
def isPalindrome1(s: str) -> bool:
    left = 0
    right = len(s) - 1
    while left < right and s[left] == s[right]:
        left += 1
        right -= 1
    if left < right:
        return False
    return True
```

```
~/CautiousBusyWheel/lecture$ python3 4.9.py
True
True
1 vs 2 =  0.006163597106933594 0.004219293594360351
6
~/CautiousBusyWheel/lecture$ python3 4.9.py
True
True
1 vs 2 =  0.12514400482177734 0.11352849006652832
~/CautiousBusyWheel/lecture$ python3 4.9.py
True
True
1 vs 2 =  0.16024136543273926 0.11439251899719238
~/CautiousBusyWheel/lecture$
```

```python
# slicing => 속도가 위의 코드보다 훨씬 빠릅니다.
def isPalindrome2(s: str) -> bool:
    return s == s[::-1]
```

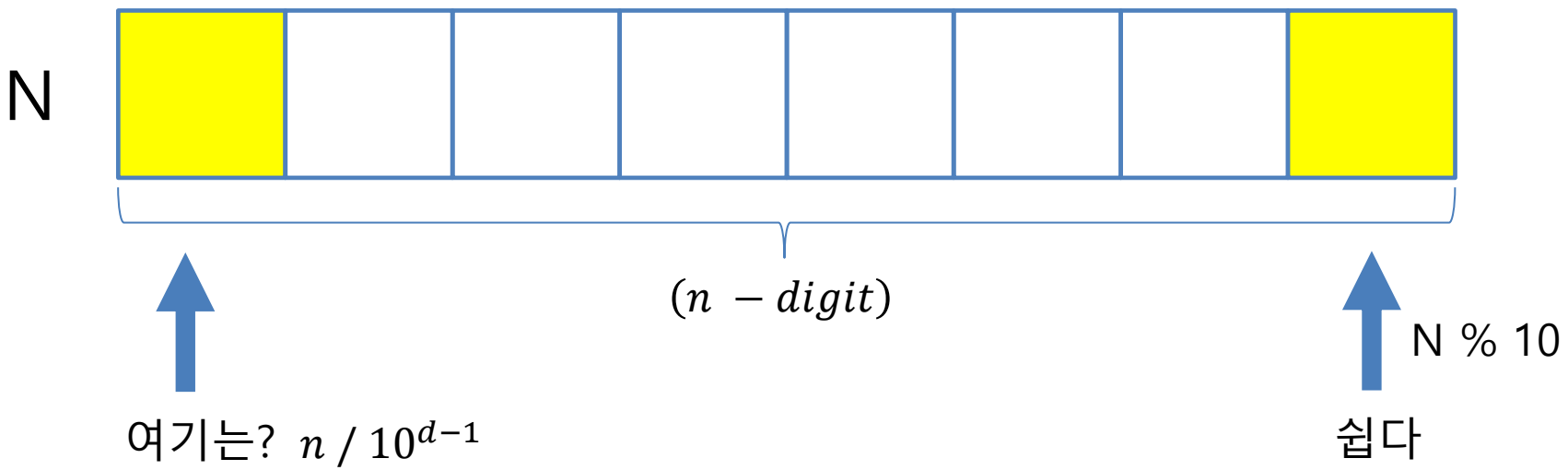
4.9 회문 확인하기(2/2)

- 다른 방법은 없을까??



4.9 회문 확인하기(2/2)

- 맨 앞과, 맨 뒷자리에 있는 숫자 구하기




```
def getDigit1(n: int) -> int:
    return len(str(n))

def getDigit2(n: int) -> int:
    d = 0
    while n > 0:
        n //= 10
        d += 1
    return d
```

시간 복잡도 $O(n)$
공간 복잡도 $O(n)$
회문 판별에 $O(n)$

시간 복잡도 $O(n)$
공간 복잡도 $O(1)$
회문 판별에 $O(n)$

4.9 회문 확인하기(2/2)

- 맨 앞자리 숫자를, 수학적 방법으로 구해보자.
  - ✓ $d = floor(\log_{10} N) + 1 : O(1)$
  - ✓ log 의 성질: $N > 0$ 제약이 생긴다.

```
def isPalindrome(n: int) -> bool:
    if n < 0:
        return False
    if n == 0:
        return True
    d = math.floor(math.log(n, 10)) + 1
    right = n % 10
    left = n // (10**(d - 1))
    while d > 1:
        if left != right:
            return False
        n -= left * (10**(d - 1))
        n //= 10
        d -= 2
        left = n // 10**(d - 1)
        right = n % 10
    return True
```

5.3 주식 사고팔기

6.6 BUY AND SELL A STOCK ONCE

This problem is concerned with the problem of optimally buying and selling a stock once, as described on Page 2. As an example, consider the following sequence of stock prices: ⟨310, 315, 275, 295, 260, 270, 290, 230, 255, 250⟩. The maximum profit that can be made with one buy and one sell is 30—buy at 260 and sell at 290. Note that 260 is not the lowest price, nor 290 the highest price.

Write a program that takes an array denoting the daily stock price, and returns the maximum profit that could be made by buying and then selling one share of that stock.

Hint: Identifying the minimum and maximum is not enough since the minimum may appear after the maximum height. Focus on valid differences.

5.3 주식 사고팔기(1/3)

- 무식하게 풀어보기

```
def maxProfit(L:List[int])->int:
    max_profit = 0
    for i in range(len(L)):
        for j in range(i + 1, len(L)):
            max_profit = max(max_profit, L[j] - L[i])
    return max_profit
```

- 시간 복잡도 $O(n^2)$

5.3 주식 사고팔기(2/3)

- 더 좋은 방법은 없을까..?
- 간단한 방법으로 시간 복잡도 $O(n)$ 가능하다!
- 최저가를 저장해두면 손쉽게 해결 가능

```python
def maxProfit(self, L: List[int]) -> int:
    min_list = [L[0]] * len(L) # 최저가를 저장해보자
    min_price = L[0]
    for i in range(1, len(L)):
        min_price = min(min_price, L[i])
        min_list[i] = min_price
    max_profit = 0
    for i in range(len(L)):
        max_profit = max(max_profit, L[i] - min_list[i])
    return max_profit
```

- 그러나 공간 복잡도가 $O(n)$ .. 더 낮출수 없을까?

5.3 주식 사고팔기(3/3)

- min_price 배열의 특징
  - ✓더 작은 price 가 나오기 전까지는
  - ✓계속 똑같은 최솟값만 저장된다는 점을 이용
  - ✓ 공간 복잡도를 줄일 수 있다!

```
def maxProfit(self, L: List[int]) -> int:
    INT_MAX = 10987654321
    max_profit = 0
    min_price = INT_MAX
    for i in range(len(L)):
        max_profit = max(max_profit, L[i] - min_price)
        min_price = min(min_price, L[i])
    return max_profit
```

- 시간 복잡도 $O(n)$
- 공간 복잡도 $O(1)$

Problems

- **Chapter 4: Primitive types**
  - ✓문제 4.8 – 숫자 뒤집기
    - LeetCode #7
  - ✓문제 4.9 – 회문 확인하기
    - LeetCode #9
    - LeetCode #125

- **Chapter 5: Arrays**
  - ✓문제 5.3 – 주식 사고팔기
    - LeetCode #121

Q & A