Algorithm

동양미래대학교 강은영

프로그램 개발 방식 설문

- 본인이 가장 선호하는 언어는?
- 가장 자신 있는 언어는?

언어	C	Java	Python	C++	JavaScript	C#	Kotlin	Nodejs	HTML, CSS	PHP	react
인원	28	44	40	5	28	19	7	4	1	1	1

- 소스코드 관리
  - github, sourcetree

동전 0

(출처) https://www.acmicpc.net/problem/11047

준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.

동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 한다. 이때 필요한 동전 개수의 최솟값을 구하는 프로그램을 작성하시오.

입력	첫째 줄에 N과 K가 주어진다. ($1 \le N \le 10, 1 \le K \le 100,000,000$) 둘째 줄부터 N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 1,000,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_{i-1}의 배수)	
출력	첫째 줄에 K원을 만드는데 필요한 동전 개수의 최솟값을 출력한다.	
입출력 예	**입력** 10 4200 1 5 10 50 100 500 1000 5000 10000 50000	**출력** **Output: 6**

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.

그리디(Greedy) 알고리즘

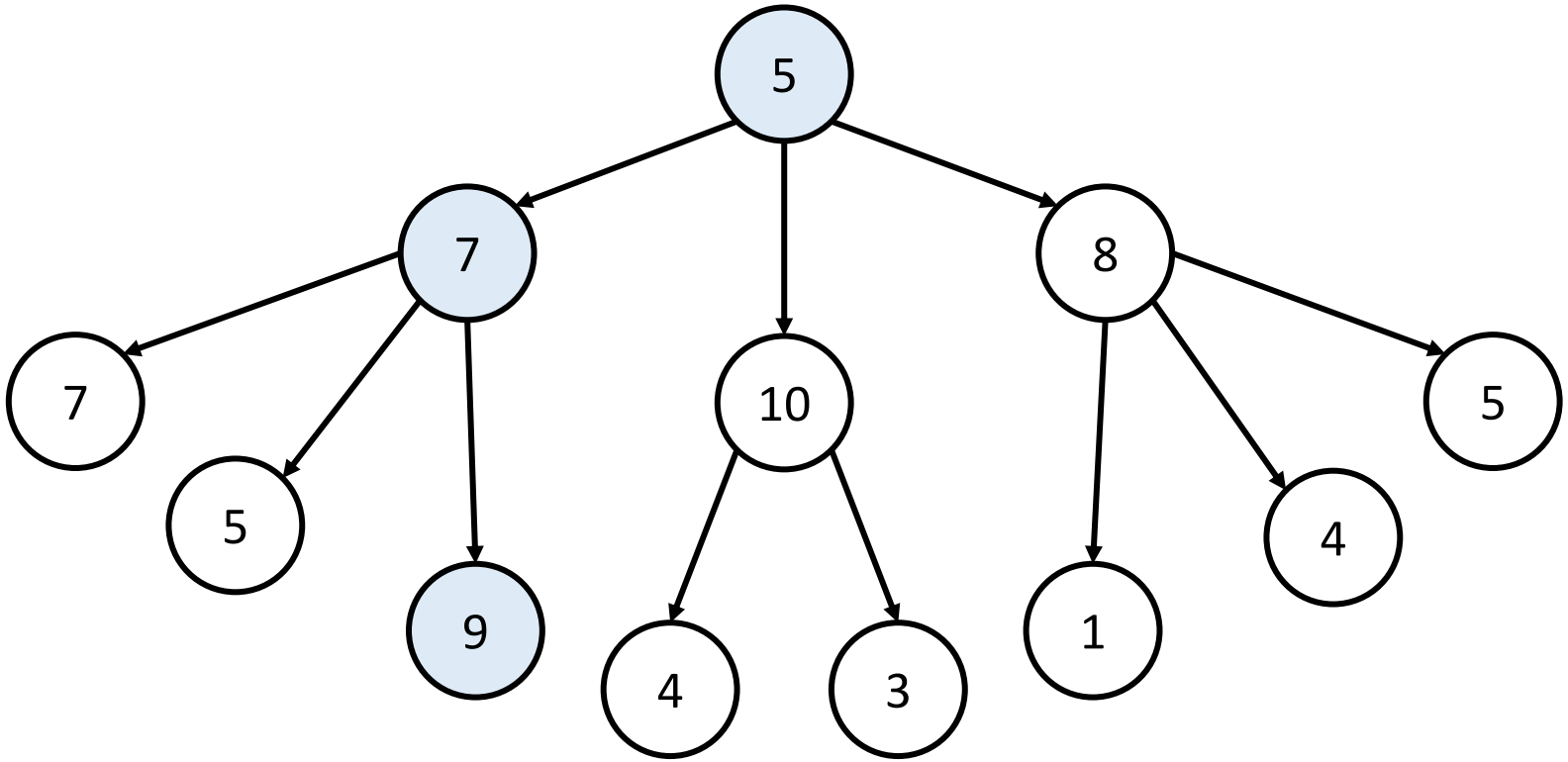
- 그리디 알고리즘(욕심쟁이 알고리즘)은 현재 상황에서 지금 당장 좋은 것만 고르는 방법을 의미합니다.
- 그 때 그 때는 최적일지도 모르지만, 최종적으로는 답이 최적이 아닐수도 있습니다.
- 그리디 해법은 그 정당성 분석이 중요합니다.
  - 단순히 가장 좋아 보이는 것을 반복적으로 선택해도 최적의 해를 구할 수 있는지 검토합니다.

