PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

O que é programação competitiva?

Maratona
PET COMPUTAÇÃO

GAIA
GRUPO DE APLICAÇÕES DE INTELIGÊNCIA ARTIFICIAL

Ola!

Eu sou Amy Kuhn Hammes

Vou ser uma dos tutores que vão acompanhar vocês nessa aula

Contato: amy@inf.ufpel.edu.br

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Ola!

Eu sou Fabricio Barbosa

Vou ser um dos tutores que vão acompanhar vocês nessa aula

Contato: fbviegas@inf.ufpel.edu.br

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

O que é uma maratona de programação?

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

> Given well-know computer science problems, solve them as quickly as possible!

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

well-know problems

Mas quais são esses problemas a serem resolvidos?

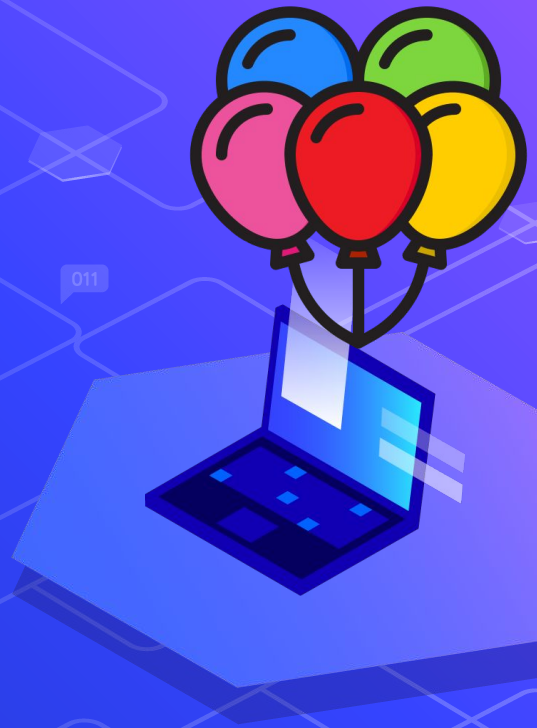
PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Principais eventos no Brasil

OBI & Maratona SBC

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Por que eu deveria querer participar dessas competições?



PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Tipos de problemas

Tópicos principais

- Ad-hoc
- Paradigmas
- Grafos
- Matemática
- Strings
- Geometria

Ad-hoc

Oque é?

Tipos mais comum de ad-hoc

- Game (Card)
- Game (Chess)
- Palindromes
- Anagrams
- Real Life
- Time
- Time Waster

Os 4 paradigmas

- Busca completa
- Dividir e conquistar
- Guloso
- Programação dinâmica

O que é a busca completa

A.K.A. força bruta, complete search, brute force

Vantagens

- Simples
- Fácil de codar
- Fácil de debugar
- Intuitivo

Desvantagens

- -Lento (TLE)
- -Geralmente existem opções melhores

Quando usamos Busca completa?

“ The Kiss principle

- **K***eep*
- **I***t*
- **S***imple and*
- **S***hort*

Podando uma busca completa

Ou, pruning

Iterativo x recursivo

Iterativo

- For/while
- Intuitivo
- Fácil de debbugar
- Performance

Recursivo

- Recursão
- Mais rápido de codar

Depende do problema e de sua preferência!

Leetcode 46

46. Permutations

Solved

Medium | Topics | Companies

Given an array `nums` of distinct integers, return all the possible permutations. You can return the answer in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Leetcode 46

```python
def permuta(elems, ans):
    if len(elems) == 0:
        return [ans]
    aux = []
    for i in range(len(elems)):
        aux += permuta(elems[0:i]+elems[i+1:len(elems)], ans+[elems[i]])
    return aux

class Solution:
    def permute(self, nums: List[int]) -> List[List[int]]:
        return permuta(nums, [])
```

O que é a dividir e conquistar

A.K.A. divide and conquer ou D&C

Passos

Dividir o problema original em subproblemas - geralmente na metade ou próximo disso

Resolver esses problemas menores, oque é mais fácil que o problema maior se necessário, juntar essas soluções para conseguir resolver o problema principal.

Busca binaria

```c
int buscaBinaria(int *arr, int piso, int teto, int val) {
    if(teto < piso) return -1;
    int meio = (piso + teto) / 2;
    if(arr[meio] == val) return meio;
    if(arr[meio] > val) return buscaBinaria(arr, piso, meio-1, val);
    return buscaBinaria(arr, meio+1, teto, val);
}
```

Busca binaria

```
int lista[16] = {1, 3, 5, 6, 7, 9, 13, 14, 17, 19, 20, 21, 22, 29, 33, 39}, a;
a = buscaBinaria(lista, 0, 15, 22);
printf("%d\n", a);
```

```
piso = 0, teto = 15, valor no meio = 14
piso = 8, teto = 15, valor no meio = 21
piso = 12, teto = 15, valor no meio = 29
piso = 12, teto = 12, valor no meio = 22
12
```

Usos menos óbvios

Leetcode 654

654. Maximum Binary Tree

Solved

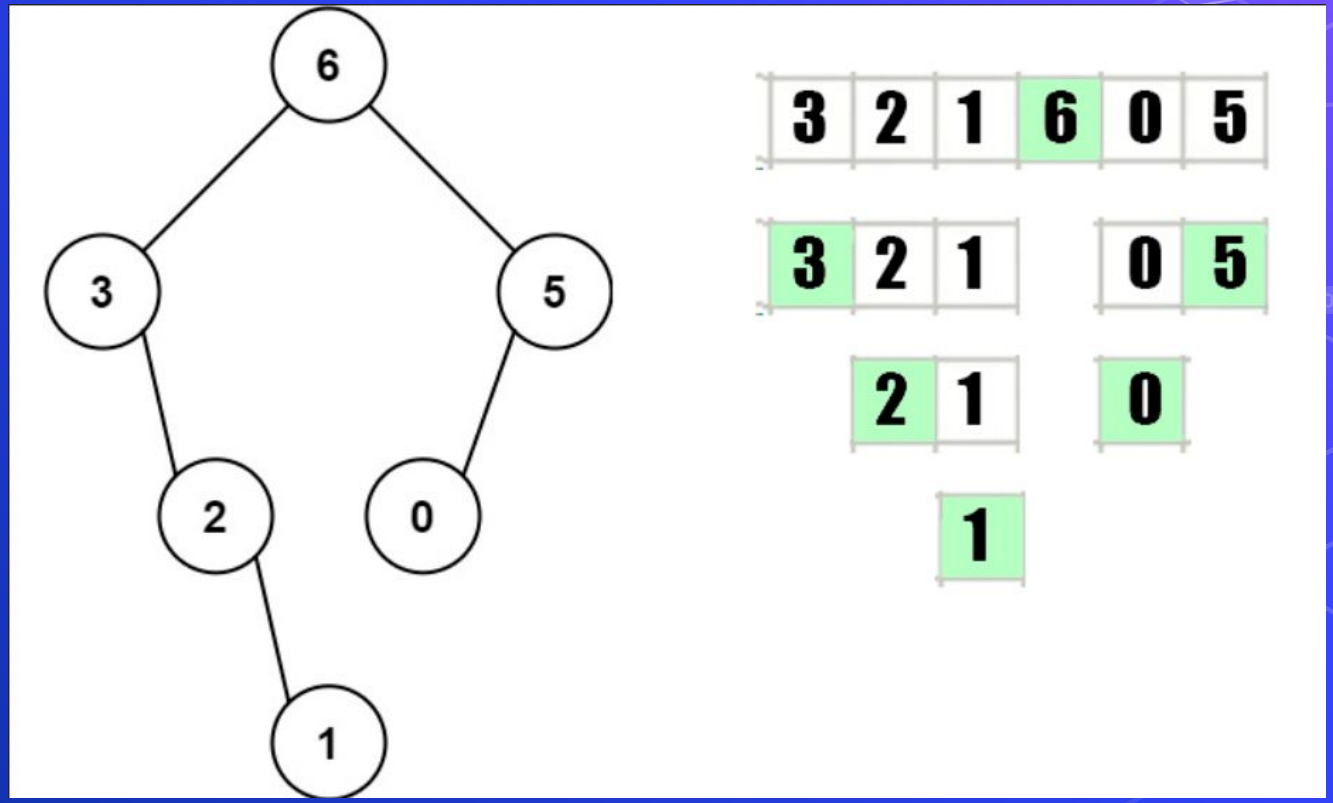
Medium | Topics | Companies

You are given an integer array `nums` with no duplicates. A **maximum binary tree** can be built recursively from `nums` using the following algorithm:

1. Create a root node whose value is the maximum value in `nums`.
2. Recursively build the left subtree on the **subarray prefix** to the **left** of the maximum value.
3. Recursively build the right subtree on the **subarray suffix** to the **right** of the maximum value.

Return *the **maximum binary tree** built from* `nums`.

