그리디1

그리디

그리디(Greedy) : 현재 상황에서 지금 당장 좋은 것만 고르는 방법, 탐욕법 이라고도 함.

그리디 알고리즘을 이용하면 매 순간 가장 좋아 보이는 것을 선택하며, 현재의 선택이 나중에 미칠 영향에 대해서는 고려하지 않음.

그리디 알고리즘은 기준에 따라 좋은 것을 선택하는 알고리즘이므로 문제에서 '가장 큰 순서대로', '가장 작은 순서대로'와 같은 기준을 알게 모르게 제시해준다. 그리디 알고리즘 문제는 자주 정렬 알고리즘과 짝을 이뤄 출제됨.

그리디 알고리즘의 정당성

대부분의 문제는 그리디 알고리즘을 이용했을 때 '최적의 해'를 찾을 수 없을 가능성이 다분함. 거스름돈 문제와 같이 '가장 큰 화폐단위부터' 돈을 거슬러 주는 것과 같이, 탐욕적으로 문제에 접근했을 때 정확한 답을 찾을 수 있다는 보장이 있을 때는 매우 효과적이고 직관적임.

그리디 알고리즘으로 문제의 해법을 찾았을 때는 그 해법이 정당한지 검토해야 함. 거스름돈 문제를 그리디 알고리즘으로 해결할 수 있는 이유는 가지고 있는 동전 중에서 큰 단위가 항상 작은 단위의 배수이므로 작은 단위의 동전들을 종합해 다른 해가 나올 수 없기 때문.

대부분의 그리디 알고리즘 문제에서는 이처럼 문제 풀이를 위한 최소한의 아이디어를 떠올리고 이것이 정당한지 검토할 수 있어야 답을 도출할 수 있음.

+ 거스름돈 문제에서 동전의 단위가 서로 배수 형태가 아니라, 무작위로 주어진 경우에는 그리디 알고리즘으로 해결할 수 없으며 DP로 해결할 수 있음.

예시문제

거스름돈

[문 제]

당신은 음식점의 계산을 도와주는 점원이다. 카운터에는 거스름돈으로 사용할 500원, 100원, 50원, 10원짜리 동전이 무한히 존재한다고 가정한다. 손님에게 거슬러 줘야 할 돈이 N원일 때 거슬러 줘야 할 동전의 최소 개수를 구하라. 단, 거슬러 줘야 할 돈 N은 항상 10의 배수이다.

[문제 해설]

중요한 아이디어 : '가장 큰 화폐 단위부터' 돈을 거슬러 주는 것

500원으로 거슬러 줄 수 있을 만큼 거슬러 준 후에, 100원, 50원, 10원 동전을 차례대로 거슬러 줄 수 있을 만큼 거슬러 주면 된다.

ex. 거스름돈이 1260원인 경우

거스름돈이 1260원일 때, 손님이 받은 동전의 최소 개수는 6개.

[예시 코드]

```
n = 1260
count = 0

# 큰 단위의 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

print(count)
```

큰 수의 법칙

[문 제]

큰 수의 법칙은 다양한 수로 이루어진 배열이 있을 때 주어진 수들을 M번 더하여 가장 큰 수를 만드는 법칙이다. 단, 배열의 특정한 인덱스에 해당하는 수가 연속해서 K번을 초과하여 더해질 수 없는 것이 이 법칙의 특징이다.

예를 들어 순서대로 2, 4, 5, 4, 6 으로 이루어진 배열이 있을 때 M이 8이고, K가 3이라고 가정하자. 이 경우 특정한 인덱스의 수가 연속해서 세 번까지만 더해질 수 있으므로 큰 수의 법칙에 따른 결과는 6 + 6 + 6 + 5 + 6 + 6 + 6 + 5 인 46이 된다.

단, 서로 다른 인덱스에 해당하는 수가 같은 경우에도 서로 다른 것으로 간주한다. 예를 들어 순서대로 3, 4, 3, 4, 3 으로 이루어진 배열이 있을 때 M이 7이고, K가 2라고 가정하자. 이 경우 두 번 째 원소에 해당하는 4와 네 번째 원소에 해당하는 4를 번갈아 두 번씩 더하는 것이 가능하다. 결과적으로 4 + 4 + 4 + 4 + 4 + 4 + 4 인 28이 도출된다.

배열의 크기 N, 숫자가 더해지는 횟수 M, 그리고 K가 주어질 때 큰 수의 법칙에 따른 결과를 출력하시오.

자연수 N, M, K가 주어진다.
둘째 줄에 N개의 자연수가 주어지고 각 자연수는 공백으로 구분한다.
입력으로 주어지는 K는 항상 M보다 작거나 같다.

[문제 해설 1]

중요한 아이디어 : 가장 큰 수를 K번 더하고 두 번째로 큰 수를 한 번 더하는 연산을 반복

반복되는 수열에 대해서 파악해야함.

[예시 답안]

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

# N개의 수를 공백으로 구분하여 입력받기
data = list(map(int, input().split()))

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

result = 0

while True:
  for i in range(K):  # 가장 큰 수를 K번 더하기
    if m == 0:  # m이 0이라면 반복문 탈출
      break
    result += first
    m -= 1  # 더할 때마다 1씩 빼기
  if m == 0:  # m이 0이라면 반복문 탈출
    break
  result += second  # 두 번째로 큰 수를 한 번 더하기
  m -= 1  # 더할 때마다 1씩 빼기

print(result)  # 최종 답안 출력
```

[문제 해설 2]

반복되는 수열에 대해서 파악

가장 큰 수가 더해지는 횟수 구하기.

M을 (K + 1) 로 나눈 몫이 수열이 반복되는 횟수. 나누어 떨어지지 않는 경우도 고려.

⇒ (M//(K + 1)) * K + M % (K + 1)

M - 가장 큰 수가 더해지는 횟수 = 두번째로 큰 수가 더해지는 횟수

[예시 답안]

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

# N개의 수를 공백으로 구분하여 입력받기
data = list(map(int, input().split()))

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

# 가장 큰 수가 더해지는 횟수 계산
count = int(m / (k + 1)) * k
count += m % (k + 1)

result = 0
result += (count) * first  # 가장 큰 수 더하기
result += (m - count) * second  # 두 번째로 큰 수 더하기


print(result)  # 최종 답안 출력
```

공통문제

A와 B 골드5 - https://www.acmicpc.net/problem/12904

센서 골드5 - https://www.acmicpc.net/problem/2212

개인문제

~~공항 골드2 - https://www.acmicpc.net/problem/10775~~

⇒ 분리집합 Union-Find 사용 하는 문제

그 외 : https://www.acmicpc.net/problemset?sort=ac_desc&algo=33