그리디(Greedy) 알고리즘

- 언제 그리디 알고리즘을 쓰나?
- 지금 이 순간 가장 좋은 경우를 선택하는 것이 가장 최적인 경우에
- 그래서 쉽다.
- 가장 어렵다.

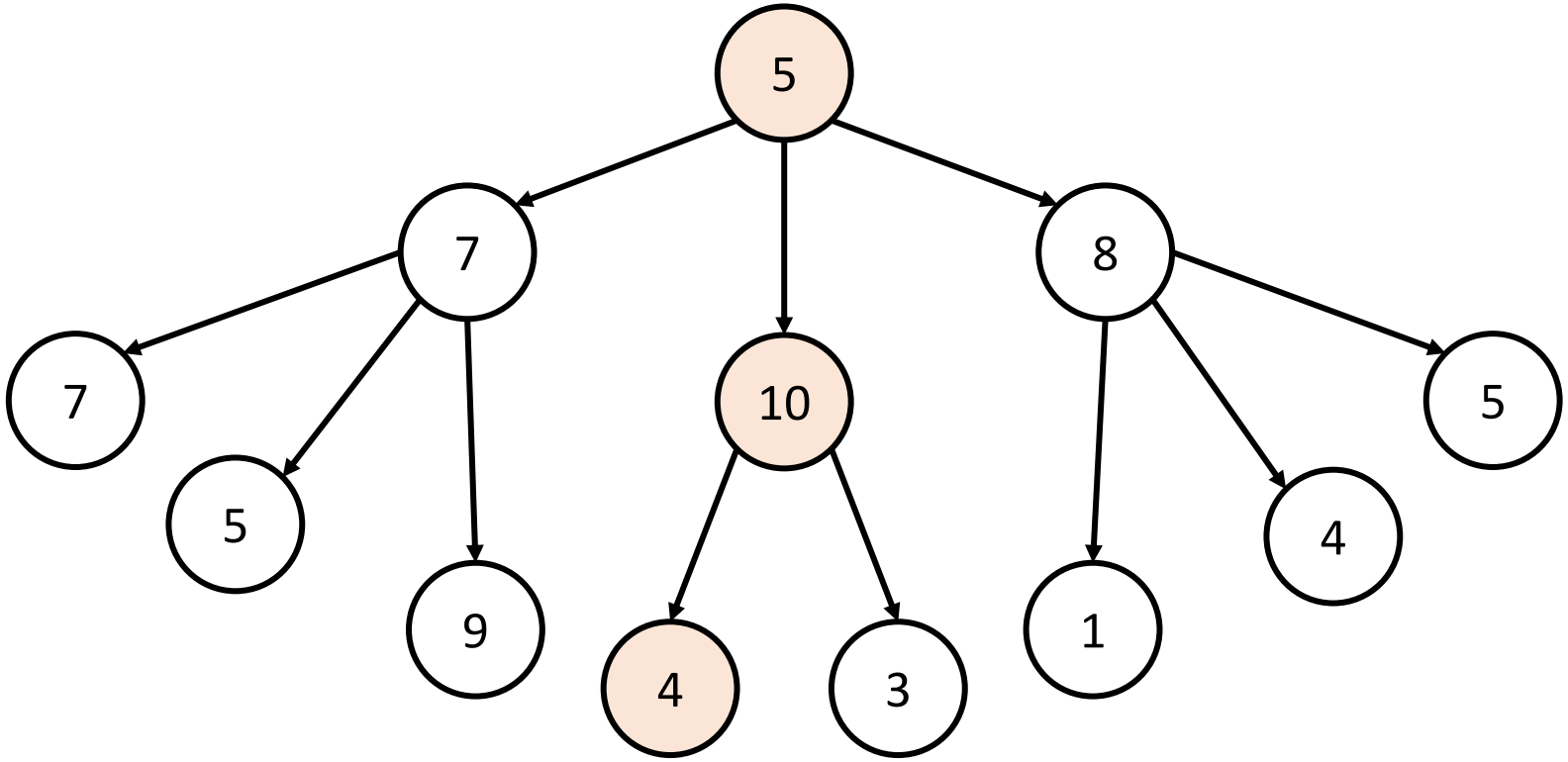
그리디 알고리즘

- [문제 상황] 루트 노드부터 시작하여 거쳐 가는 노드 값의 합을 최대로 만들고 싶습니다.
  - Q. 최적의 해는 무엇인가요?



그리디 알고리즘

- 루트 노드부터 시작하여 거쳐 가는 노드 값의 합을 최대로 만들고 싶습니다.
  - 단순히 매 상황에서 가장 큰 값만 고른다면 어떻게 될까요?



그리디(Greedy) 알고리즘

- 일반적인 상황에서 그리디 알고리즘은 최적의 해를 보장할 수 없을 때가 많음.
- 하지만 코딩 테스트에서의 대부분의 그리디 문제는 탐욕법으로 얻은 해가 최적의 해가 되는 상황에서 이를 추론할 수 있어야 풀 수 있어야 함.

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.

거스름 돈 문제

