1. Greedy(그리디) Algorithm

정보보호학과 17학번 이혜민

INDEX

1. 당장 좋은 것만 선택하는 ‘greedy’

- 그리디는 가장 단순하지만 강력한 문제 해결 방법
- 우리나라에서는 탐욕법(?)으로도 소개되고 있음
- 현재 상황에서 지금 당장 좋은 것만 고르는 방법을 의미
- 현재의 선택이 나중에 미칠 영향 따윈 고려하지 않아 !!!

즉, 그리디 알고리즘을 풀기 위해서는 창의력, 최소한의 아이디어 능력을 요구한다.

1. 당장 좋은 것만 선택하는 ‘greedy’

예제 1. 거스름돈

당신은 음식점의 계산을 도와주는 점원이다.
카운터에는 거스름돈으로 사용할 **500원, 100원, 50원, 10원짜리** 동전이 무한히 존재한다.

손님에게 거슬러 줘야 할 돈이 N원 -> 거슬러 줘야 할 동전의 최소 개수는?
(단, N은 항상 10의 배수)

접근 방법

1. ‘가장 큰 화폐 단위부터’ 돈을 거슬러 주기
2. 500원, 100원, 50원, 10원 순서대로 거슬러 최소의 동전 개수로 계산하기

1. 당장 좋은 것만 선택하는 ‘greedy’

1번 예제. 거스름돈 개수 구하기

```
# n이 1260일 경우?

n = 1260
count = 0

# 큰 단위 화폐부터 차례대로 확인
coin_types = [500, 100, 50, 10]

for coin in coin_types :
  count += n // coin # 해당 화폐로 거슬러 줄 수 있는 동전의 개수 세기
  n %= coin # n = n % coin

print(count)
```

```
6
```

예제 1. 거스름돈

당신은 음식점의 계산을 도와주는 점원이다.
카운터에는 거스름돈으로 사용할 **500원, 100원, 50원, 10원짜리** 동전이 무한히 존재한다.

손님에게 거슬러 줘야 할 돈이 N원 -> 거슬러 줘야 할 동전의 최소 개수는?
(단, N은 항상 10의 배수)

만약 N = 1,260 이라면..?

(코드 링크 : https://colab.research.google.com/drive/101lXtRDi6xDmPnT3ojpf_pUsvAT1t5aD?usp=sharing)

2. 그리디 알고리즘의 정당성

- 그리디 알고리즘으로 정답을 찾은 경우, 해법이 정당한지 검토하는 과정이 중요함
- 앞의 동전 문제에서 해법이 정당한지 쉽게 검토가 가능한 이유는?

  -> 가지고 있는 동전 중에서 큰 단위가 항상 작은 단위의 배수라, 다른 해가 나올 수 없음
- 만약 문제를 처음 만나게 된다면 어떤 사고방식이 필요할까?

  -> 차근차근 작은 수 부터 생각하면서, 방법을 거슬러 올라가서 생각하는 방식이 필요함 !!

3. 큰 수의 법칙

‘큰 수의 법칙’

- 다양한 수로 이루어진 배열에서, 주어진 수들을 M번 더하여 가장 큰 수 만들기

1. 배열의 특정한 인덱스(번호)에 해당하는 수가 연속해서 K번을 초과해서 더해질 수 없음

2. **만약 2,4,5,4,6의 배열**인 경우라면?
   - M이 8이고, K가 3일 때, 아래와 같은 결과가 나옴
   - 6+6+6+5+6+6+6+5 = 46

3. 만약 3,4,3,4,3의 배열인 경우라면?
   - 같은 수이지만, 다른 인덱스라면 -> 서로 다른 것으로 간주함
   - M이 7이고, K가 2일 때, 다른 수라도 계속 더해질 수 있음
   - 4+4+4+4+4+4+4 = 28

3. 큰 수의 법칙

입력 예시	출력 예시
5 8 3 2 4 5 4 6	46

‘큰 수의 법칙’

- 다양한 수로 이루어진 배열에서, 주어진 수들을 M번 더하여 가장 큰 수 만들기
- 배열의 특정한 인덱스(번호)에 해당하는 수가 연속해서 K번을 초과해서 더해질 수 없음

입력 조건

- 첫째 줄에 N, M, K의 자연수가 주어지며, 각 자연수는 공백으로 구분된다.
- 둘째 줄에 N개의 자연수가 주어진다. 각 자연수는 공백으로 구분된다.
  각, 자연수는 1이상 10,000 이하의 수로 주어진다.

출력 조건

- 첫째 줄에 동빈이의 큰 수의 법칙에 따라 더해진 답을 출력한다.

