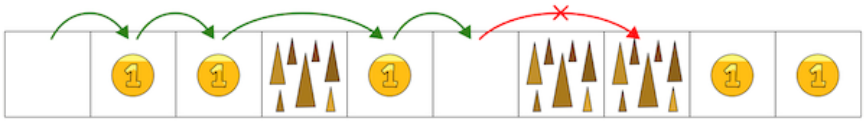
LAPORAN PENUGASAN 2 PAA

Anggota Kelompok : Abdan Hafidz dan Muhammad Farhan Arya Wicaksono

SOAL 1 : THORNS & COIN [https://codeforces.com/problemset/problem/1932/A]

a) **Deskripsi soal**

Kita diberikan sebuah grid 1-Dimensi, di mana petak ke – i akan memuat koin, duri, atau tidak memuat apapun, berturut – turut direpresentasikan sebagai ".","@", dan "*". Kita ingin menentukan berapa banyak koin maksimal yang bisa didapatkan selama pergerakan dari petak ujung kiri ke ujung kanan atau mungkin kita hanya berjalan beberapa langkah tidak sampai akhir.



Diberikan ketentuan bahwa kita hanya bisa bergerak dari suatu petak ke satu petak atau dua petak berikutnya, diberikan juga contoh masukan dan keluaran yang relevan.

Example

input Copy

```
3
10
.@@*@.**@@
5
.@@@@
15
.@@..@***..@@@*
```

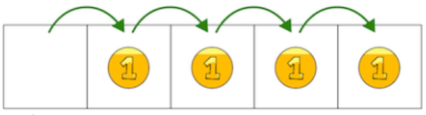
output Copy

```
3
4
3
```

Note

The picture for the first example is in the problem statement.

Here is the picture for the second example:



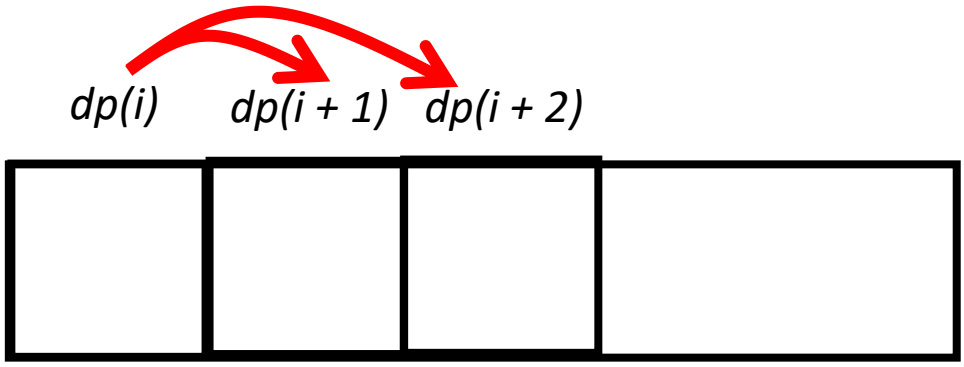
And here is the picture for the third example:

b) **Abstraksi**

Kita bisa melakukan *traversing* atau pergerakan dari petak ke – i menuju petak i + 1 atau petak i + 2, dalam permasalahan kali ini kita akan menentukan pergerakan mana yang menghasilkan jumlah koin secara maksimal.

Misalkan *dp(i)* = adalah jumlah koin maksimal yang bisa didapatkan dan item yang ada (koin, kosong, atau duri) pada petak ke – i sebagai *item[i]* .Kami bisa memodelkan secara *top-down* bahwa kita akan memaksimalkan langkah mana yang bisa mendapatkan koin paling banyak dengan menggunakan metode bottom – up memilih pegerakan ke satu petak berikutnya ini akan membuat dp(i + 1) atau dua petak berikutnya yaitu dp(i + 2).

GAMBAR PERMODELAN :



Jangan lupa untuk setiap pergerakan kita akan mengambil koin yang ada yaitu *item[i]* , kami mempunyai teknik yang mempermudah dalam penghitungan jumlah koin yaitu dengan melakukan encoding "." menjadi 0, "@" menjadi 1, dan "*" menjadi "-1" , untuk mempermudah perhitungan.

