〈문제〉 거스름 돈: 문제 설명

- 당신은 음식점의 계산을 도와주는 점원입니다. 카운터에는 거스름돈으로 사용할 500원, 100원, 50원, 10원짜리 동전이 무한히 존재한다고 가정합니다. 손님에게 거슬러 주어야 할 돈이 N원일 때 거슬러 주어야 할 동전의 최소 개수를 구하세요. 단, 거슬러 줘야 할 돈 N은 항상 10의 배수입니다.

대표적인 그리디 알고리즘 문제로 거스름 돈 문제를 함께 풀어보겠습니다.

〈문제〉 거스름 돈: 문제 해결 아이디어

- 최적의 해를 빠르게 구하기 위해서는 **가장 큰 화폐 단위부터** 돈을 거슬러 주면 됩니다.
- N원을 거슬러 줘야 할 때, 가장 먼저 500원으로 거슬러 줄 수 있을 만큼 거슬러 줍니다.
  - 이후에 100원, 50원, 10원짜리 동전을 차례대로 거슬러 줄 수 있을 만큼 거슬러 주면 됩니다.
- N = 1,260일 때의 예시를 확인해 봅시다.

거스름 돈 문제의 해결 아이디어는 간단합니다.

거스름 돈 문제는 그리디 알고리즘을 설명하기 위해

자주 등장하는 문제 예시로서 문제 해결 아이디어는 위와 같습니다.

〈문제〉 거스름 돈: 문제 해결 아이디어

- **[Step 0]** 초기 단계 – 남은 돈: 1,260원

(아무것도 없는 상태)

화폐 단위	500	100	50	10
손님이 받은 개수	0	0	0	0

- 본래 점원이 거슬러 줘야 할 돈은 보이지 않고 잔돈 무더기가 보여야 논리적으로 맞습니다. 이해를 돕고자 미리 거스름돈을 시각적으로 표현했습니다.

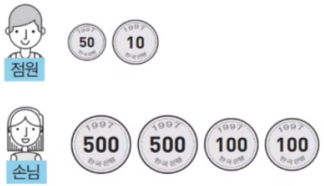
〈문제〉 거스름 돈: 문제 해결 아이디어

- **[Step 1]** 남은 돈: 260원

화폐 단위	500	100	50	10
손님이 받은 개수	2	0	0	0

〈문제〉 거스름 돈: 문제 해결 아이디어

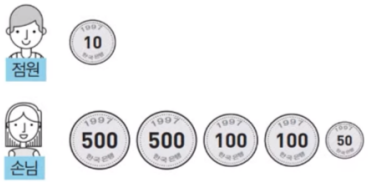
- **[Step 2]** 남은 돈: 60원



화폐 단위	500	100	50	10
손님이 받은 개수	2	2	0	0

〈문제〉 거스름 돈: 문제 해결 아이디어

- **[Step 3]** 남은 돈: 10원



화폐 단위	500	100	50	10
손님이 받은 개수	2	2	1	0

〈문제〉 거스름 돈: 문제 해결 아이디어

- **[Step 4]** 남은 돈: 0원

화폐 단위	500	100	50	10
손님이 받은 개수	2	2	1	1

결과적으로 이렇게 총 6개의 동전으로 1,260원을 거슬러 줄 수 있었습니다.

그렇다면 이렇게 가장 큰 화폐부터 거슬러 주는 것으로 최적의 해를 보장할 수 있는 이유는 무엇일까요?

정당성 분석이 중요합니다.

〈문제〉 거스름 돈: 정당성 분석

- 가장 큰 화폐 단위부터 돈을 거슬러 주는 것이 최적의 해를 보장하는 이유는 무엇일까요?
  - 가지고 있는 동전 중에서 **큰 단위가 항상 작은 단위의 배수이므로 작은 단위의 동전들을 종합해 다른 해가 나올 수 없기 때문**입니다.
- 만약에 800원을 거슬러 주어야 하는데 화폐 단위가 500원, 400원, 100원이라면 어떻게 될까요?
- 그리디 알고리즘 문제에서는 이처럼 문제 풀이를 위한 최소한의 아이디어를 떠올리고 이것이 정당한지 검토할 수 있어야 합니다.

만약 큰 단위가 작은 단위의 배수가 아니라면

위와 같은 알고리즘을 이용해서 최적의 해를 보장할 수 없는 것입니다.

〈문제〉 거스름 돈: 답안 예시 (Python)

```python
n = 1260
count = 0

# 큰 단위의 화폐부터 차례대로 확인하기
array = [500, 100, 50, 10]

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

print(count)
```



```python
n = 1260
count = 0

#큰 단위의 화폐부터 작은단위 화폐까지 리스트에 담아주기
array = [500, 100, 50, 10]

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

print(count)
```


Python 으로 해결할 때는 위와 같이 코드를 작성할 수 있습니다.

〈문제〉 거스름 돈: 시간 복잡도 분석

- 화폐의 종류가 K라고 할 때, 소스코드의 시간 복잡도는 $O(K)$입니다.
- 이 알고리즘의 시간 복잡도는 거슬러줘야 하는 금액과는 무관하며, 동전의 총 종류에만 영향을 받습니다.

이제 거스름 돈 문제의 풀이 방법을 분석해보겠습니다.

화폐의 종류가 K 라고 할 때 화폐의 종류만큼만 반복을 수행하면 답을 도출할 수 있다는거죠.

〈문제〉 거스름 돈: 답안 예시 (C++)

```cpp
#include <bits/stdc++.h>

using namespace std;

int n = 1260;
int cnt;

int coinTypes[4] = {500, 100, 50, 10};

int main(void) {
    for (int i = 0; i < 4; i++) {
        cnt += n / coinTypes[i];
        n %= coinTypes[i];
    }
    cout << cnt << '\n';
}
```

```cpp
#include <bits/stdc++.h>  # C++ 에서 표준라이브러리를 불러오기 위한 헤더파일

using namespace std;

int n = 1260;
int cnt;

int coinTypes[4] = {500, 100, 50, 10};

int main(void) {
    for (int i = 0; i < 4; i++) {
        cnt += n / coinTypes[i];
        n %= coinTypes[i];
    }
    cout << cnt << '\n';
}
```

C++ 에서의 코드 구현

〈문제〉 거스름 돈: 답안 예시 (Java)

```java
public class Main {

    public static void main(String[] args) {
        int n = 1260;
        int cnt = 0;
        int[] coinTypes = {500, 100, 50, 10};

        for (int i = 0; i < 4; i++) {
            cnt += n / coinTypes[i];
            n %= coinTypes[i];
        }

        System.out.println(cnt);
    }
}
```

그리디 알고리즘 나동빈

```java
public class Main {

    public static void main(String[] args) {
        int n = 1260;
        int cnt = 0;
        int[] coinTypes = {500, 100, 50, 10};

        for (int i = 0; i < 4; i++) {
            cnt += n / coinTypes[i];
            n %= coinTypes[i];
        }

        System.out.println(cnt);
    }
}
```

JAVA 에서의 코드 구현

JAVA 소스 코드를 제출할 때는 class 이름을 Main 이라고 요구하는 경우가 많습니다.