Leetcode 654



Leetcode 654

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

Leetcode 654

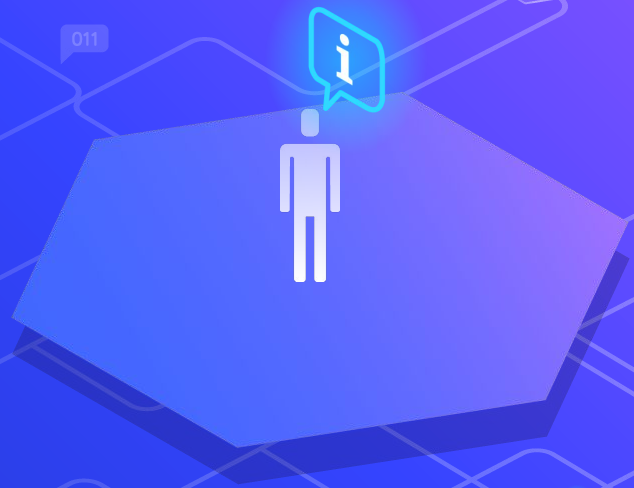
```python
class Solution:
    def constructMaximumBinaryTree(self, nums: list[int]) -> TreeNode:
        if len(nums) == 0:
            return None

        aux = nums.index(max(nums))

        node = TreeNode()
        node.val = nums[aux]
        node.left = self.constructMaximumBinaryTree(nums[0: aux])
        node.right = self.constructMaximumBinaryTree(nums[aux+1: len(nums)])

        return node
```

O que é o algoritmo guloso

A.K.A. Greedy

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Propriedade Greedy



Vantagens

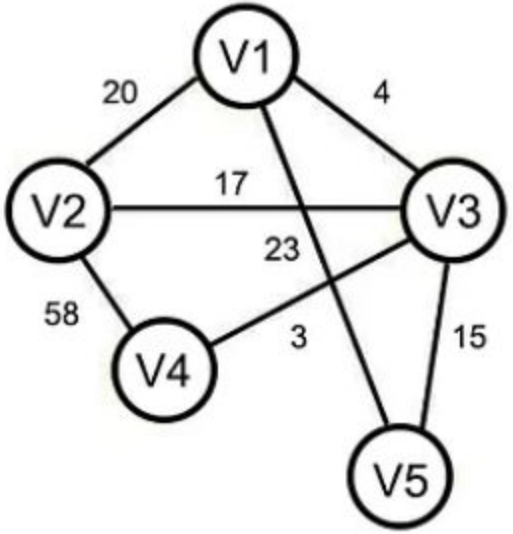
- Fácil de codar
- Fácil de debbugar
- Rápido

Desvantagens

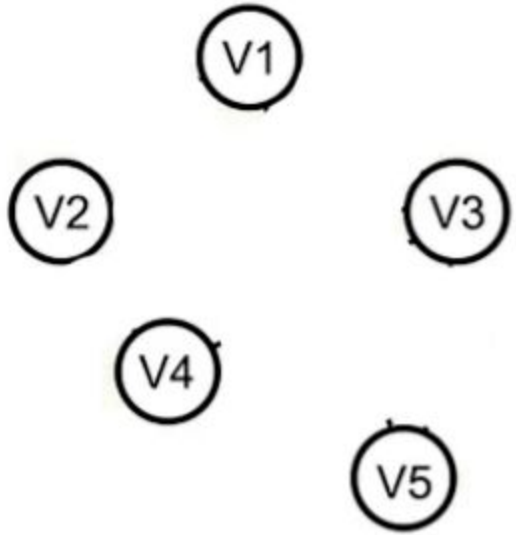
- Nem todos os problemas possuem a propriedade greedy
- Difícil de identificar quais problemas podem ser resolvidos por greedy

Kruskal

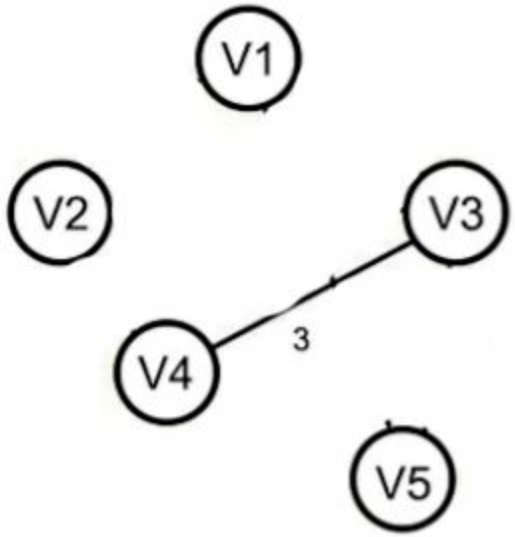
Uso de Greedy para encontrar
a MSP de uma árvore



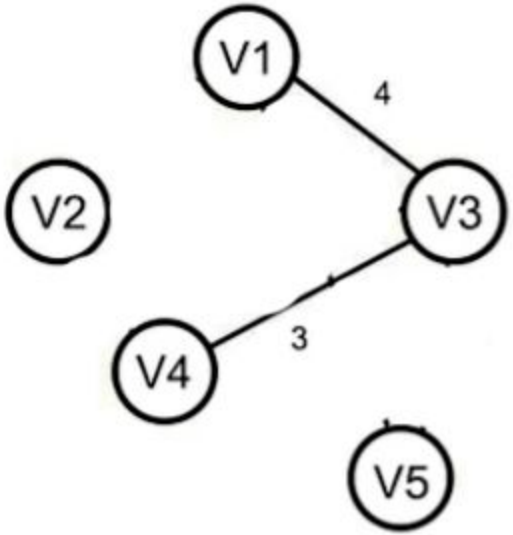
	V1	V2	V3	V4	V5
V1	0*	20	4	0*	23
V2	20	0*	17	58	0*
V3	4	17	0*	3	15
V4	0*	58	3	0*	0*
V5	23	0*	15	0*	0*



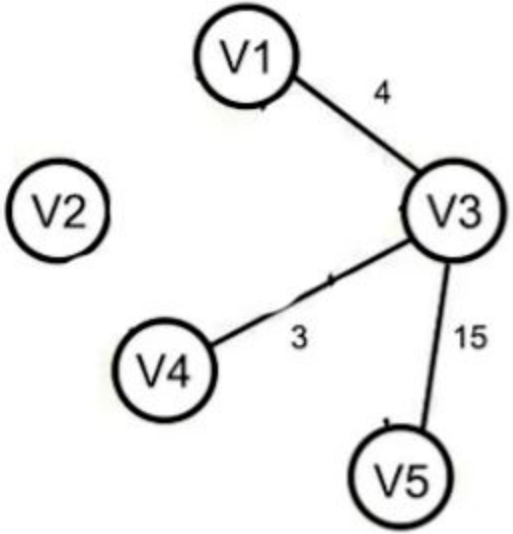
	V1	V2	V3	V4	V5
V1	0*	0*	0*	0*	0*
V2	0*	0*	0*	0*	0*
V3	0*	0*	0*	0*	0*
V4	0*	0*	0*	0*	0*
V5	0*	0*	0*	0*	0*



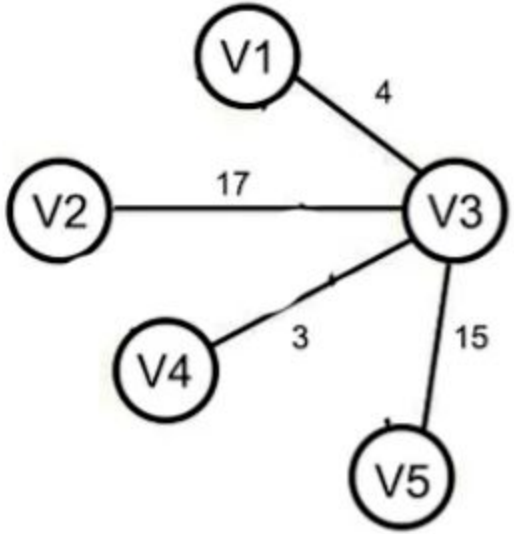
	V1	V2	V3	V4	V5
V1	0*	0*	0*	0*	0*
V2	0*	0*	0*	0*	0*
V3	0*	0*	0*	3	0*
V4	0*	0*	3	0*	0*
V5	0*	0*	0*	0*	0*



	V1	V2	V3	V4	V5
V1	0*	0*	4	0*	0*
V2	0*	0*	0*	0*	0*
V3	4	0*	0*	3	0*
V4	0*	0*	3	0*	0*
V5	0*	0*	0*	0*	0*



	V1	V2	V3	V4	V5
V1	0*	0*	4	0*	0*
V2	0*	0*	0*	0*	0*
V3	4	0*	0*	3	15
V4	0*	0*	3	0*	0*
V5	0*	0*	15	0*	0*



	V1	V2	V3	V4	V5
V1	0*	0*	4	0*	0*
V2	0*	0*	17	0*	0*
V3	4	17	0*	3	15
V4	0*	0*	3	0*	0*
V5	0*	0*	15	0*	0*

Beecrowd 1524

beecrowd | 1524

Fila do Bandejão

Por Marcio Oshiro, USP Brazil

Timelimit: 1

Um fenômeno muito comum na fila do bandejão (também conhecido como restaurante universitário) é ver uma pessoa recém chegada entrar no interior na fila em vez de no final. Isso ocorre sempre que tal pessoa encontra alguém de seu grupo já na fila.

Interessado em estudar esse fenômeno, um amigo pediu para você escrever um programa para estudar os grupos presentes na fila. Podemos supor que existem **K** grupos diferentes e toda pessoa pertence a exatamente um desses grupos. O tamanho de um grupo é definido pela distância entre as duas pessoas mais distantes dentro do grupo. Se o grupo consiste de apenas uma pessoa, seu tamanho é zero. Considerando que os grupos se organizam de forma que a soma dos tamanhos dos **K** grupos seja mínima, seu programa deve determinar qual é o valor dessa soma.

