Upsolving - 7 puentes

David Betancourt Montellano
Onder Francisco Campos Garcia

Ordenamiento súper básico - David

Un simple ordenamiento

+ Considerar el caso donde n = 0.
+ Usar un diccionario o map. Para ahorrar memoria. Ejemplo:

Número entero	Cantidad de veces que se repite
21	2
42	1

Cifrando Mensajes

Analizando el problema:

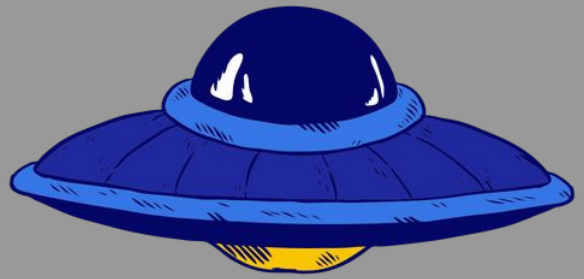
Pepito quiere comunicarse con seres de otro mundo, pero para que el gobierno no se entere necesita cifrarlo,Tu misión es ayudarle a pepito a hacer un programa que cuando encuentre palabras entre paréntesis escriba la palabra al revés y quitar los paréntesis.

Input:

1. Un string

Salida:

El string cifrado

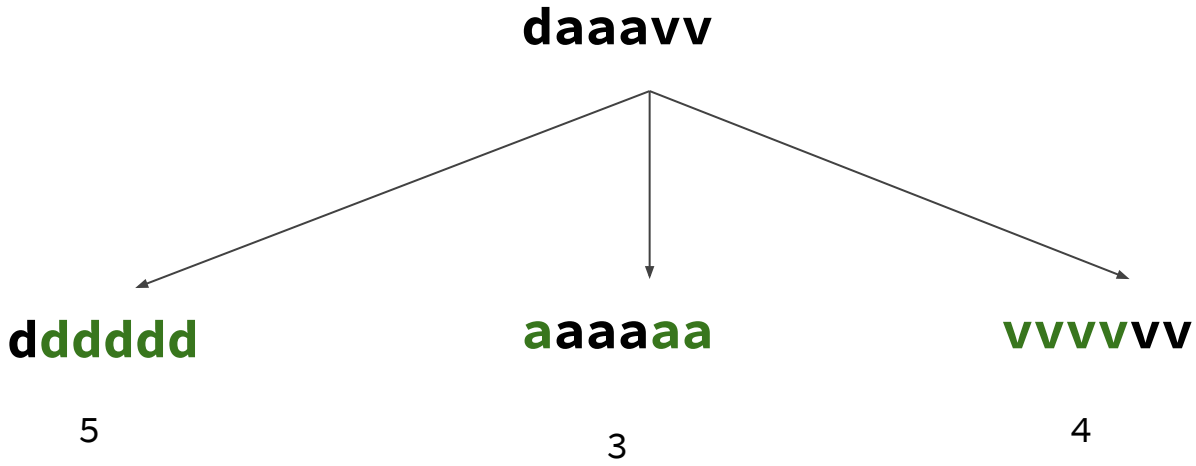


El problema es muy fácil, consiste en buscar substrings que están entre paréntesis, y a cada uno de ellos quitar los paréntesis y poner al revés el substring.

CR Cadenas Suaves - David

Cadenas suaves (si todos sus caracteres son iguales)

Hecho: X de {a-z} y Y de {a-z}. Al convertir X != Y.



Usar una tabla hash, f(x)=x. Y obtener el máximo, y |cadena| - máximo.

Just Eat It!

Analizando el problema:

El problema nos dice que Yasser y Adel están en una tienda de cupcakes,Yasser quiere probar todos los n pasteles, cada pastel tiene asignado un valor del sabor, Adel por su parte solo elegirá una parte de todos los pasteles un intervalo [l,r] donde l > 0, r < n.