Kemudian tetapkan juga base-case bahwa kita akan berhenti pada saat sampai di petak paling kiri yaitu i = 1, saat di sini kita akan mendapatkan -1,0, atau 1 tergantung dengan nilai pada *item[i]*, kami juga mempertimbangkan adanya early-stopper dalam proses traverse saat dp(i – 1) dan dp(i – 2) memiliki nilai item = -1 atau kita hanya bisa bertemu dengan duri.

Model Fungsi :

$$dp(i) = \begin{cases} 0, & i = sizeof(\,item) \; or \; item[i] = -1 \\ \max\big(dp(i+1), dp(i+2)\big) + item[i], & i \geq 1 \end{cases}$$

c) Pseudocode

```
Function dp(i):
    If i >= size of item or item[i] == -1:
        Return 0
    Else:
      Return max(dp(i + 1), dp(i + 2)) + item[i]
```

d) Implementasi

Solution : O(N)

```
def Solve():
    N = int(input())
    inItem = input()
    item = []
    for r in inItem:
        if(r == '.'):
            item.append(0)
        if(r == "@"):
            item.append(1)
        if(r == '*'):
            item.append(-1)
    def dp(i):
        if(i > N):
            return 0
        else:
            if(item[i - 1] == -1): return 0
            # print(item[i - 1],end="")
            return max(dp(i + 1),dp(i + 2)) + item[i - 1]
    print(dp(1))

T = int(input())
for _ in range(T):
    Solve()
```

e) Lampiran

#	When	Who	Problem	Lang	Verdict	Time	Memory
292516623	Nov/20/2024 23:39 UTC+7	mengCP	1932A - Thorns and Coins	Python 3	Accepted	93 ms	0 KB

https://codeforces.com/problemset/submission/1932/292516623

SOAL 2 : Dice Combinations (Link: https://basecamp.eolymp.com/en/problems/9036)

a) **Deskripsi Soal**

Kita diberikan bilangan bulat N, diminta untuk menghitung ada berapa banyak cara atau kombinasi menjumlahkan beberapa angka asli (1 – 6) sehingga hasilnya = N.

Contoh untuk N = 3 ada 4 cara yaitu :

```
1+1+1
1+2
2+1
3
```

b) **Abstraksi**

Kita dapat menyelesaikan dengan metode top-down di mana misalkan dp(x) menyatakan banyak cara untuk membuat kombinasi penjumlahan sehingga hasilnya adalah x dengan penjumlahnya adalah $1 \leq i \leq 6$. Kita dapat menentukan bahwa setiap kali kita menemukan bahwa y memenuhi sebagai penjumlah maka x akan berkurang sebesar y, nantinya y adalah bilangan asli $\leq$ x sehingga kita dapat membuat model dynamic programming-nya adalah :

$$dp(x,y) = \begin{cases} 0, & x \leq 0 \\ 1, & x = 0 \\ \sum_{i=1}^{6} dp(x-i), & x > 0 \end{cases}$$

c) **Pseudocode**

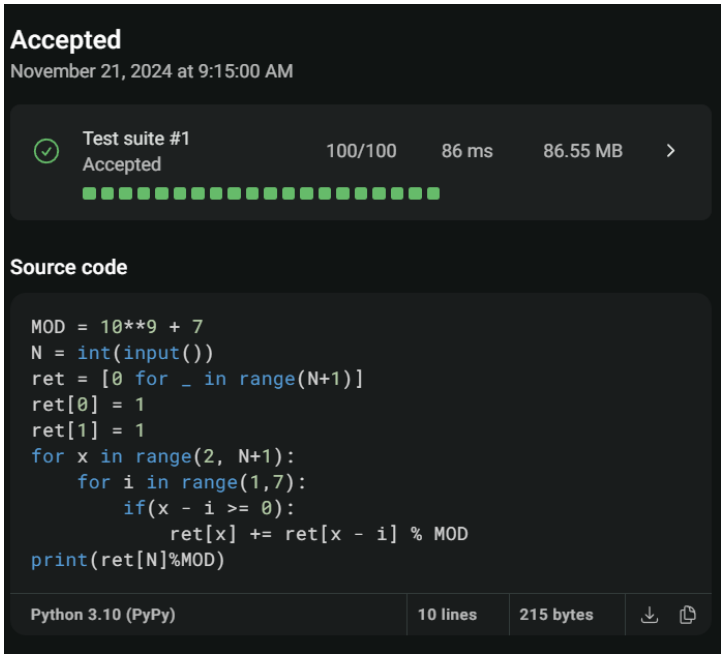
```
function dp(x, y):
    if x <= 0:
        return 0
    if x == 0:
        return 1

    result = 0
    for i from 1 to 6:
        result = result + dp(x - i, y)

return result
```