Beecrowd 1524

Entrada

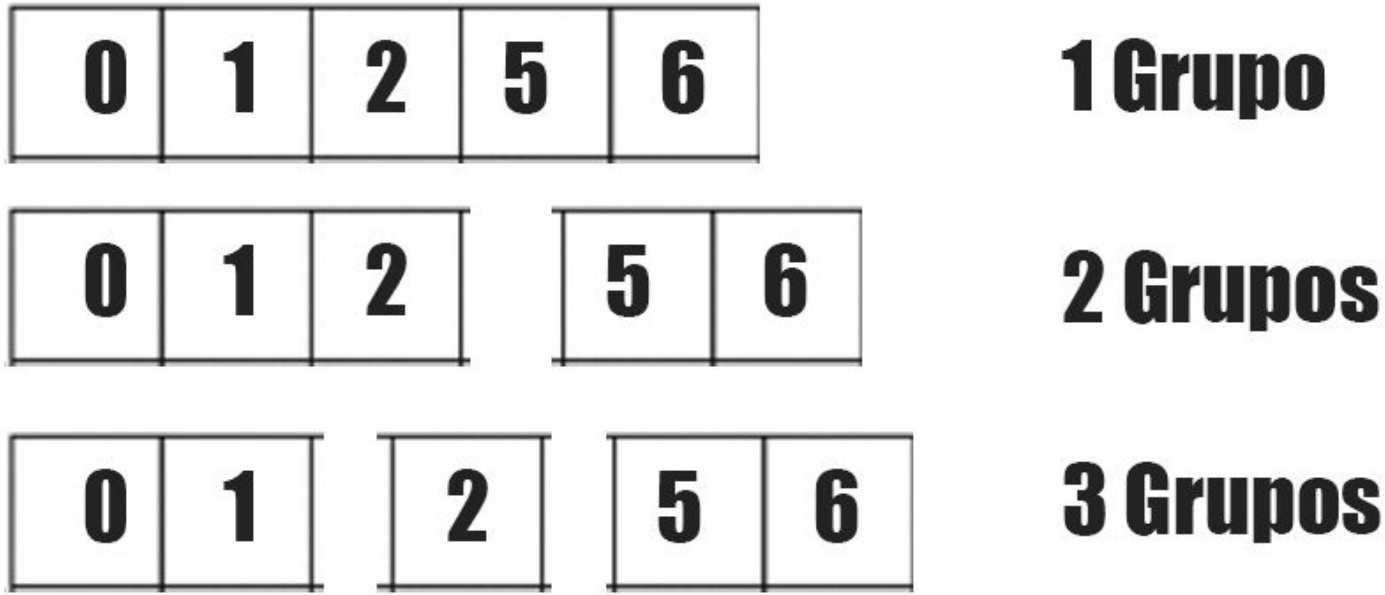
A entrada é composta por diversas instâncias e termina com o final de arquivo (EOF). A primeira linha de cada instância contém os inteiros **N**, indicando o número de pessoas na fila, e **K**, indicando o número de grupos **(1 ≤ K < N ≤ 1.000)**. Na linha seguinte são apresentados **N – 1** inteiros, $a_2, \ldots, a_N$, **(0 ≤ a_2 ≤ ··· ≤ a_N ≤ 1.000.000)** indicando as posições de cada pessoa em relação à primeira pessoa da fila **A** posição da primeira pessoa é omitido, pois é sempre zero.

Saída

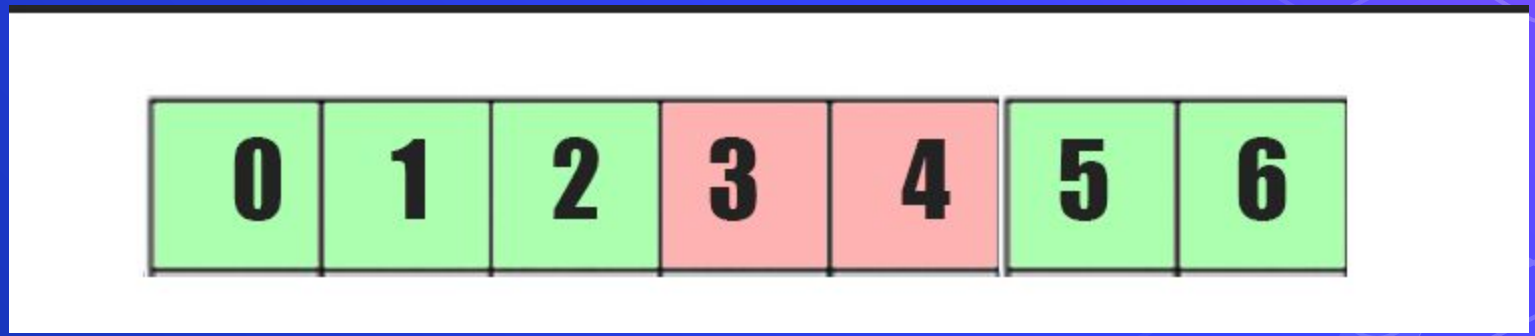
Para cada instância, imprima uma única linha contendo o valor mínimo que a soma dos tamanhos dos **K** grupos pode ter.

Exemplo de Entrada	Exemplo de Saída
5 2 1 2 5 6 4 3 0 1 2	3 0

Beecrowd 1524



Beecrowd 1524



Beecrowd 1524

```
while True:
    try:
        
        
        
        
        
        
        
        
        
        
        
        
        
    except EOFError:
        break
```

Beecrowd 1524

```python
while True:
    try:
        n, k = list(map(int, input().split()))
        l = list(map(int, input().split()))
    except EOFError:
        break
```

Beecrowd 1524

```python
while True:
    try:
        n, k = list(map(int, input().split()))
        l = list(map(int, input().split()))
        l.insert(0, 0)
        l.sort()
    except EOFError:
        break
```

Beecrowd 1524

```python
while True:
    try:
        n, k = list(map(int, input().split()))
        l = list(map(int, input().split()))
        l.insert(0, 0)
        l.sort()
        buracos = []
        for i in range(n-1):
            buracos.append(l[i+1] - l[i])
    except EOFError:
        break
```

Beecrowd 1524

```python
while True:
    try:
        n, k = list(map(int, input().split()))
        l = list(map(int, input().split()))
        l.insert(0, 0)
        l.sort()
        buracos = []
        for i in range(n-1):
            buracos.append(l[i+1] - l[i])
        buracos.sort(reverse=True)
        ans = l[-1] - sum(buracos[0: k-1])
        print(ans)

    except EOFError:
        break
```

Beecrowd 1524

SUBMISSÃO # 43158527

PROBLEMA:	1524 - Fila do Bandejão
RESPOSTA:	**Accepted**
LINGUAGEM:	Python 3.9 (Python 3.9.4) [+1s]
TEMPO:	0.071s
TAMANHO:	392 Bytes
MEMÓRIA:	-
SUBMISSÃO:	23/01/2025 15:22:17

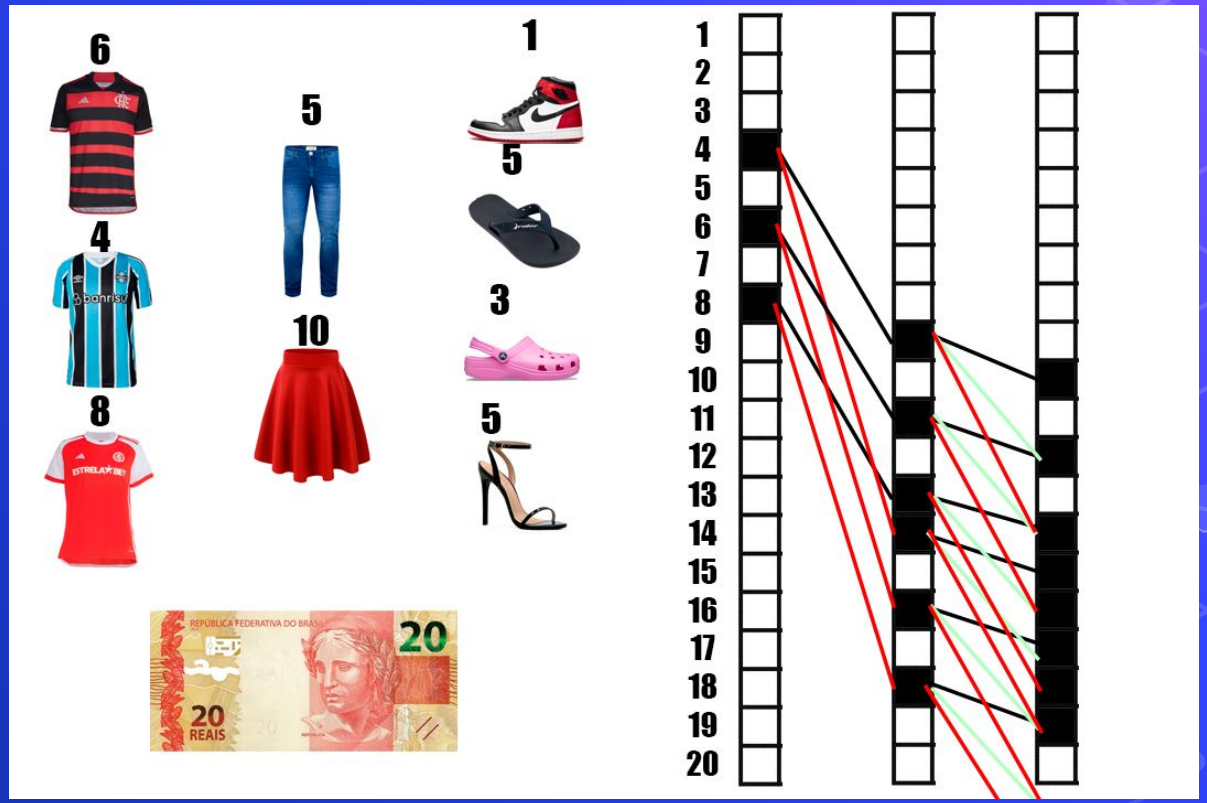
O que é Programação Dinâmica?

A.K.A. Dynamic Programming ou DP

Oque resolvemos com DP?