Input:

1. Un entero m que representa el número de casos que se analizaran
2. Para cada caso se pide dos líneas.
3. La primera línea es un entero n el cual representa el número de pasteles
4. La segunda línea cuenta con n enteros representando el valor de cada pastel

Salida:

Imprimir “YES“ si la suma de los pasteles que probó Yasser es estrictamente mayor que los de Adel, de lo contrario imprimir “NO”.

El problema se puede resumir saber si suma total de todos los pasteles es mayor que el máximo subarreglo [l,r] donde l > 0, r < n

Noche de cine

Analizando el problema:

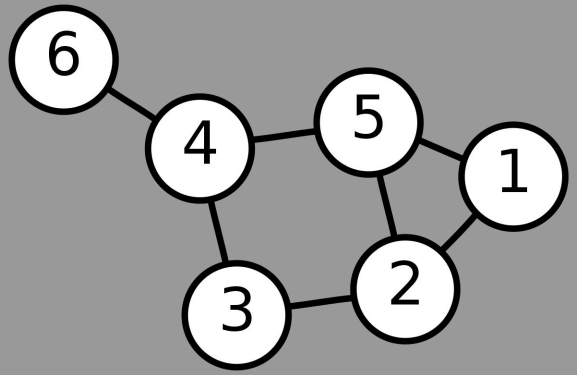
El problema nos dice que unos chicos acaban de salir de un película de una sala y quieren visitar las demás salas las cuales están organizadas de una forma específica donde cada sala tiene m salas como vecinas.

Input:

1. El número de salas(n) la sala inicial(inicio)
2. n líneas, en cada línea se indica el número de la sala que se dará la especificación de las salas que tiene de vecinas, el número de vecinos y los vecinos (formato) n_i m m_1..m_2..m_m.

Salida:

El orden en que los chicos visitaran todas las salas iniciando en la sala en la que estaban.