Think-1

- 가장 큰 화폐 단위부터 돈을 거슬러 주면 됩니다.
- 이 때 사용하는 지폐와 동전의 개수를 최소로 해야 한다.
- 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 문제

- [Step 0] 초기 단계 – 남은 돈: 1,260원

(아무것도 없는 상태)

화폐 단위	500	100	50	10
손님이 받은 개수	0	0	0	0

거스름 돈 문제

- [Step 1] 남은 돈: 260원

화폐 단위	500	100	50	10
손님이 받은 개수	2	0	0	0

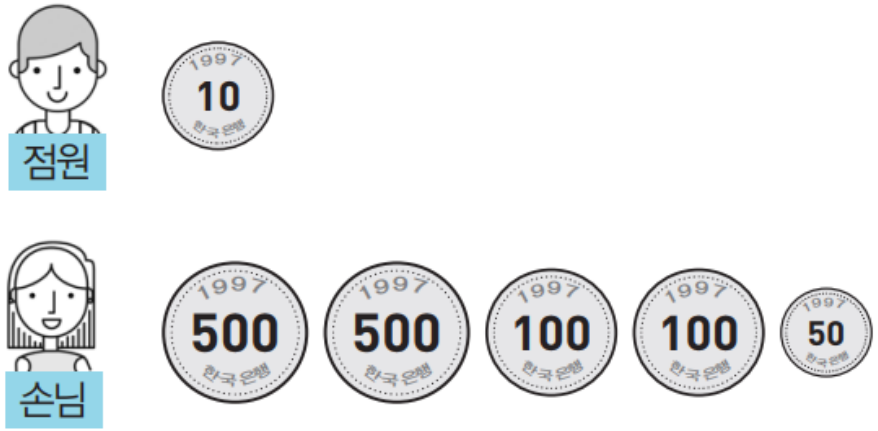
거스름 돈 문제

- [Step 2] 남은 돈: 60원

화폐 단위	500	100	50	10
손님이 받은 개수	2	2	0	0

거스름 돈 문제

- [Step 3] 남은 돈: 10원



화폐 단위	500	100	50	10
손님이 받은 개수	2	2	1	0

거스름 돈 문제

- [Step 4] 남은 돈: 0원

(아무것도 없는 상태)