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Exemplo



Top-Down

- Recursivo
- Prós
  - Parece bastante com a busca completa
  - Computa os casos apenas quando necessário
- Contras
  - Pode ser mais lento se os casos forem muito revisitados
  - Tabela de memória tende a ser maior

Bottom-up

- Iterativo
- Prós
  - É mais rápido se os casos forem muito revisitados
  - Tabela de memória tende a ser menor
- Contras
  - menos intuitivo
  - preenche casos que podem não serem usados

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Leetcode 198

198. House Robber

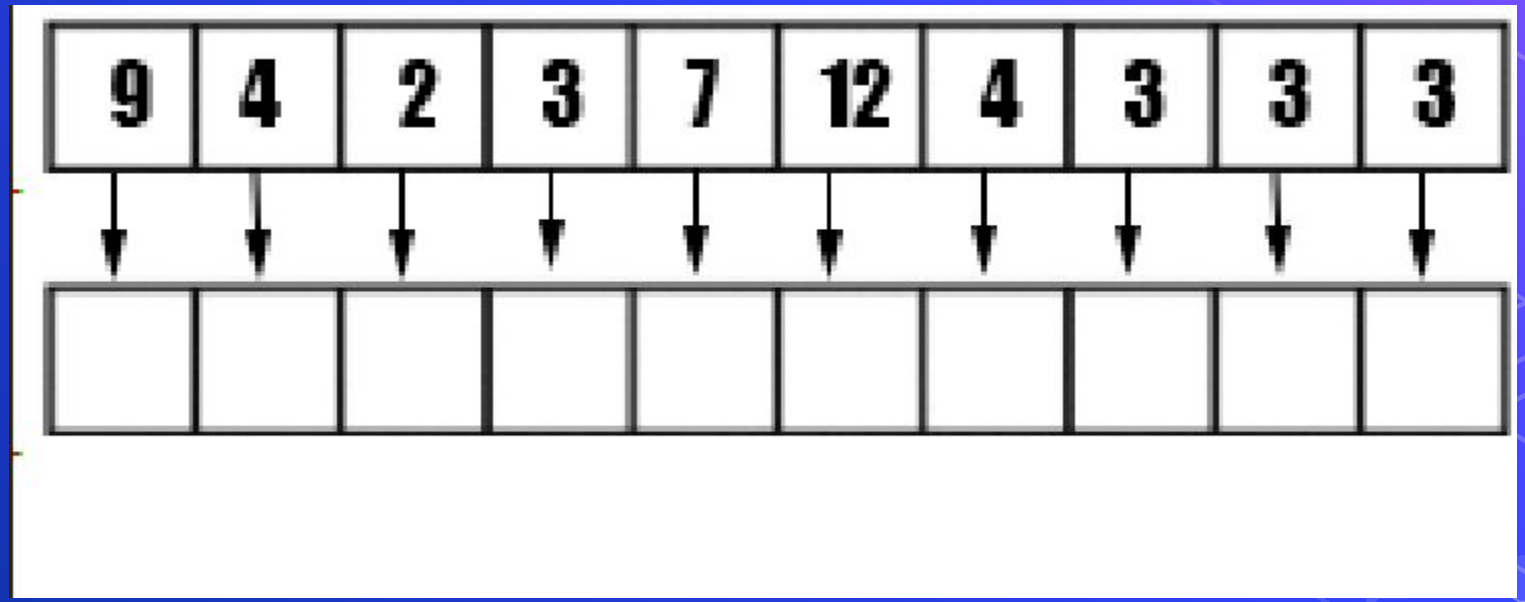
Solved

Medium | Topics | Companies

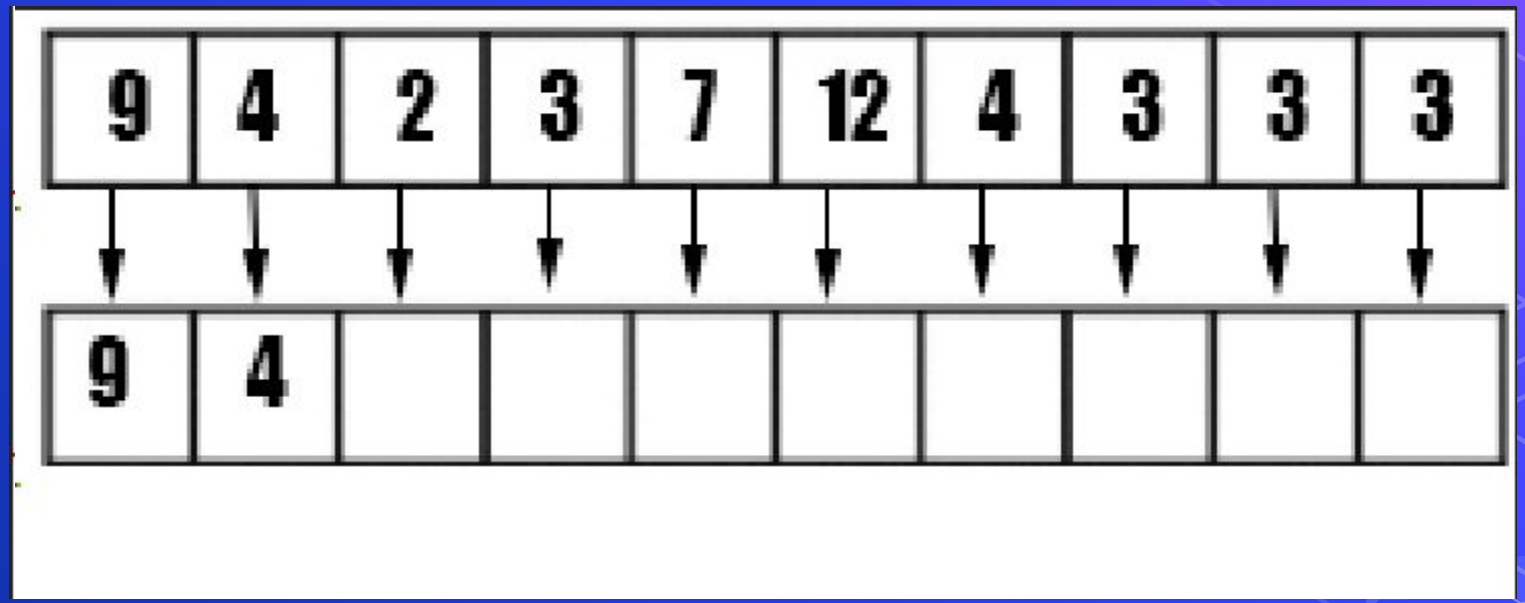
You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