d) **Implementasi**

Solusi : O(N)

```
MOD = 10**9 + 7
N = int(input())
ret = [0 for _ in range(N+1)]
ret[0] = 1
ret[1] = 1
for x in range(2, N+1):
    for i in range(1,7):
        if(x - i >= 0):
            ret[x] += ret[x - i] % MOD
print(ret[N]%MOD)
```

e) **Lampiran**



SOAL 3 : Kefa and First Step (Link: https://codeforces.com/problemset/problem/580/A)

a) **Deskripsi Soal**

Soal ini sangat mirip dengan problem *Longest Increasing Subsequences (LIS)* namun dengan sedikit modifikasi di mana kita diberikan angka a_i sebanyak n untuk ($1 \leq i \leq n$) kita akan memilih sub-sekuens dari a sehingga $a_j \leq a_{j+1} \leq a_{j+2} \leq \dots$.

b) **Abstraksi**

Transisi DP secara top-down yang kita miliki adalah misalkan saat ini kita berada di indeks i , kita cukup memeriksa apakah bilangan arr[i] >= arr [i - 1] , jika ya maka kita akan ambil dan menghitung pertambahan panjang subsekuens yang memenuhi dengan 1, sehingga kita dapat membuat permodelan :

$$dp(i) = \begin{cases} 0, & x \leq 0 \\ 1, & x = 1 \\ dp(i-1) + 1, & if\ arr[i] \geq arr[i-1] \end{cases}$$

c) **Pseudocode**

```
function dp(x, y):
    if x <= 0:
        return 0
    if x == 0:
        return 1

    result = 0
    for i from 1 to x:
        result = result + dp(x - i, y)

    return result
```

d) **Implementasi**

Solusi : O(N)

```
def solve(arr):
    n = len(arr)
    dp = [1] * n
    for i in range(1, n):
        if arr[i] >= arr[i - 1]:
            dp[i] = dp[i - 1] + 1
        else:
            dp[i] = 1

    return max(dp)
n = int(input())
arr = list(map(int,input().split()))
print(solve(arr))
```

e) **Lampiran**

My Submissions

#	When	Who	Problem	Lang	Verdict	Time	Memory
292559176	Nov/21/2024 10:33 UTC+7	mengCP	A - Kefa and First Steps	Python 3	Accepted	124 ms	13200 KB

https://codeforces.com/contest/580/submission/292559176

SOAL 4 : Tri Tiling (Link: https://basecamp.eolymp.com/en/problems/482)

a) **Deskripsi Soal**

Kita diberikan lantai berukuran 3 x N kita dapat mengisi lantai tersebut dengan menggunakan ubin ukuran 2 x 1, tentukan ada berapa banyak cara penyusunan ubin pada lantai tanpa ada tumpang tindih.

b) **Abstraksi**

Kita dapat memisalkan dp(n) = banyak cara mengisi lantai berukuran 3 x N dengan ubin 2 x 1.

