‹ Back

Python different soluctions: DFS , BFS, DP

gyh75520 ★245 Last Edit: July 2, 2019 4:40 AM 3.0K VIEWS

34

DFS: TLE

```
#DFS:TLE
def findTargetSumWays(self, nums: List[int], S: int) -> int:
    self.res = 0
    def DFS(nums,i,summary):
        if i >= len(nums):
            if summary == S:
                self.res += 1
            return
        DFS(nums,i+1,summary+nums[i])
        DFS(nums,i+1,summary-nums[i])
    DFS(nums,0,0)
    return self.res
```

DFS + memo

```
#DFS + memo
def findTargetSumWays(self, nums: List[int], S: int) -> int:
    memo = {}
    def DFS(nums,i,summary):
        if i == len(nums):
            if summary == S:
                memo[(i,summary)] = 1
            else:
                memo[(i,summary)] = 0
        if (i,summary) not in memo:
            memo[(i,summary)] = DFS(nums,i+1,summary+nums[i]) + DFS(nums,i+1,summary-nums[i])
        return memo[(i,summary)]
    DFS(nums,0,0)
    return memo[(0,0)]
```

BFS: using summary,cnt in queue(level traversal)

```
# BFS:summary,cnt in queue
def findTargetSumWays(self, nums: List[int], S: int) -> int:
    import collections
    queue = {0:1}
    for n in nums:
        tmp = collections.defaultdict(int)
        for summary,cnt in queue.items():
            tmp[summary+n]+=cnt
            tmp[summary-n]+=cnt
        queue = tmp
        # print(queue)
    return queue[S]
```

DP: dp[k][i] = dp[k-1][i+nums[k]] + dp[k-1][i-nums[k]]
dp[k][i] which means the number of ways for first k-th element to reach a sum i
bottom-to-up：use Scrolling array

```
    # DP:dp[k][i] = dp[k-1][i+nums[k]] + dp[k-1][i-nums[k]]
    # dp[k][i] which means the number of ways for first k-th element to reach a sum i
    # bottom-to-up
    def findTargetSumWays(self, nums: List[int], S: int) -> int:
        summary = 0
        for n in nums:
            summary += abs(n)
        if summary < S:
            return 0

        dp_k0 = [0]*(2*summary+1)
        dp_k0[summary] = 1
        for n in nums:
            dp_k1 = [0]*(2*summary+1)
            for i in range(2*summary+1):
                a = dp_k0[i-n] if i-n >=0 else 0
                b = dp_k0[i+n] if i+n <(2*summary+1) else 0
                dp_k1[i] = a + b
            dp_k0 = dp_k1
        return dp_k0[S+summary]
```

DP: dp[k][i] = dp[k-1][i+nums[k]] + dp[k-1][i-nums[k]]
dp[k][i] which means the number of ways for first k-th element to reach a sum i
top-to-down： recurisive ， dp_Ksize[k][sum] == -1 means not visited

```
    # DP:dp[k][i] = dp[k-1][i+nums[k]] + dp[k-1][i-nums[k]]
    # dp[k][i] which means the number of ways for first k-th element to reach a sum i
    # top-to-down
    def findTargetSumWays(self, nums: List[int], S: int) -> int:
        summary = 0
        for n in nums:
            summary += abs(n)
        if summary < S:
            return 0

        dp_Ksize = [[-1]*(2*summary+1) for k in range(len(nums)+1)]
        for i in range(2*summary+1):
            dp_Ksize[0][i] = 0
        dp_Ksize[0][summary] = 1

        def recurisive(k,sum):
            if sum < 0 or sum>=(2*summary+1):
                return 0
            if dp_Ksize[k][sum] == -1:
                dp_Ksize[k][sum] = recurisive(k-1,sum+nums[k-1]) + recurisive(k-1,sum-nums[k-1])
            return dp_Ksize[k][sum]

        recurisive(len(nums),S+summary)
        return dp_Ksize[len(nums)][S+summary]
```


kimhyoil ★0 August 10, 2021 11:16 PM

what is time/space complexity each case?

0 Reply

Petersburg ★937 April 30, 2020 4:20 AM

In DFS + memo, why do you want memo[(0,0)] and what the purpose of summary? @gyh75520

0 Show 2 replies Reply

xueshengluanfei ★37 May 12, 2021 12:39 AM

DFS+memo get TLE

-2 Hide 1 reply Reply

xueshengluanfei ★37 May 12, 2021 12:41 AM

my bad, it works

0 Reply