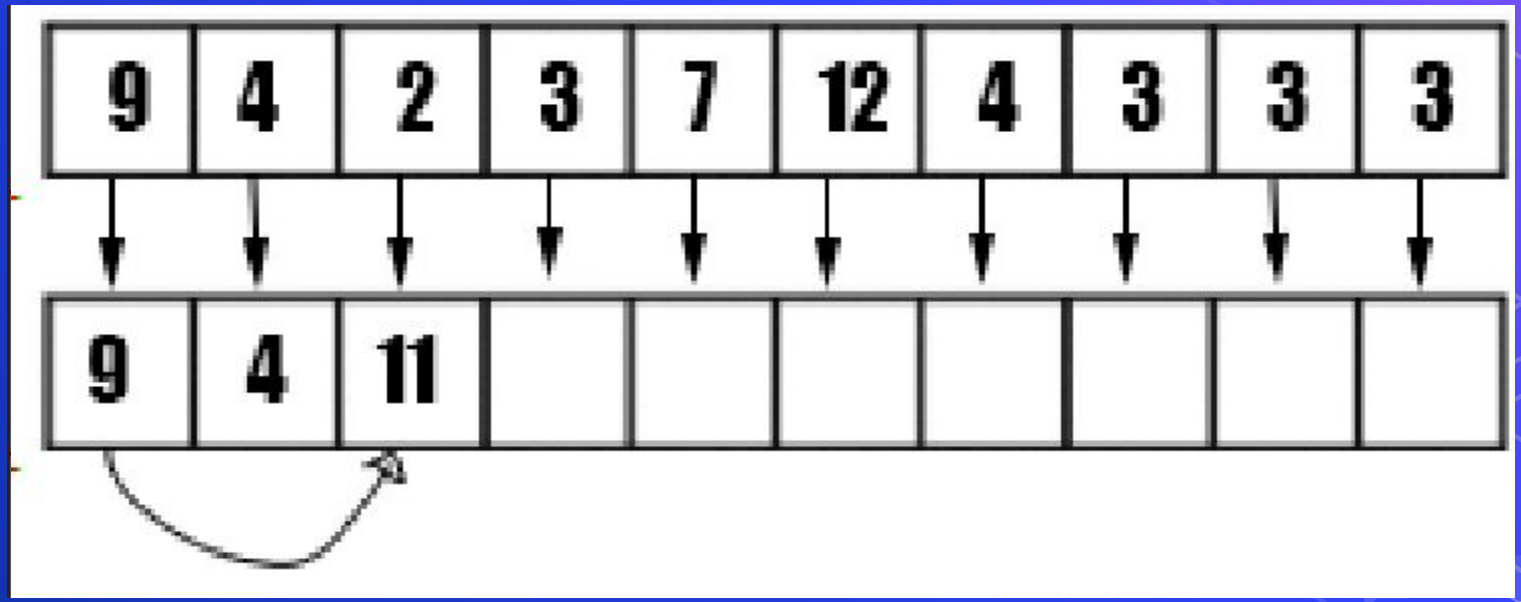
Leetcode 198



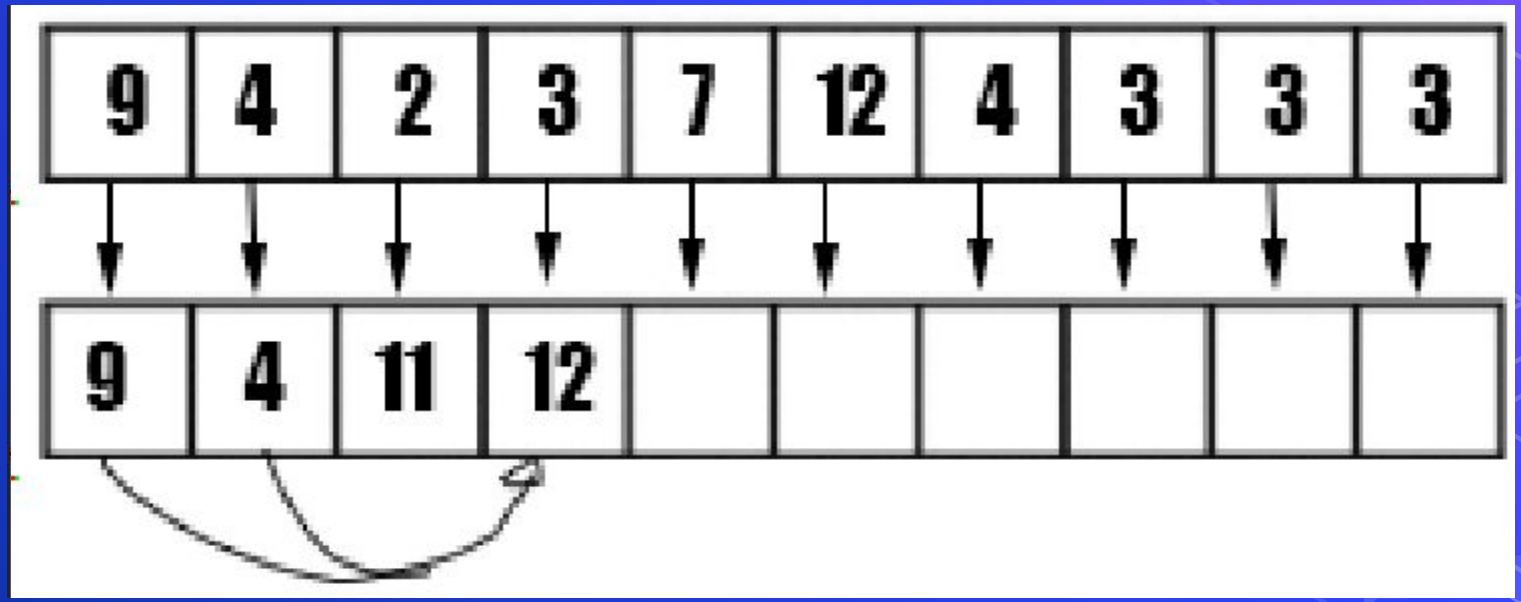
Leetcode 198



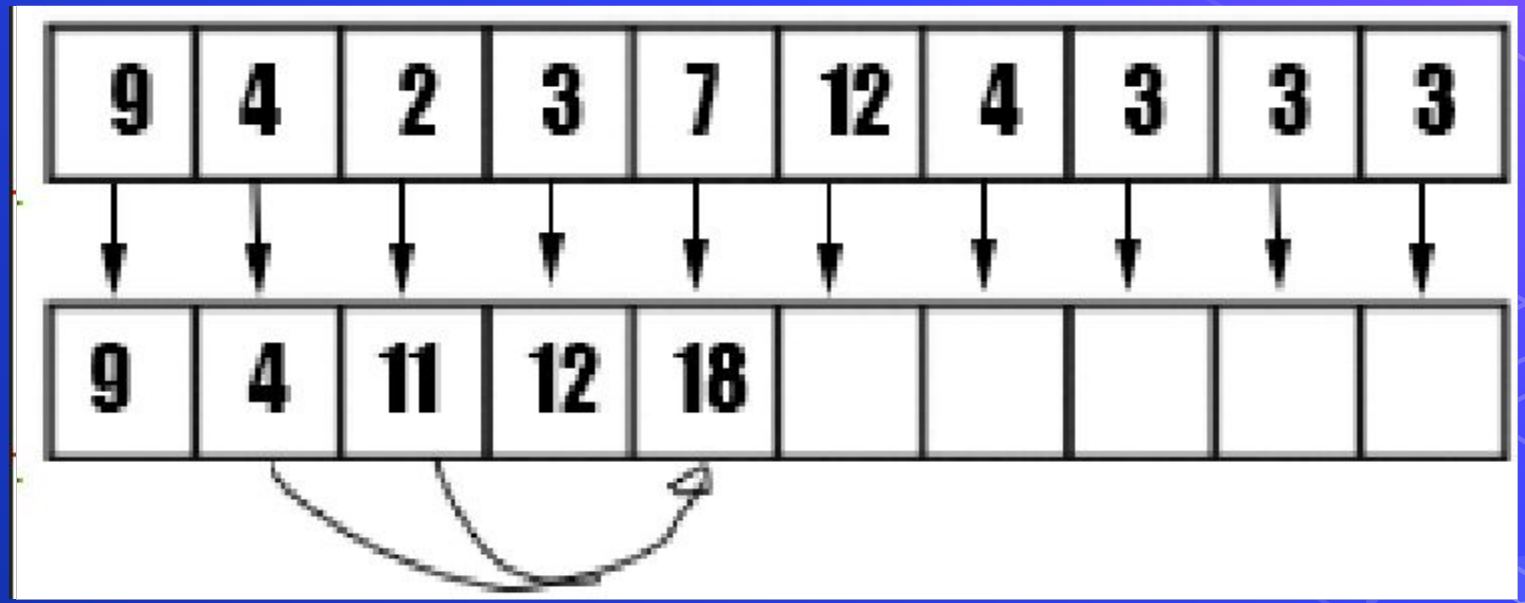
Leetcode 198



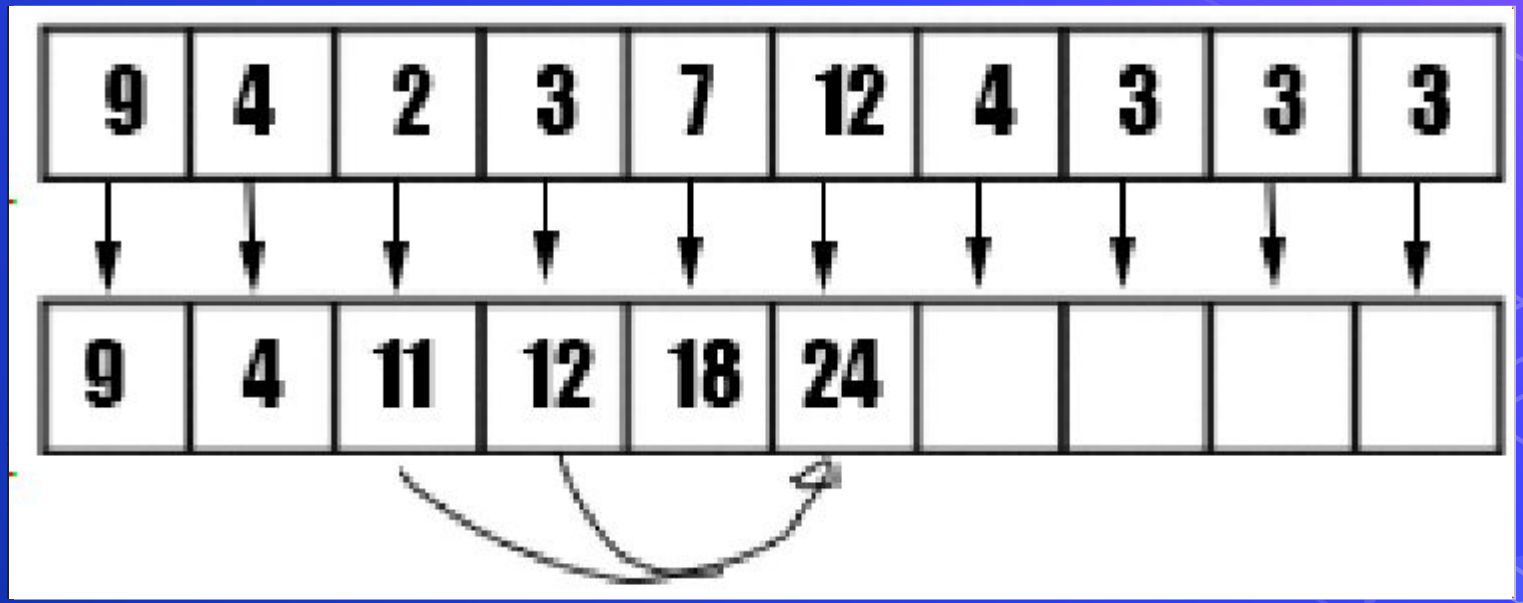
Leetcode 198



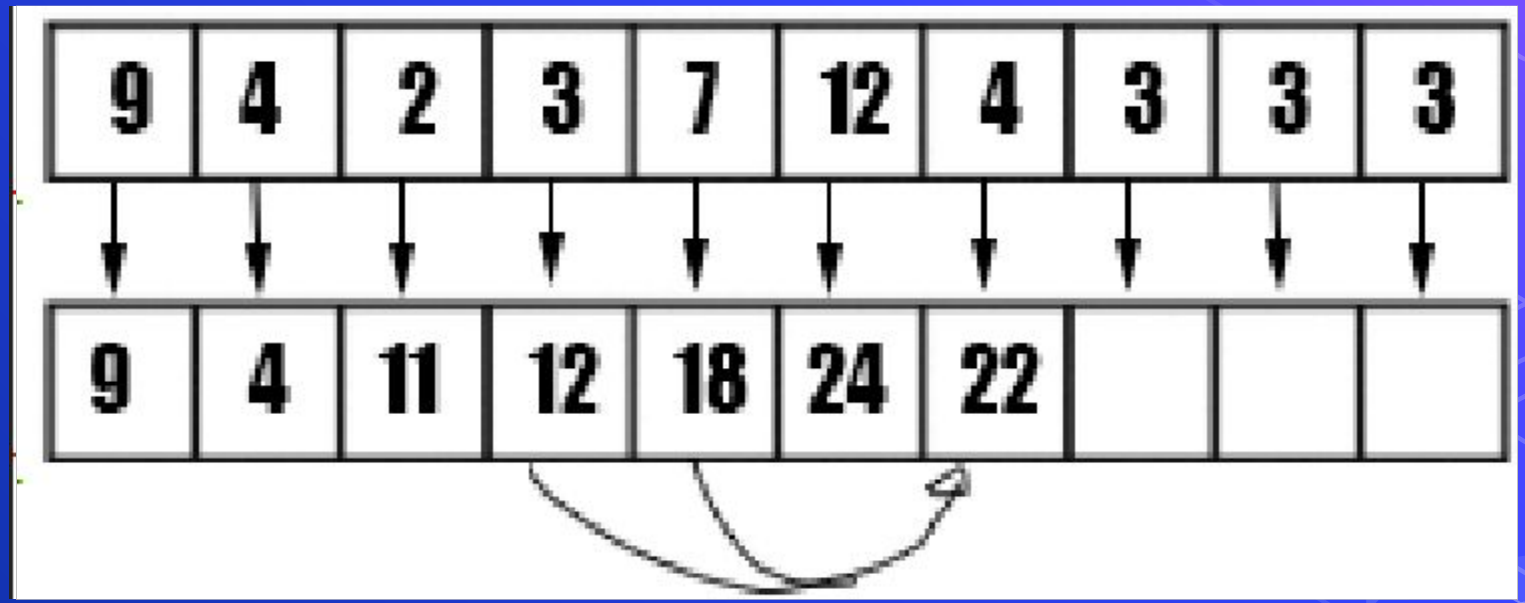
Leetcode 198



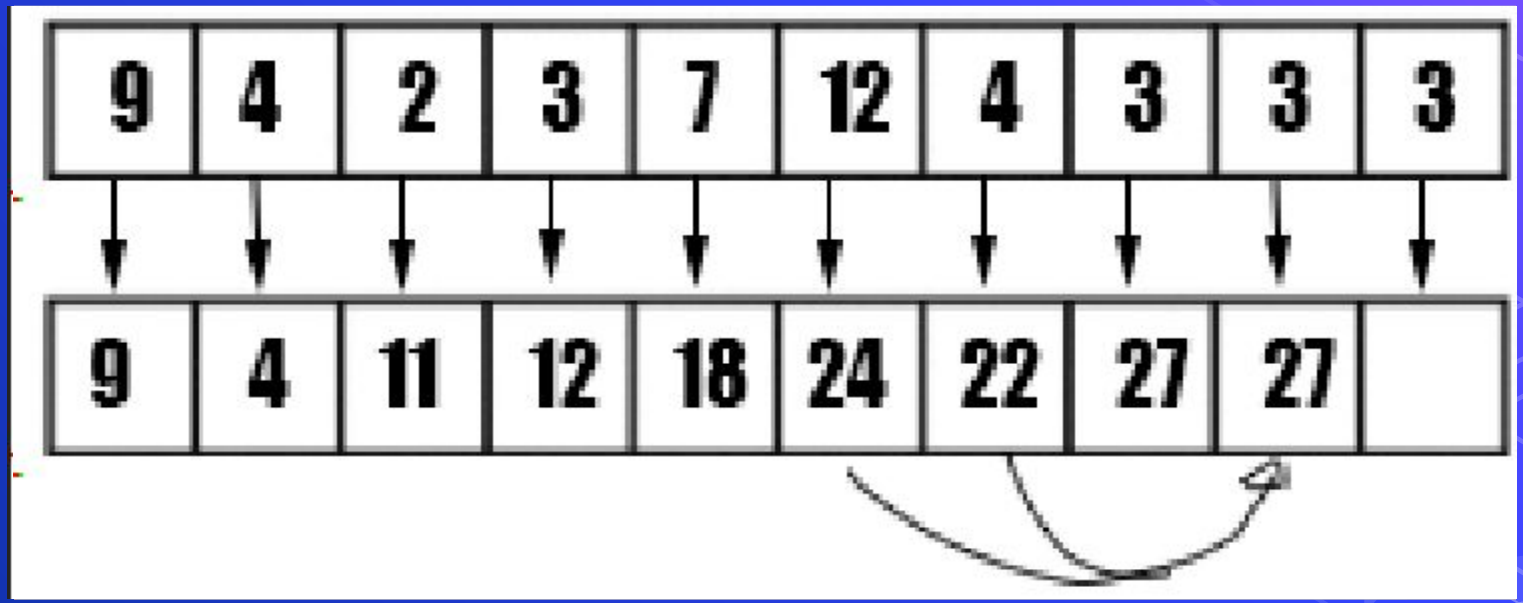
Leetcode 198



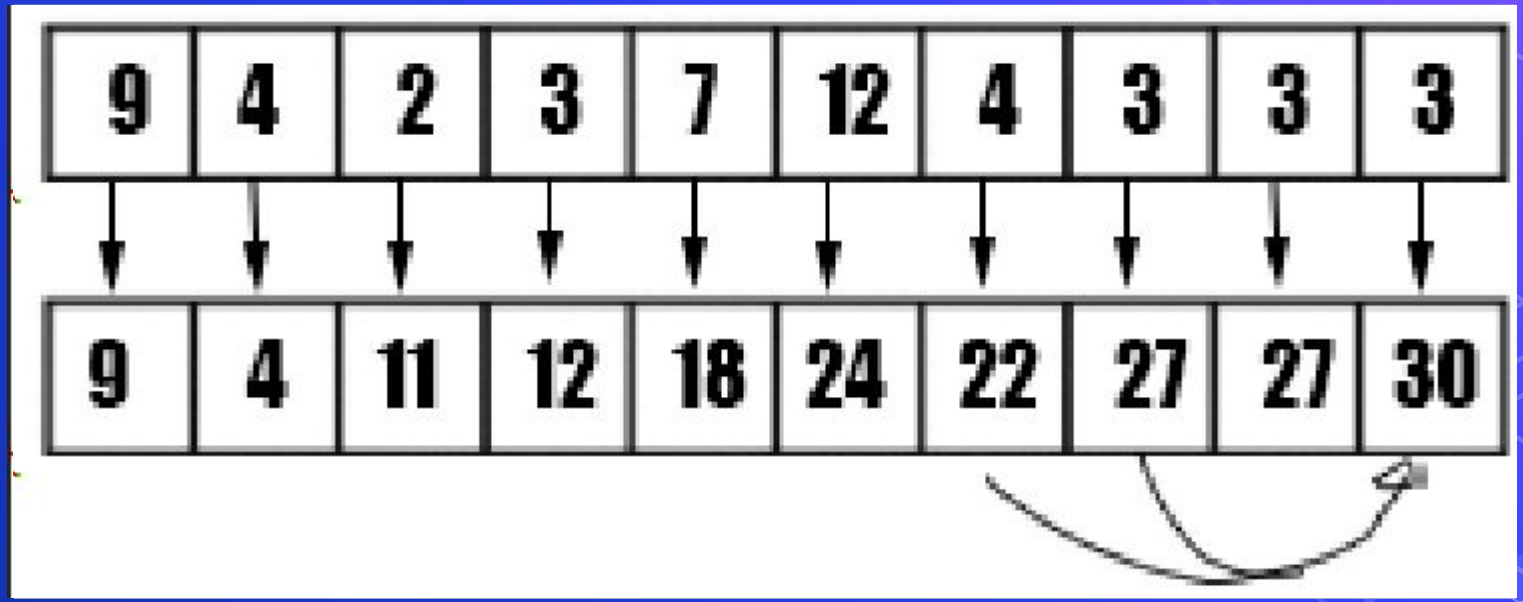
Leetcode 198



Leetcode 198



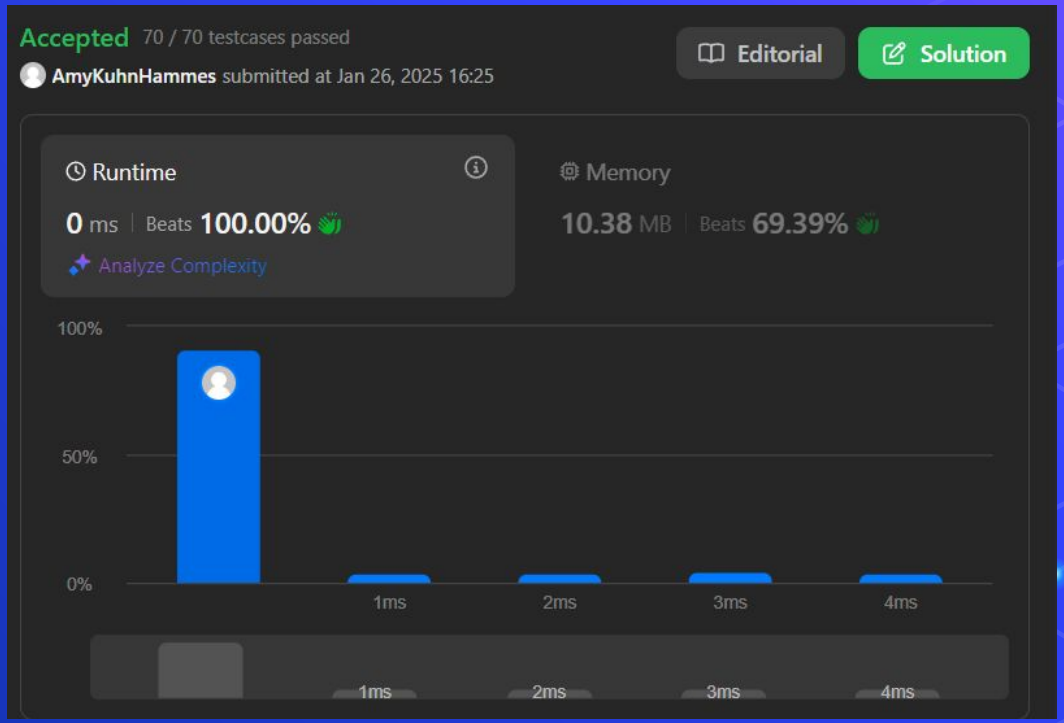
Leetcode 198



Leetcode 198 - Bottom-up

```
int max(int a, int b) {
    if(a > b) return a;
    return b;
}
int rob(vector<int>&nums) {
    if(nums.size() == 1) return nums[0];
    if(nums.size() == 2) return max(nums[0], nums[1]);
    int memo[410] ={nums[0], nums[1], nums[0]+nums[2]}, i;
    for(i = 3; i < nums.size(); i++) {
        memo[i] = max(memo[i-2] + nums[i], memo[i-3] + nums[i]);
    }
    return max(memo[nums.size()-2], memo[nums.size()-1]);
}
```