$f(n) = f(n - 2) + 2g(n-1)$ ...(1)

$g(n) = f(n - 1) + g(n - 2)$ ...(2)

dari substitusi persamaan (1) diperoleh bahwa

$2g(n - 1) = f(n) - f(n - 2)$ .. (3)

dari substitusi persamaan (2) diperoleh bahwa

$f(n - 1) = g(n) - g(n - 2)$ .. (4)

dari substitusi persamaan (3) diperoleh bahwa

$g(n) = (f(n+1) - f(n - 1)) / 2$

dan ini membuat

$g(n - 2) = (f(n - 1) - f(n - 3)) /2$ .. (5)

subsitusi persamaan (3) ke (2)

$g(n) = f(n - 1) + (f(n - 1) / 2 - f(n - 3) / 2)$ .. (6)

subsitusi persamaan (6) ke (1)

$2g(n-1) = f(n-2)/2 - f(n-4)/2)$

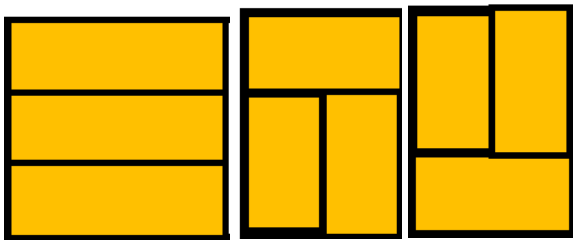
$f(n) = f(n-2) + 2(f(n-2) + f(n-2)/2 - f(n-4)/2)$

$f(n) = f(n-2) + 2f(n-2)) + f(n-2) - f(n-4)$

Ditemukan hasilnya adalah :

$f(n) = 4f(n-2) - f(n-4)$, untuk basecase $f(0) = 1$, $f(1) = 0$, $f(2) = 3$, $f(3) = 0$

karena terdapat 3 cara untuk mengisi lantai berukuran 3 x 2 :



$f(1)$ dan $f(3) = 0$ karena bukan kelipatan 2 dan 3.

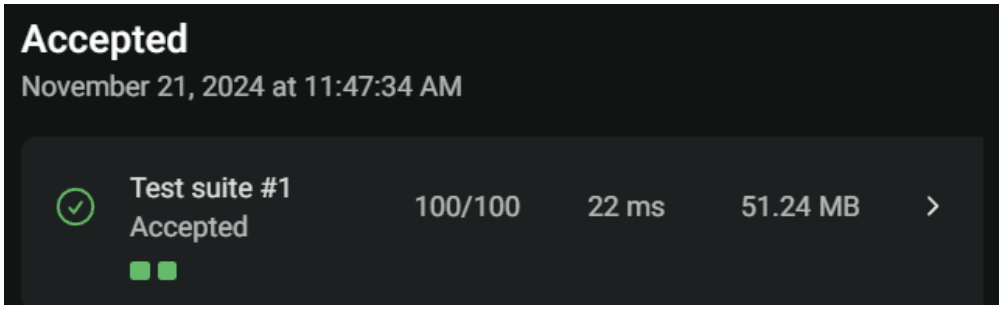
c) Pseudocode

```
function f(n):
    if n == 0:
        return 1
    if n == 1:
        return 0
    if n == 2:
        return 3
    if n == 3:
        return 0

    return 4 * f(n - 2) - f(n - 4)
```

d) Implementasi

```
memo = {
    0:1,
    1:0,
    2:3,
    3:0
}
def dp(n):
    if(n<0):return 0
    # print(n)
    if(memo.get(n) != None):
        return memo[n]
    else:
        memo[n] = 4*dp(n - 2) - dp(n-4)
        return memo[n]
while(1):
    N = int(input())
    if(N == -1):
        break
    print(dp(N))
```

e) **Lampiran**



SOAL 5 : Great Julya Calendar (Link:https://codeforces.com/problemset/problem/331/C1)

a) **Deskripsi Soal**

Kita diberikan bilangan bulat N, diminta untuk mengurangkan N dengan setiap digit yang ada sehingga hasilnya 0, kita akan menghitung berapa langkah pengurangan minimum yang diperlukan.