3. 큰 수의 법칙

3번 예제. 큰 수의 법칙

```
# N, M, K를 공백으로 받기
n,m,k = map(int, input().split())
# N개의 수를 공백으로 구분하여 입력받기
data = list(map(int, input().split()))


data.sort() #입력받은 수 정렬하기
first = data[n-1] #가장 큰 수
second = data[n-2]

result = 0

while True :
  for i in range(k): # 가장 큰 수를 k번 더하기
    if m == 0:
      break
    result += first
    m -= 1 # 더할 때 마다 1씩 빼기
  if m == 0: # m이 0이라면 반복문 탈출
    break
  result += second # 두번재로 큰 수를 한 번 더하기
  m -= 1 # 더할 때마다 1씩 빼기

print(result) #최종 답안 출력
```

```
3 8 2
3 9 7
68
```

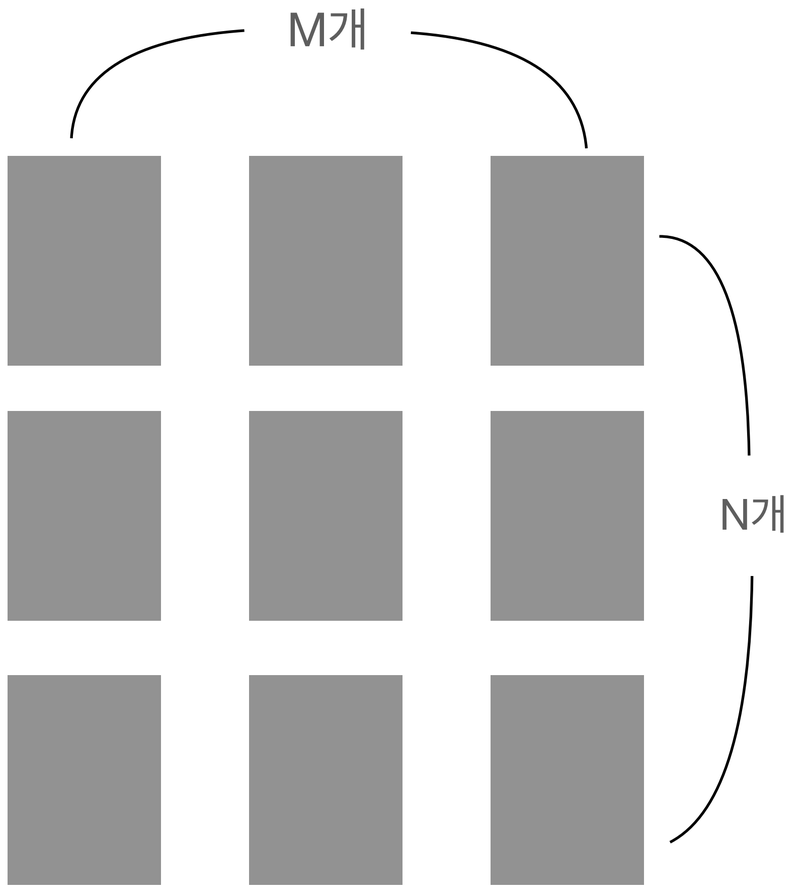
‘큰 수의 법칙’

- 다양한 수로 이루어진 배열에서, 주어진 수들을 M번 더하여 가장 큰 수 만들기
- 배열의 특정한 인덱스(번호)에 해당하는 수가 연속해서 K번을 초과해서 더해질 수 없음

문제 접근 방법

- 가장 큰 수와, 가장 두 번째로 큰 수만 저장하면 된다.
- 연속으로 더할 수 있는 횟수는 최대 K번
- **가장 큰 수를 K번 더하고, 두번째로 큰 수를 1번 더하는 연산을 반복**하면 쉽게 해결!