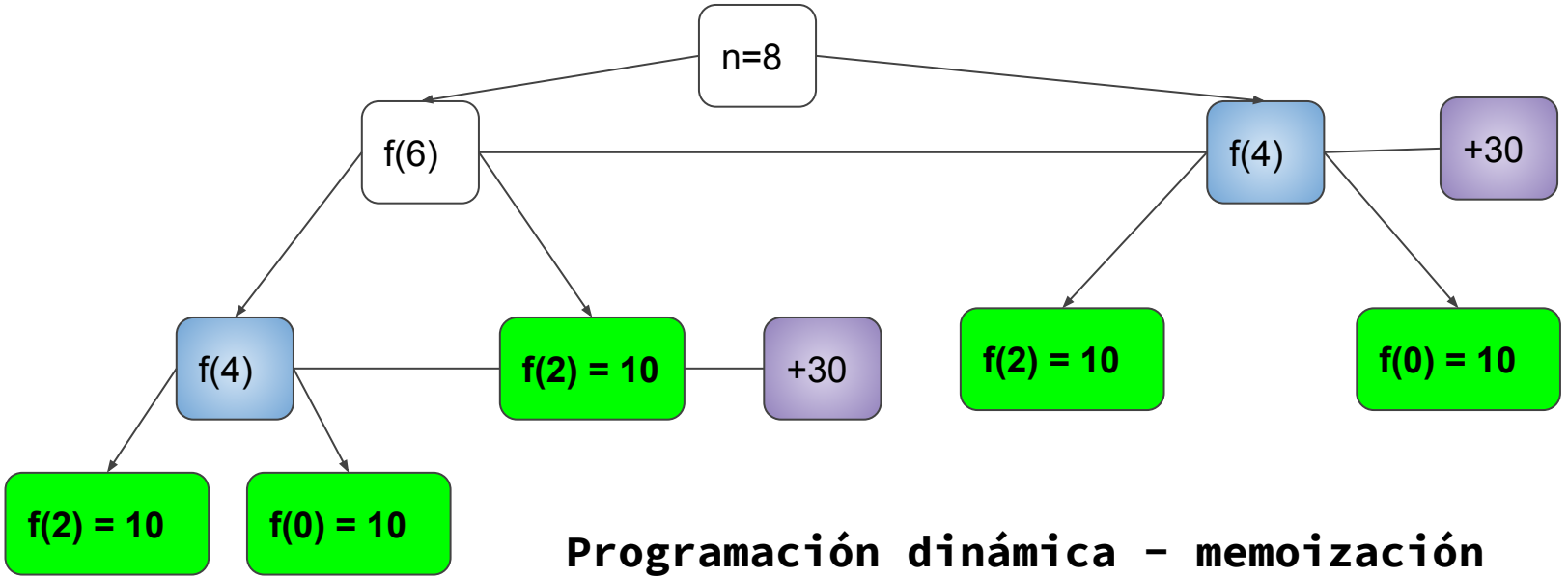


Recorrer el grafo desde el inicio hasta recorrer todas las salas, en teoría de grafos existen dos formas de recorrer un grafo **dfs** y **bfs** el problema no da muchas pistas de cuál debe usarse(Comportamiento) pero se acerca más a la forma en la que se recorre con **bfs.**

Fórmula recursiva ocho - David

Fórmula recursiva ocho

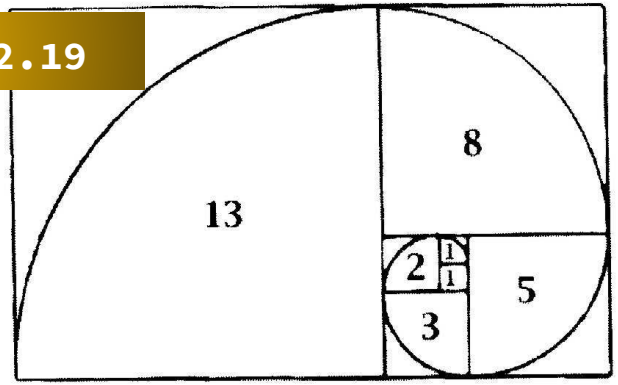
$$f(n) = \begin{cases} 10 & \text{para toda n menor 4} \\ f(n-2) + f(n-4) + 30 & \text{para toda n mayor o igual a 4} \end{cases}$$



Programación dinámica - memoización

Posición Fibonacci - David

`ln_1.618(2^64) = 92.19`



Posición Fibonacci

Dado un número n ∈ N, n < 2^64.

Encontrar posición que ocupa en la sucesión de Fibonacci, y si no es elemento de la sucesión, imprimir -1.

> Si n == 1, **entonces a respuesta es 1.**
> De otra forma: Contar la generación de los elementos hasta que fn < n, **inicialmente contador = 2.**

- Si fn == n, entonces **la respuesta es el contador.**
- **La respuesta es -1.**

El Gato de Springfield

Analizando el problema:

El problema dice que el ladrón el “gato” quiere robar el “zircón” el cual está en el museo de Springfield el cual cuenta con n salas y m pasillos, el gato empieza en la sala 1 y el zircon se encuentra en la sala n,Homero Simpson sabe que el gato siempre roba usando el camino más corto por lo que Homero puso seguridad en todos esos pasillos.El gato sabe eso por lo que se pregunta que si puede llegar al zircon sin usar esos caminos si si cuanto tiempo le llevará.

Input:

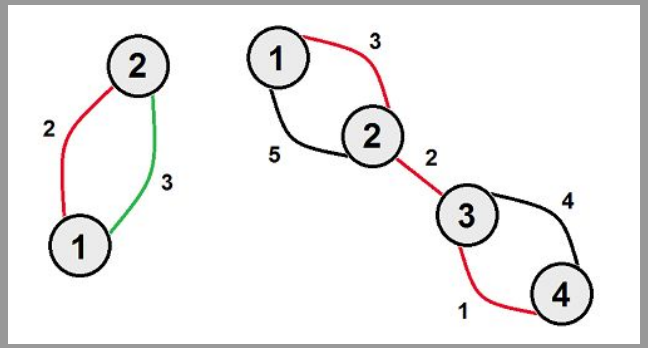
- Hay muchos casos de prueba
- La primera línea dos enteros el primero n el número de salas el segundo m el número de pasillos.
- m líneas con tres enteros fuente destino costo, la fuente es de que sala parte, destino a cual se dirige y el costo es cuanto tiempo cuesta ir de fuente al destino.
- Mientras que n y m sean diferentes de 0.

