ㅣ

jeon, sihyeon (전시현)

# 과제: HW-3

* Code Test: https://www.acmicpc.net/problem/11047
* Code Test: https://www.acmicpc.net/problem/12865
* 해당 프로그램에 주석 (comment)를 한글로 단다. (추가할 수 있는 것은 최대한 추가)

# #HW 3-1

|  |  |
| --- | --- |
|  | N종류의 동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 할 때 필요한 동전의 개수의 최솟값 |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>

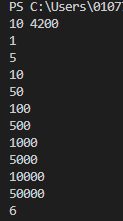
N종류의 동전을 적절히 사용해서 그 가치의 합을 K로 만들려고 할 때 필요한 동전의 개수의 최솟값을 찾아야한다

## SOURCE CODE

|  |
| --- |
| /\*\*\*\*      \*file : 11047\_coins.c      \*      \* Author : Jeon Sihyeon (wjstlgus13@gachon.ac.kr)      \*      \* Date : 10/11/2022      \*      \* Partner : I worked alone      \* Course : Algorithm (14465\_001)      \*      \* Summary of File:      \*   This file contains code which shows the user the number of coins to make K with      \*   given coins      \*  \*/  #include <stdio.h>  int findcoinnum(int coin[], int num, int n);    //function declaration  int main(){      int coin[10];      int n, k;      scanf("%d %d", &n, &k);     //n : 종류의 개수, k :가치의 합      for (int i = 0; i<n; i++){          scanf("%d", &coin[i]);  //동전의 가치들      }      int result = findcoinnum(coin, k, n);   //findcoinnum 함수 call      printf("%d", result);      return 0;  }  /\*\*   \* int findcoinnum(int coin[], int num, int n)   \*   \* findcoinnum 함수:   \*          given array의 값을 이용하여 주어진 K값을 만들기 위해 필요한   \*          동전 개수를 return 한다.   \*   \* parameters : int, int array   \*   \* return value : int   \*   \*/  int findcoinnum(int coin[], int num, int n){      int sum = 0;      int last = n;      for (int i=0; i<n; i++){          last--;          sum+=num/coin[last];    //coin[last]로 나눴을 때 몫만큼 동전 개수를 늘린다.          num=num%coin[last];     //num = coin[last]의 나머지      }      return sum; //sum을 return 한다.  } |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [제출 결과] |
| (유의사항:제출 결과에 대한 분석 및 간단 설명) | |



일반 상황일 때 output 6

텍스트, 장치이(가) 표시된 사진

자동 생성된 설명

만들고자 하는 값이 최대값보다 작을 때 output : 6

텍스트, 장치, 측정기이(가) 표시된 사진

자동 생성된 설명

만들고자 하는 값이 0일 때 output : 0

텍스트, 장치이(가) 표시된 사진

자동 생성된 설명

만들고자 하는 값이 한 동전의 값과 같을 때 output : 1

코드 설명 :

1. findcoinnum function을 call한다.
2. Last에 n값을 넣는다

-----------------------------------------------------

1. for문에 들어간 뒤 last값을 1 줄인다
2. Sum(필요한 코인의 개수)를 num/coin[last]만큼 더한다
3. num 에 coin[last]로 나눈 나머지를 저장한다

-----------------------------------------------------

1. sum을 return 한다
2. 값을 출력한다.

# #HW 3-2

|  |  |
| --- | --- |
|  | 각 물건은 무게 W와 가치 V를 가지고 최대 K만큼의 무게만을 넣을 수 있는 배낭만 들고 다닐 수 있다. 배낭에 넣을 수 있는 물건들의 가치의 최댓값을 찾아야한다. |

**Requirements**

* < 요구조건 #1: 간략한 핵심 요구 조건 설명>

각 물건은 무게 W와 가치 V를 가지고 최대 K만큼의 무게만을 넣을 수 있는 배낭만 들고 다닐 수 있다. 배낭에 넣을 수 있는 물건들의 가치의 최댓값을 찾아야한다.

## SOURCE CODE

|  |
| --- |
| n, k = map(int, input().split())    #n과 k값 받기  res = [[0 for i in range(k+1)]for j in range(n+1)]  arr = [[0,0]]  arr[0] = list(map(int, input().split()))    #첫번째 값 받기  for i in range(0, n-1):      arr.append(list(map(int, input().split()))) #두번째부터 n까지 받기  for i in range(0, n):      for j in range(0, k):   #무게 k인 배낭이 가지는 최대 가치 예제 : 7          weight = arr[i][0]  #weight 가져오기          value = arr[i][1]   #value 가져오기            if j+1 < weight:  #weight가 j를 넘는다면              res[i][j] = res[i-1][j]          else:           #넘지 않는다면              if(i==0):                  res[i][j] = max(value, res[0][j])              else :                  res[i][j] = max(res[i-1][j], res[i-1][j-weight]+value)  #res[i-1][j]과 res[i-1][j-weight]+value을                                                                          #비교하여 더 큰 값을 저장.  print(res[n-1][k-1])    #결과 출력 |

## OUTPUT (Screen Shots)

|  |  |
| --- | --- |
| #1 | [제출 결과] |
| (유의사항:제출 결과에 대한 분석 및 간단 설명) | |

텍스트이(가) 표시된 사진

자동 생성된 설명

일반 경우일 때 output : 14

텍스트이(가) 표시된 사진

자동 생성된 설명

최대 weight가 0일 때 output : 0

텍스트이(가) 표시된 사진

자동 생성된 설명

만들고자 하는 weight가 다른 모든 weight 보다 작아 들어갈 수 없을 때

Output : 0

Res[n][k]의 배열을 이용하여 N번째 물건까지의 각각의 weight와 value를 이용해 무게가 k의 배낭의 최대 가치를 찾아 결과값을 출력하게 된다.