$24 \rightarrow 20 \rightarrow 18 \rightarrow 10 \rightarrow 9 \rightarrow 0$

24 dikurangi dengan 4 menjadi 20
20 dikurangi dengan 2 menjadi 18
18 dikurangi dengan 8 menjadi 10
10 dikurangi dengan 1 menjadi 9
9 dikurangi dengan 9 menjadi 0

b) **Abstraksi**

Permisalan dp(n) = langkah minimum mengurangi n dengan digit – digit yang tersedia pada setiap langkah pengurangan sampai hasilnya 0, kita akan menginisiasi base case saat *digit_size(n) = 1* kembalian akan bernilai 1. Permodelan yang dibuat adalah kita akan memilih langkah minimum di antara langkah pengurangan n dengan i di mana i adalah semua anggota himpunan digit dari n saat ini, jangan lupa setiap pengurangan yang kita pilih kita catat sebagai 1 langkah

$$\left(\min_{\text{i=digits of n}}(dp(n-i)\right)+1$$

Untuk basecase

dp(n) = 1 jika n hanya satu digit dan dp(n) = 0 jika n $\leq$ 0

c) **Pseudocode**

```
FUNCTION dp(n):
    # Base cases
    IF n < 0:
        RETURN 0
    IF n < 10:
        RETURN 1

    # Initialize minimum value
    min_value ← ∞

    # Iterate through the digits of n
    FOR each digit d in digits_of(n):
        IF d ≠ 0:  # Ignore zero to avoid infinite loop
            min_value ← MIN(min_value, dp(n - d))

    RETURN min_value + 1
```

d) **Implementasi**

Notes : awalnya saya mencoba untuk mengirimkan submission menggunakan Python dengan kode sebagai berikut :

```
import sys
sys.setrecursionlimit(10 ** 9)

memo = dict()

def dp(n):
    if(memo.get(n) != None):
        return memo[n]
    else:
        if(n <= 0): return 0
        if(n%10 == n and n > 0):
            memo[n] = 1
            ret = 1
        else:
            def solve(n,i):
                if(i != '0'):
                    return dp(int(n) - int(i))
                else:
                    return float('inf')
            ret = min([solve(n,i) for i in set(str(n))]) + 1
            memo[n] = ret
        return memo[n]
N = int(input())
print(dp(N))
```

https://codeforces.com/problemset/submission/331/292592582

Namun terdapat issue pada judger codeforces yaitu stack-overflow mempertimbangkan proses komputasi bahasa Python yang sangat lambat.

Test: #11, time: 92 ms., memory: 2800 KB, exit code: -1073741571 (STATUS_STACK_OVERFLOW), checker exit code: 0, verdict: RUNTIME_ERROR

Input

```
12343
```

Checker Log

```
Exit code is -1073741571
```

Oleh karenanya saya membuat salinan kode dalam bahasa C agar lebih cepat :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <stdbool.h>

typedef struct {
    long long key;
    int value;
    bool is_set;
} HashNode;

#define HASH_MAP_SIZE 1000003

HashNode hash_map[HASH_MAP_SIZE];

long long hash(long long key) {
    return (key % HASH_MAP_SIZE + HASH_MAP_SIZE) %
HASH_MAP_SIZE;
}

void set_hash(long long key, int value) {
    long long idx = hash(key);
    while (hash_map[idx].is_set && hash_map[idx].key != key)
{
        idx = (idx + 1) % HASH_MAP_SIZE;
    }
    hash_map[idx].key = key;
    hash_map[idx].value = value;
    hash_map[idx].is_set = true;
}

bool get_hash(long long key, int* value) {
    long long idx = hash(key);
    while (hash_map[idx].is_set) {
        if (hash_map[idx].key == key) {
            *value = hash_map[idx].value;
            return true;
        }
        idx = (idx + 1) % HASH_MAP_SIZE;
    }
    return false;
}

int dp(long long n) {
    int memo_value;
    if (get_hash(n, &memo_value)) {
        return memo_value;
    }
```

```
    if (n <= 0) return 0;
    if (n < 10) return 1;

    int min_value = INT_MAX;
    long long temp = n;
    while (temp > 0) {
        int digit = temp % 10;
        temp /= 10;
        if (digit != 0) {
            min_value = (min_value < dp(n - digit)) ?
min_value : dp(n - digit);
        }
    }
    int result = min_value + 1;