Leetcode 198 - Bottom-up



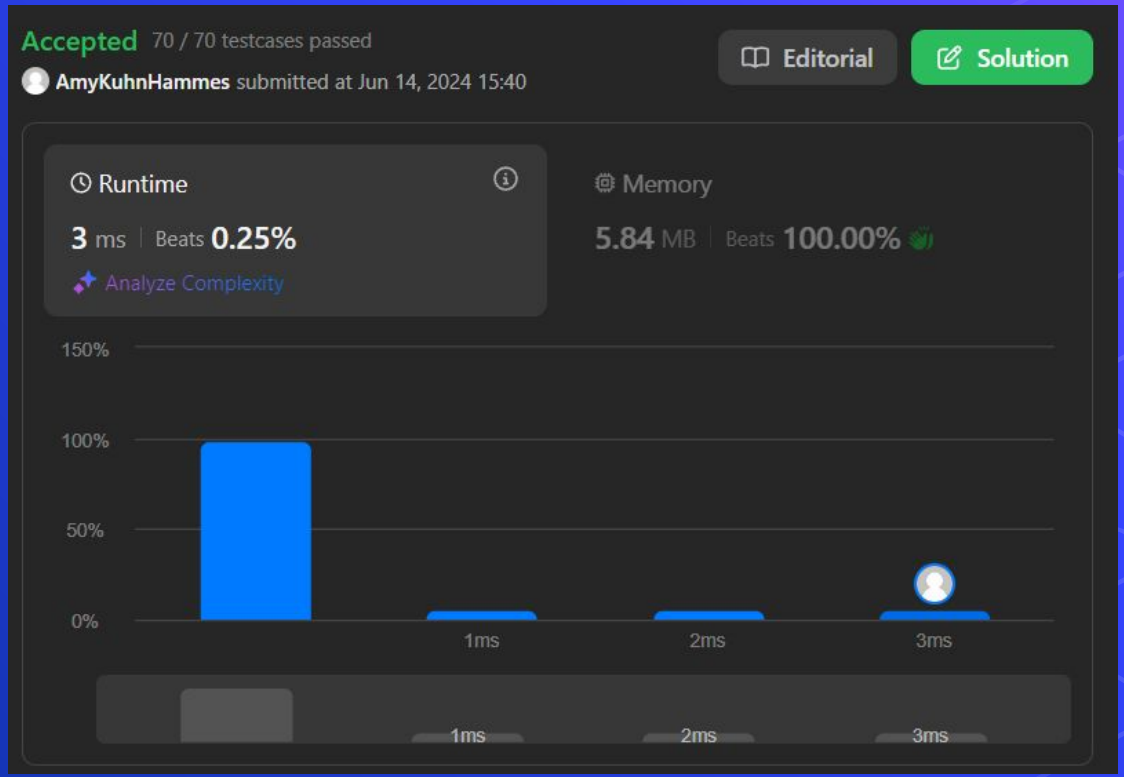
Leetcode 198 - Top-down

```c
int *casas, numCasas, memo[101];

int max(int a, int b) {
    if(a>b) return a;
    return b;
}

int rob(int* nums, int numsSize) {
    casas = nums;
    numCasas = numsSize;
    memset(memo, -1, sizeof(memo));
    return max(rouba(0, 0), rouba(1, 0));
}
```

Leetcode 198 - Top-down

```
int rouba(int atual, int grana) {
    if(atual >= numCasas) return grana;
    if(memo[atual] != -1 && memo[atual] >= grana) return 0;
    memo[atual] = grana;
    return max(
        rouba(atual+2, grana+casas[atual]),
        rouba(atual+3, grana+casas[atual])
    );
}
```

Leetcode 198 - Top-down



Leetcode 322.

322. Coin Change

Solved

Medium | Topics | Companies

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return *the fewest number of coins that you need to make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

Os 4 paradigmas

- Busca completa -> TLE
- Dividir e conquistar -> Não é possível
- Guloso -> Funciona só em casos específicos
- Programação dinâmica
  - Bottom-up -> Forma correta
  - Top-Down -> Funciona, porém, mais lento e mais verboso

1.
Greedy (WA)

(Guloso)

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Greedy

```c
#include <stdio.h>
int main() {
    int i, coins[5] = {50, 25, 10, 5, 1}, aux = 0, amount = 155;
    for(i = 0; i < 5; i++) {
        aux += amount/coins[i];
        amount %= coins[i];
    }
    printf("%d\n", aux);
}
```

```
PS C:\Users\amyuh\Documents\VScode\Coins\output> & .\'greedy.exe'
4
```

Greedy

```c
int compare (const void * a, const void * b) {
  return ( *(int*)b - *(int*)a );
}
int coinChange(int* coins, int coinsSize, int amount) {
    int i, aux = 0;
    qsort(coins, coinsSize, sizeof(int), compare);
    for(i = 0; i < coinsSize; i++) {
        aux += amount/coins[i];
        amount %= coins[i];
    }
    if(amount != 0) return -1;
    return aux;
}
```

Greedy

Wrong Answer 51 / 189 testcases passed

submitted at Jan 31, 2025 17:10

Editorial

Input

Use Testcase

coins =

```
[186,419,83,408]
```

amount =

```
6249
```

Output

```
-1
```

2. Complete search (TLE)

(Força bruta)

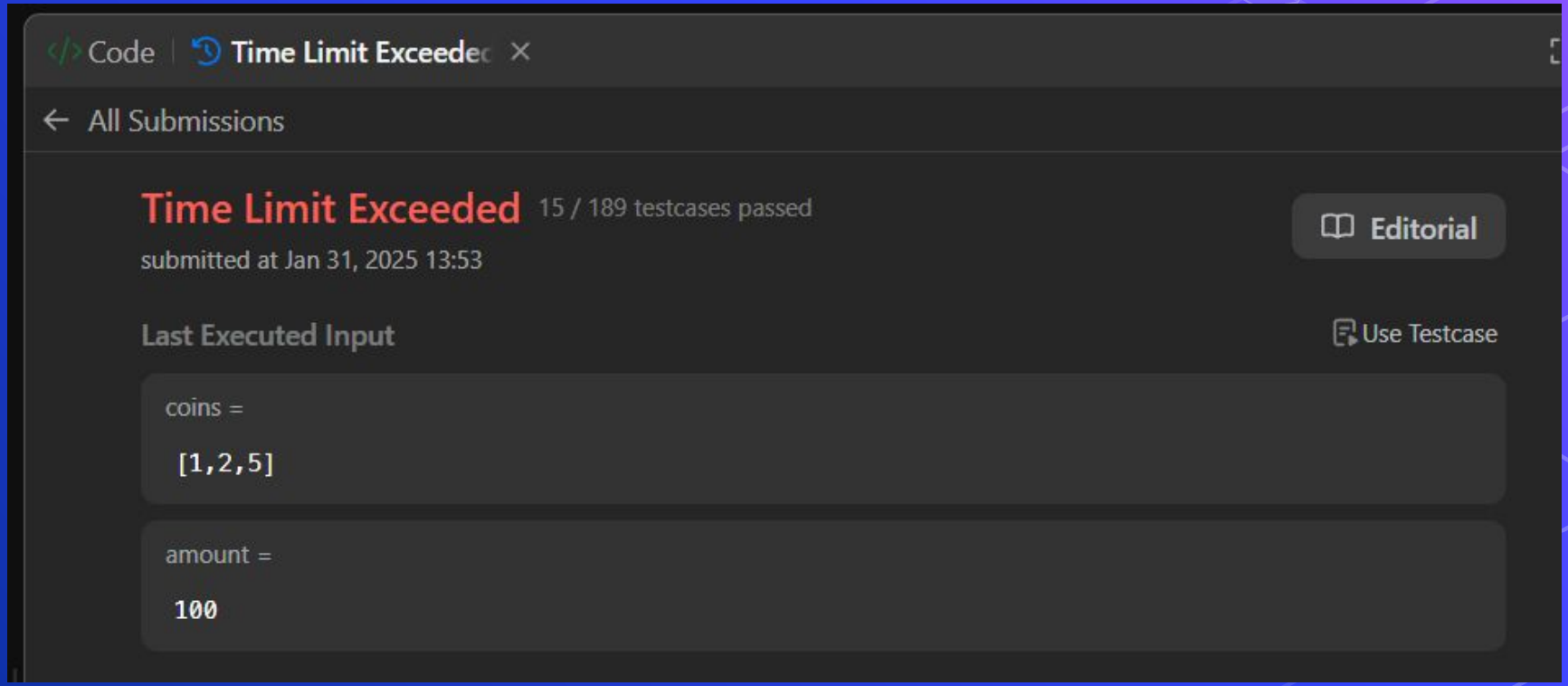
Complete search

```
int min(int a, int b) {
    if(a == -1) return b;
    if(b == -1) return a;
    if(a < b) return a;
    return b;
}
int coinChange(int* coins, int coinsSize, int amount) {
    return change(coins, coinsSize, amount, 0);

}
```

Complete search

```
int change(int* coins, int coinsSize, int amount, int ans) {
    int i, aux = -1;
    if(amount == 0) return ans;
    if(amount < 0) return aux;
    for(i = 0; i < coinsSize; i++) {
        aux = min(aux, change(coins, coinsSize, amount-coins[i], ans+1));
    }
    return aux;
}
```

Complete search



3.
DP - Top-down (AC)

(Programação dinâmica)