4. 숫자 카드 게임



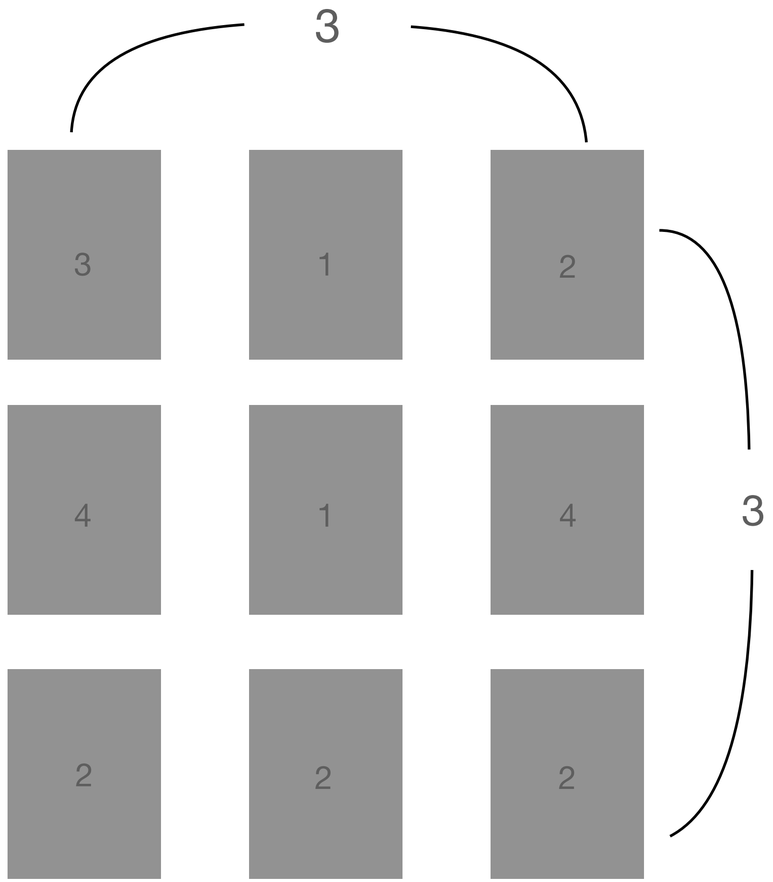
숫자 카드 게임은?
- 여러 개의 숫자 카드 중, 가장 높은 숫자가 쓰인 카드 한 장을 뽑는 게임

1. 숫자가 쓰인 카드들이 N X M 형태로 놓여 있다.
2. N은 행의 개수를 의미, M은 열의 개수를 의미 한다.
3. 먼저 뽑고자 하는 카드가 포함되어 있는 행을 선택한다.
4. 그 다음 선택된 행에 포함된 카드들 중 가장 숫자가 낮은 카드를 뽑아야 한다.

접근 방법

처음에 카드를 골라낼 행을 선택할 때,
해당 행에서 **가장 숫자가 낮은 카드를 뽑을 것**을 고려하여 최종적으로 가**장 높은 숫자의 카드를 뽑을** 수 있도록 전략을 세워야 함

3. 숫자 카드 게임



숫자 카드 게임은?

- 만약 3 x 3 형태로 카드가 놓여있다고 가정해보자

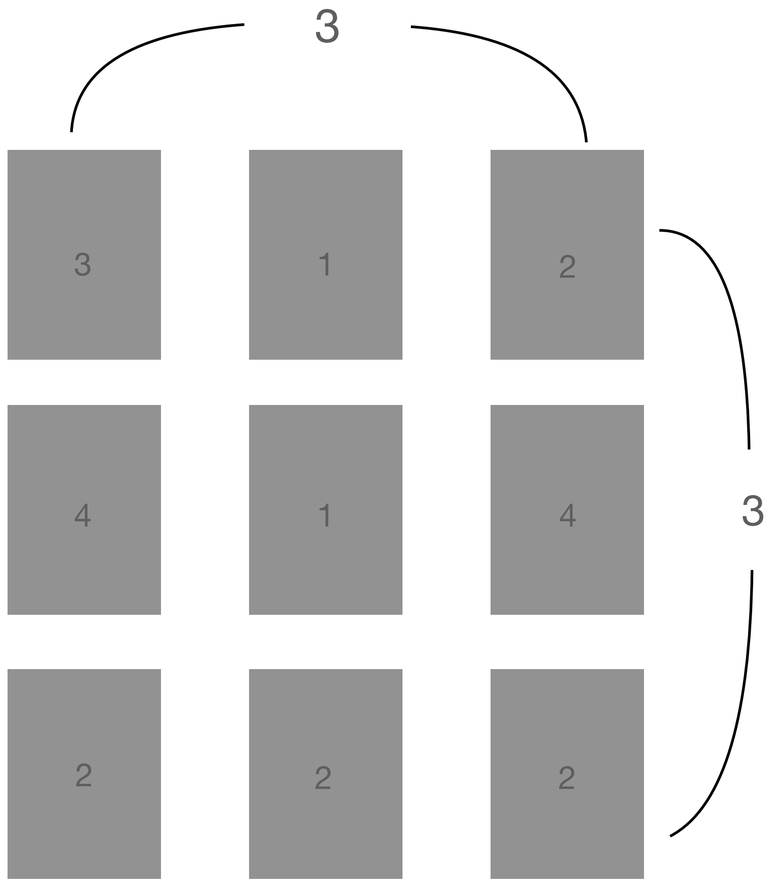
입력 조건

- 첫째 줄에 숫자 카드들이 놓인 행의 개수 N과, 열의 개수 M이 공백을 기준으로 하여 각각 자연수로 주어진다.
- 둘째 줄부터 N개의 줄에 걸쳐 각 카드에 적힌 숫자가 주어진다.
- 숫자는 1이상, 10,000 이하의 자연수이다.