    set_hash(n, result);
    return result;
}

int main() {
    memset(hash_map, 0, sizeof(hash_map));
    long long N;
    scanf("%lld", &N);
    printf("%d\n", dp(N));
    return 0;
}
```

Dan alhamdulillah secara ajaib solusipun AC

https://codeforces.com/problemset/submission/331/292593578

#	When	Who	Problem	Lang	Verdict	Time	Memory
292593578	Nov/21/2024 16:54 UTC+7	mengCP	331C1 - The Great Julya Calendar	GNU C11	Accepted	812 ms	31300 KB

e) Lampiran

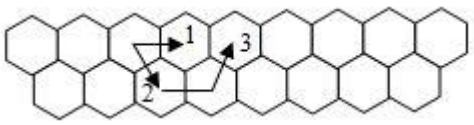
https://codeforces.com/problemset/submission/331/292593578

#	When	Who	Problem	Lang	Verdict	Time	Memory
292593578	Nov/21/2024 16:54 UTC+7	mengCP	331C1 - The Great Julya Calendar	GNU C11	Accepted	812 ms	31300 KB

SOAL 6 : Honeycomb (Link: https://basecamp.eolymp.com/en/problems/5092)

a) Deskripsi Soal

Kita diberikan sebuah sarang lebah berukuran N tentukan ada berapa banyak konfigurasi / cara bergerak dari pentagon kecil paling kiri ke paling kanan, jika model pergerakan dicontohkan sebagai berikut :



b) Abstraksi Soal

Kita misalkan dp(n) = banyak cara pergerakan lebah menuju suatu lokasi / pentagon kecil ke – n. Lebah dapat bergerak ke petak tersebut dalam satu langkah sehingga sisa pergerakannya adalah n – 1 atau bergerakan dalm dua langkah (dengan melewati pentagon tetangga terlebih dahulu) dan ini membuat sisa pergerakan yang ada n – 2, dari sini kita dapat memodelkan metode DP nya adalah dp(n) = dp(n – 1) + dp(n – 2) untuk base case dp(1) = 1 karena untuk menuju pentagon pertama hanya ada satu cara dan menuju pentagon kedua ada satu cara saja ini membuat dp(2) = 1.

Sebenarnya soal ini sangat umum dan familiar, sudah sering di bahas untuk topik fungsi rekursif. Kita juga dapat melakukan uji pola dan didapat banyak cara pergerakan ke suatu pentagon sama seperti angka pada barisan fibonacci.

Fungsi DP yang diterapkan akan sama seperti fungsi fibonacci, oleh karenanya kita dapat memodelkan :

$$dp(n) = \begin{cases} 1, & n = 1 \\ 1, & n = 2 \\ dp(n-1) + dp(n-2), & n > 2 \end{cases}$$

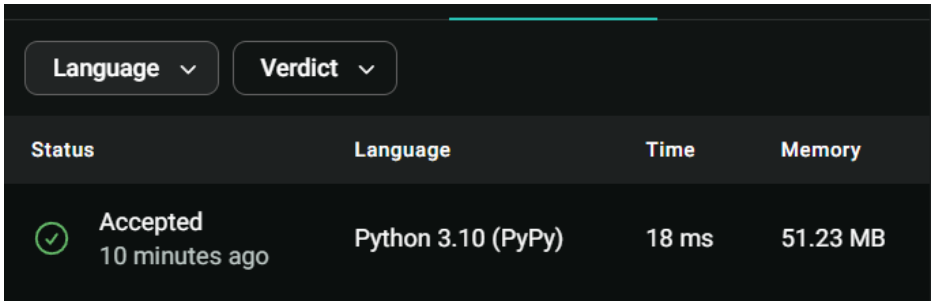
c) Pseudocode

```
FUNCTION dp(n):
    IF n = 0 THEN
        RETURN 0
    ELSE IF n = 1 THEN
        RETURN 1
    ELSE
        RETURN dp(n - 1) + dp(n - 2)
```

d) **Implementasi**

```
memo = dict()
def dp(n):
    if(memo.get(n) != None):
        return memo[n]
    else:
        if(n == 1 or n == 2):
            return 1
        else:
            memo[n] = dp(n - 1) + dp(n - 2)
            return memo[n]
N = int(input())
print(dp(N))
```

e) Lampiran


Language
Verdict
Status
Language
Time
Memory
Accepted
10 minutes ago
Python 3.10 (PyPy)
18 ms
51.23 MB