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

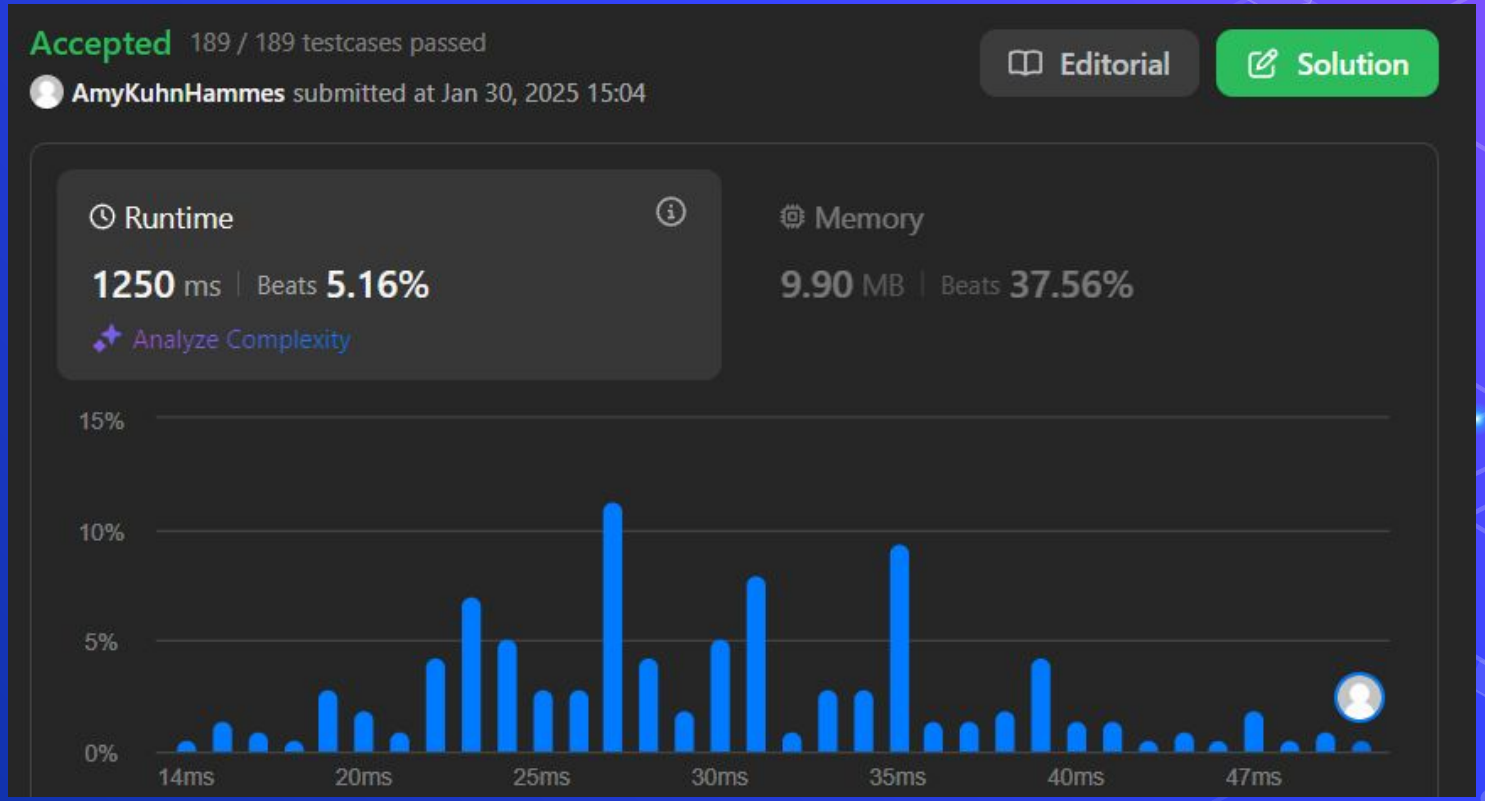
DP - Top-down

```
int min(int a, int b) {
    if(a == -1) return b;
    if(b == -1) return a;
    if(a < b) return a;
    return b;
}

int coinChange(int* coins, int coinsSize, int amount) {
    int memo[10001] = {0};
    return change(memo, coins, coinsSize, amount, 0);
}
```

DP - Top-down

```c
int change(int* memo, int* coins, int coinsSize, int amount, int ans) {
    int i, aux = -1;
    if(amount == 0) return ans;
    if(amount < 0) return aux;
    if(memo[amount] != 0 && memo[amount] <= ans) return aux;
    memo[amount] = ans;
    for(i = 0; i < coinsSize; i++) {
        aux = min(aux, change(memo, coins, coinsSize, amount-coins[i], ans+1));
    }
    return aux;
}
```

DP - Top-down


Accepted 189 / 189 testcases passed
AmyKuhnHammes submitted at Jan 30, 2025 15:04
Editorial
Solution
Runtime
1250 ms | Beats 5.16%
Analyze Complexity
Memory
9.90 MB | Beats 37.56%
15%
10%
5%
0%
14ms
20ms
25ms
30ms
35ms
40ms
47ms

4.
DP - Bottom-up (AC)

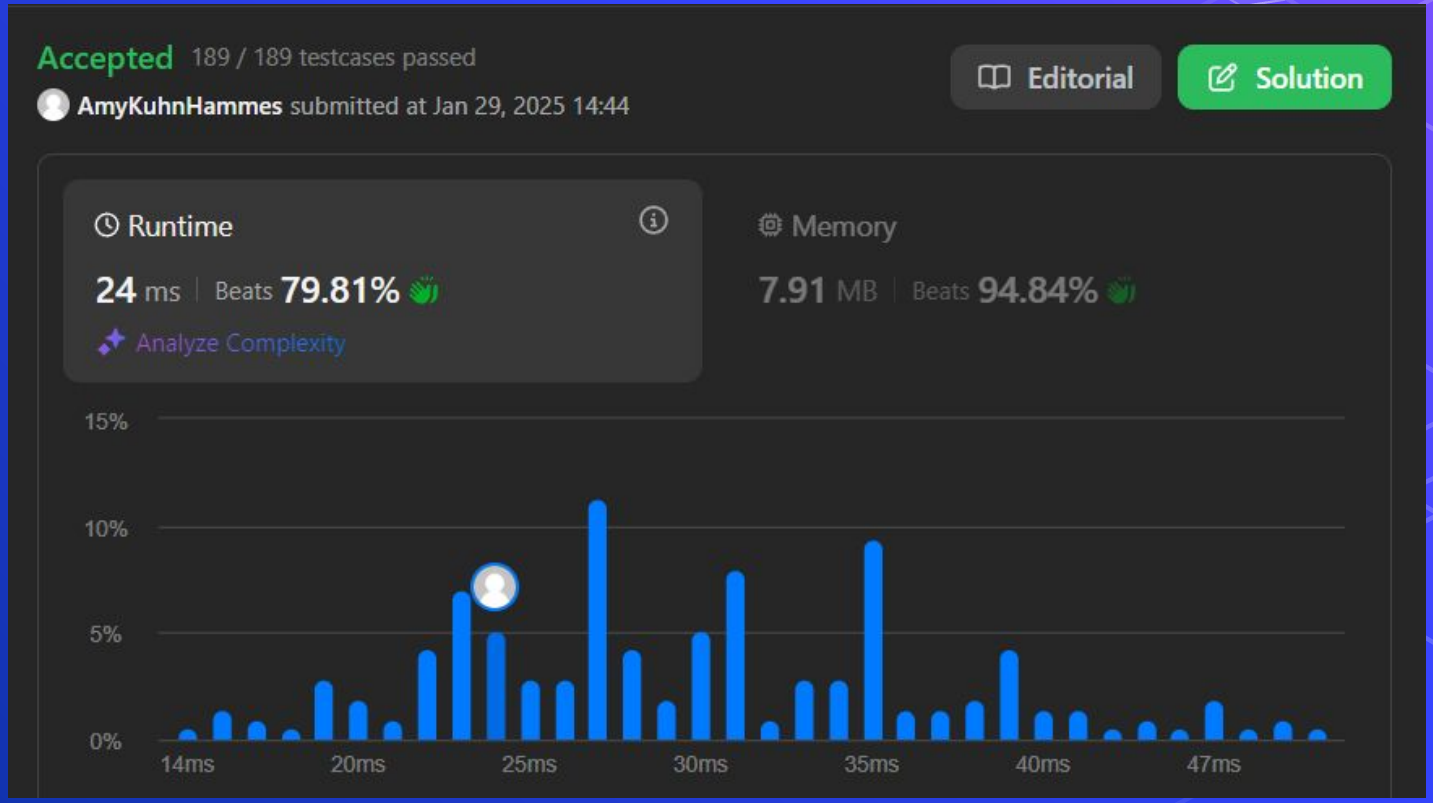
(Programação dinâmica)

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

DP - Bottom-up

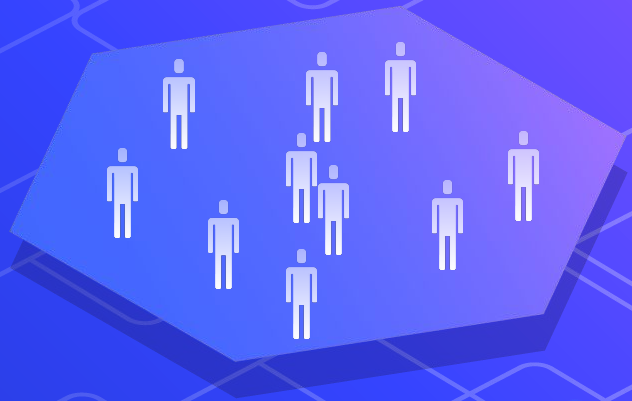
```
int coinChange(int* coins, int coinsSize, int amount) {
    long memo[10001] = {0}, i, j;
    memset(memo+1, -1, sizeof(memo)- sizeof(long));
    for(i = 0; i < amount; i++) {
        for(j = 0; j < coinsSize; j++) {
            if(i+coins[j] <= amount && memo[i] != -1 &&
            (memo[i+coins[j]] > memo[i]+1 || memo[i+coins[j]] == -1)) {
                memo[i+coins[j]] = memo[i]+1;
            }
        }
    }
    return memo[amount];
}
```

DP - Bottom-up


Accepted 189 / 189 testcases passed
AmyKuhnHammes submitted at Jan 29, 2025 14:44
Editorial
Solution
Runtime
24 ms | Beats 79.81%
Analyze Complexity
Memory
7.91 MB | Beats 94.84%
15%
10%
5%
0%
14ms
20ms
25ms
30ms
35ms
40ms
47ms

Abandone o trivial, comece a programar paralelo!

Hora da prática!



PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL

Obrigada!

Alguma pergunta?

Me mande um email!

amy@inf.ufpel.edu.br

fbviegas@inf.ufpel.edu.br

PET COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE PELOTAS - UFPEL