화폐 단위	500	100	50	10
손님이 받은 개수	2	2	1	1

거스름 돈 문제

- 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 때문입니다.
- 만약에 800원을 거슬러 주어 하는데 화폐 단위가 500원, 400원, 100원이라면 어떻게 될까요?
- 그리디 알고리즘 문제에서는 이처럼 문제 풀이를 위한 최소한의 아이디어를 떠올리고 이것이 정당한지 검토할 수 있어야 합니다.

그리디 알고리즘의 정당성

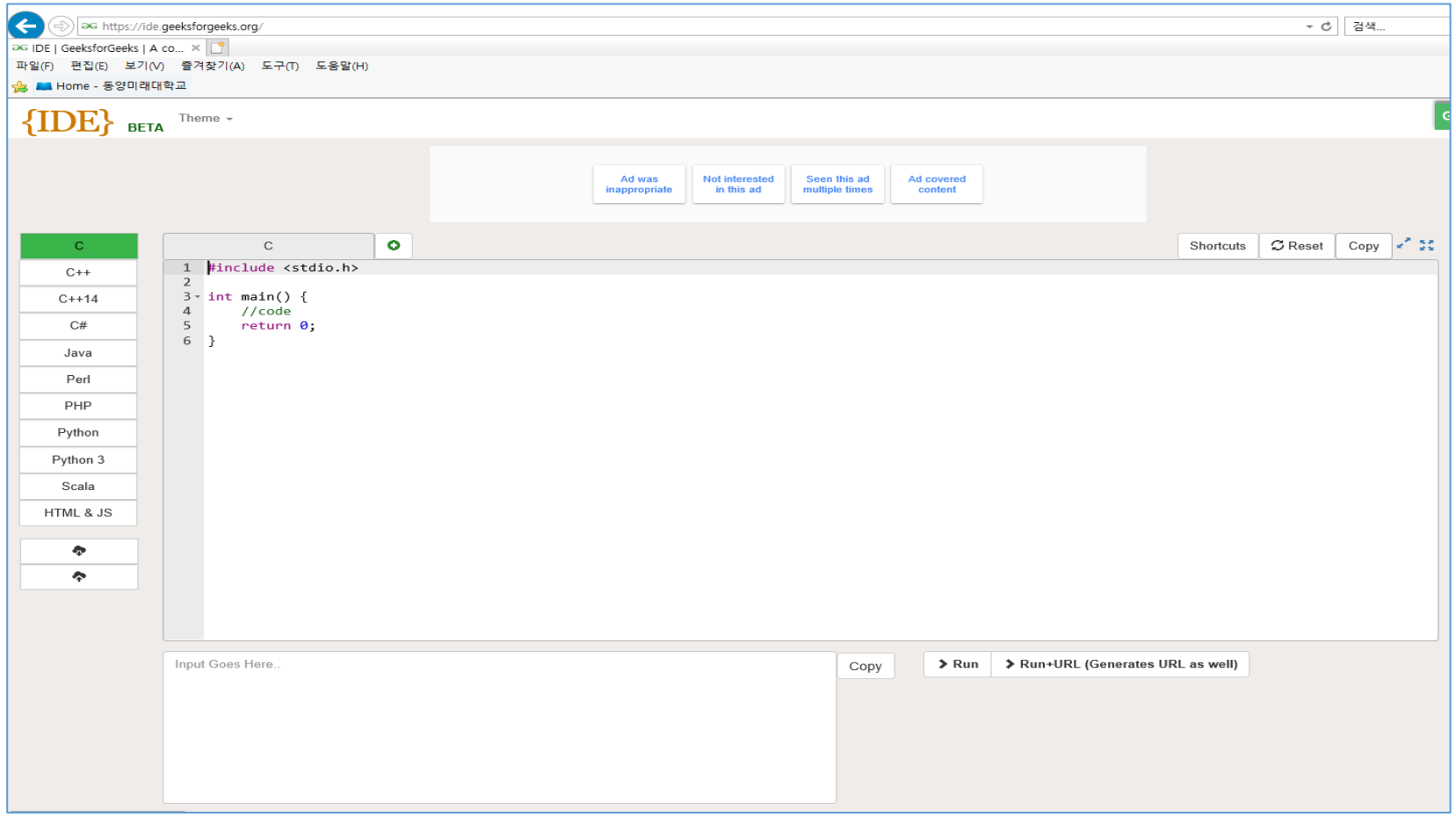
- 가지고 있는 동전 중에서 큰 단위가 항상 작은 단위의 배수이므로 작은 단위의 동전들을 종합해 다른 해가 나올 수 없기 때문에
  - 예) 800원 거슬러 줄 경우
    - (500원*1 + 100원*3=>4)
    - (화폐단위가 500원,400원,100원인경우 400원*2=>2)
  - 큰 단위가 작은 단위의 배수 이므로, 가장 큰 단위의 화폐부터 가장 작은 단위의 화폐까지 차례대로 확인하여 거슬러 주는 작업만을 수행해도 된다.
    - (예 500원 = 1개, 500원/100원 =5개)

