Recursion__Examples__Algoringo

Recursion__Examples__Algoringo

Abstract

Make note atomic.

Point of capturing note is that I want get into the habit of capturing and if I feel the friction of, this is not atomic enough, not properly then less likely to capture notes as a habit. So you've given yourself the permission to just capture notes however you want to do it.

This note should have Questions, Ideas, Supplementary tools. Question tags question, Ideas tags paradox or counter-intuitive and Supplementary tools tags quote, anecdote and scientific study.

Status could be build(develop ideas), check(confirm idea, for accuracy, source, etc.), to-read and to-watch(find time to consume and take notes) as Input area. As output could anything if you want to contribute or publish status could be magazine name or instagram etc.

Title should be easy to find via keyword(general word). Putting the keywords that you think you will search for later down the line.

Idea:

12490__sum-of-subset

link

1부터 12까지의 숫자를 원소로 가진 집합 A가 있다. 집합 A의 부분 집합 중 N개의 원소를 갖고 있고, 원소의 합이 K인 부분집합의 개수를 출력하는 프로그램을 작성하시오.

해당하는 부분집합이 없는 경우 0을 출력한다. 모든 부분 집합을 만들어 답을 찾아도 된다.

예를 들어 N = 3, K = 6 경우, 부분집합은 { 1, 2, 3 } 경우 1가지가 존재한다.

Input

```
3
3 6
5 15
5 10
```

Code

```
import sys

sys.stdin = open('./sw_12490__sum-of-subset__input.txt', 'r')

lst = [i for i in range(1, 13)]
lst_len = len(lst)

for ts in range(int(input())):
    N, K = map(int, input().split())

    result = []
    for i in range(1 << lst_len):
        tmp = []
        for j in range(lst_len):
```

```
            if i & (1 << j):
                tmp.append(lst[j])
        result.append(tmp)

    cnt = 0
    for el in result:
        if len(el) == N:
            result_sum = 0
            for j in el:
                result_sum += j
            if result_sum == K:
                cnt += 1
    print(f'#{ts + 1} {cnt}')


sys.stdin.close()
```

2817__sum-of-subsequence

link

A1, A2, ... , AN의 N개의 자연수가 주어졌을 때, 최소 1개 이상의 수를 선택하여 그 합이 K가 되는 경우의 수를 구하는 프로그램을 작성하시오.

input

```
1
4 3
1 2 1 2
```

Code

```
def searching():
    cnt = 0
    for i in range(1, 1 << n):
        s = 0
        for j in range(n):
            if s > k:
                break
            if i & (1 << j):
                s += nums[j]
        if s == k:
            cnt += 1

    return cnt


for tc in range(1, int(input())+1):
    n, k = map(int, input().split())
    nums = list(map(int, input().split()))

    result = searching()
    # print(nums)
    print(f'#{tc} {result}')
```

- memory: 59,522 kb // time 2,300ms

```
def searching2(idx=0, sum_v=0):
    global cnt

    if sum_v == k:
        cnt += 1
        return

    if idx > n-1:
```

```
        return

    searching2(idx+1, sum_v+nums[idx])
    searching2(idx+1, sum_v)


for tc in range(1, int(input())+1):
    n, k = map(int, input().split())
    nums = list(map(int, input().split()))
    cnt = 0


    searching2()

    print(f'#{tc} {cnt}')
```

- memory 63,784 kb // time 760ms

```
def subsequences_with_sum(arr, target_sum):
    dp = [0] * (target_sum + 1) # dp[j]에는 수열의 합이 j인 경우의 수들이 들어감.
    dp[0] = 1  # 합이 0인 경우는 빈 부분 수열도 포함하므로 1로 초기화.

    #. target_sum 부터 num-1 까지 역순으로 탐색하기에 j-num은 항상 양수이며 idx를 벗어나지 않음
    #. 현재 합 j에서 현재 숫자 num을 뺀 경우의 수를 더하며, dp[j]에는 현재 수를 포함하지 않는 경우의 수가 이미 포함돼 있음.
    #. 역순으로 탐색해야 정답이 나오는데, 정순으로 탐색하면 dp[j]에는 수열의 합이 j인 경우의 수가 들어가지 않는다.
    #. 예로, num=1, j=2인 경우 dp[2]에는 dp[1]의 값 1이 더해지며, dp를 사용하는 의도와 전혀 다르게 동작한다.
    for num in arr:
        for j in range(target_sum, num - 1, -1):
        # for j in range(num, target_sum+1):
            dp[j] += dp[j - num]
            print(dp)

    return dp[target_sum]


for tc in range(1, int(input())+1):
    n, k = map(int, input().split())
    nums = list(map(int, input().split()))

    result = subsequences_with_sum(nums, k)
    print(f'#{tc} {result}')
```

- memory 44,528 kb // 119 time

Bearing:

> **Info**
>
> There is four area, West, East, North, South. These based on Question, Evidence and Conclusion.
> West means What's similar to X. In this field supplementary tools and related concepts are involved.
> East means What's opposite of X.
> North means Where X comes from.
> South means Where X leads to.

West: similar

East: opposite

North: theme / question

South: what does this lead to

Sources