출력 조건

- 첫째 줄에 게임의 룰에 맞게 선택한 카드에 적힌 숫자를 출력한다.

4. 숫자 카드 게임



풀이 접근 방법

- 입력 값이 모두 10,000보다 작기 때문에 단순하게 배열로 접근 가능
- 가장 작은 수를 찾는 기본 문법을 이용해서 풀기
- 각 행마다 가장 작은 수를 찾고 -> 찾은 수 중에서 가장 큰 수 찾기

사용할 함수

- min( ) 함수
- 2중 반복문 사용

4. 숫자 카드 게임

2번 예제 - 1. min() 함수를 이용하는 답안

```python
# N, M을 공백으로 구분하여 입력받기

n, m = map(int, input().split())

result = 0

# 한 줄씩 입력받아 확인

for i in range(n):
  data = list(map(int, input().split())) #한 줄 입력받아 list로 배열 만들기
  # 현재 줄에서 '가장 작은 수' 찾기
  min_value = min(data)
  # '가장 작은 수'들 중에서 가장 큰 수 찾기
  # 만약 min_value값이 더 크다면 result값으로 지정 / 기존 result값이 더 크다면 그대로 result값 지정
  result = max(result, min_value)

print(result)
```

```
2 2
1 2
2 4
2
```

2번 예제 - 2. 이중 반복문 구조 이용하는 답안

```python
# N, M을 공백으로 구분하여 입력받기

n, m = map(int, input().split())

result = 0

# 한 줄씩 입력받아 확인

for i in range(n):
  data = list(map(int, input().split())) #한 줄 입력받아 list로 배열 만들기
  # 현재 줄에서 '가장 작은 수' 찾기
  min_value = 10001
  for a in data :
    min_value = min(min_value, a)

  # '가장 작은 수'들 중에서 가장 큰 수 찾기
  result = max(result, min_value)

print(result) # 최종 답안 출력
```

```
3 3
1 3 1
2 5 3
4 5 9
4
```

https://colab.research.google.com/drive/101lXtRDi6xDmPnT3ojpf_pUsvAT1t5aD?usp=sharing

5. 1이 될 때 까지

입력 예시	출력 예시
25 5	2

어떠한 수 N이 1이 될 때까지 두 과정 중 하나를 반복적으로 선택해서 수행한다.
- 2번째 과정은 N이 K로 나누어떨어질 때만 선택할 수 있다.

1. N에서 1을 뺀다.
2. N을 K로 나눈다.

입력 조건

- 첫째 줄에 N과 K가 공백으로 구분되며 각각 자연수로 주어진다.
  이때, 입력으로 주어지는 N은 항상 K보다 크거나 같다.

출력 조건

- N과 K가 주어질 때, 1번 혹은 2번을 수행해야 하는 총 최솟값, 횟수를 구해라.

5. 1이 될 때 까지

5번 예제. 1이 될 때까지

```python
# n, k을 공백으로 구분하여 입력받기

n, k = map(int, input().split())
result = 0

while True:
  # (N == K로 나누어떨어지는 수)가 될 때까지 1씩 빼기
  target = (n//k) * k  # 6, 4 -> target = 4
  result += (n-target) # 6 - 4 = 2 / 1빼기가 2번 수행되었다고 볼 수 있음
  n = target

  # N이 K보다 작을 때 (더 이상 나눌 수 없을 때) 반복문 탈출
  if n < k:
    break
  result += 1
  n //= k

# 마지막으로 남은 수에 대해서 1씩 빼기
result += (n-1)
print(result)
```

```
3 15
2
```

어떠한 수 N이 1이 될 때까지 두 과정 중 하나를 반복적으로 선택해서 수행한다.
- 2번째 과정은 N이 K로 나누어떨어질 때만 선택할 수 있다.

1. N에서 1을 뺀다.
2. N을 K로 나눈다.

풀이 전략

- 최대한 많이 나누기가 중요 : 실행되는 숫자가 훨씬 적기 때문

- 문제에서는 N의 범위가 10만 이하, N이 커질수록 나누기 전력이 더 유리 !
- N이 K의 배수가 되도록 한번에 마이너스 하는 과정도 중요 !

Thank You