그리디 알고리즘의 정당성

- 동전의 종류가 총 N종류,
- 동전의 가치 A_i 가 오름차순으로 주어진다.(1<= A_i <=1000000, A_1=1, A_i는 A_{i-1}의 배수)
- 가치가 A_i(100)인 동전을 A_{i+1}/A_i(500/100=5)개보다 적게 사용한 것이 정답이다.
- A_i를 A_{i+1}/A_i개 사용했다면 1개로 변경하면 더 최소가 된다.
- A_i는 최대 A_{i+1}/A_i개를 사용할 수 있기 때문에, A_i를 사용하지 않고 만들 수 있는 최대 금액은 다음과 같다.
- $(A_2/A_1-1)A_1 + (A_3/A_2-1)A_2 + ... + (A_i/A_{i-1} -1)A_{i-1}$ 예) 1*9 + (10*4) + (50*1) + (100*4)
- => $(A_2-A_1) + (A_3-A_2) + ... + (A_i-A_{i-1})$
- => $A_i-A_1 = A_i -1$
- 따라서, 금액 > =A_i면 A_i가 꼭 포함되어야 한다.

온라인 개발 환경(C, C++, Java, Python)

- https://ide.geeksforgeeks.org/



거스름 돈 문제 (Java)

```java
import java.io.*;

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);
    }
}
```

거스름 돈 문제(C)

```c
#include <stdio.h>
int main (void)
{
    int n = 1260;
    int cnt = 0;
    int coinTypes[4] = {500, 100, 50, 10};
    int size = sizeof(coinTypes)/sizeof(coinTypes[0]);

    for (int i = 0; i < size; i++) {
        cnt += n / coinTypes[i];
        n %= coinTypes[i];
    }
    printf("%d" , cnt);

}
```

거스름 돈 문제 (Python)

```python
n = 1260
cnt = 0

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

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

print(cnt)
```

거스름 돈 문제(C++)

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

using namespace std;

int n = 1260;
int cnt = 0;

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';
}
```

거스름 돈 문제

- 화폐의 종류가 n라고 할 때, 소스코드의 시간 복잡도는 $\boldsymbol{O(n)}$입니다.
- 이 알고리즘의 시간 복잡도는 거슬러줘야 하는 금액과는 무관하며, 동전의 총 종류에만 영향을 받습니다.

소스코드 관리

- 거스름돈 관리 문제를 풀고, 동전0을 풀어봅니다.

- 동전0 소스코드를 JAVA, C, C++, Python 중 2가지를 택하여 작성하고
- 과제(LMS(eclass.dongyang.ac.kr)에 제출해 주세요.