Salida:

Si el gato puede robar el zircon con la restricción antes expuesta imprimer el tiempo que se tardó de lo contrario imprimir “Do'oh!”

El problema consiste en encontrar el camino más corto que no es el camino más corto.

- Encontrar el camino más corto.
- Eliminarlo / Etiquetarlo
- Encontrar el camino más corto sin el camino antes encontrado.
- Si no se puede llegar a la sala n imprimir “Do'oh!” de lo contrario imprimir el valor del camino resultante



- Dijkstra's algorithm
- Bellman-Ford algorithm
- A* search algorithm
- Floyd-Warshall algorithm
- Johnson's algorithm Floyd-Warshall
- Viterbi algorithm
- DP

Compuertas Lógicas - David

Compuertas lógicas

+ Idea básica, resolver de adentro hacia afuera, i.e. resolver compuertas de la forma: XOR(TRUE,TRUE), para así poder resolver: XOR(NOT(FALSE), AND(TRUE, FALSE)).

+ Antes de ir resolviendo corregir el balance de paréntesis: **“No todas las líneas tienen la cantidad correcta de paréntesis. Deberán verificar y corregirlas si este es el caso”**.

+ **Usar eval() de Python** o simular un eval() en C++.

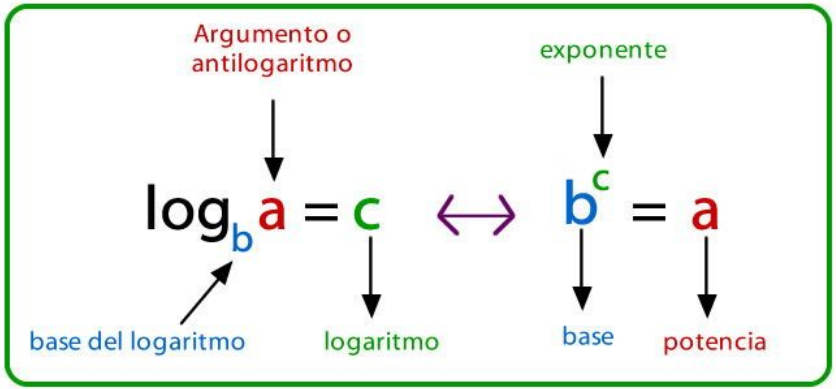
```
def f_creator():
    expr = stdin.readline()
    TRUE = True
    true = True
    false = False
    FALSE = False
    try:
        y = eval(expr)
    except Exception as e:
        y = False
    if(y):
        print("TRUE")
    else:
        print("FALSE")

if __name__ == '__main__':
    f_creator()
```

```
from math import *
from sys import stdin, stdout

def NOT(x):
    return not(x)

def AND(x, y):
    return x and y

def OR(x, y):
    return x or y

def NAND(x, y):
    return (not(x and y))

def NOR(x, y):
    return (not(x or y))

```

Factorial Enorme - David

Factorial enorme

Entrada: 1 <= n <= 2,000,000



¿Cuántos dígitos tendrá el producto: 3*8?

Idea base: Nuestro sistema decimal es posicional (<-), cada posición es una potencia de 10.
(3 * 10^0) * (8 * 10^0) = 10^0 * (24) = 24.
En ese producto, cada dígito “aporta” potencialmente entre [0,1) dígito(s). log_10(3) + log_10(8) = log_10(3*8) = 1.38

Propiedad de logaritmos

ceil(1.38) = 2

Memorizando cadenas con poca memoria

Buddy el buscador

Analizando el problema:

En este problema debemos encontrar el conjunto de números enteros positivos **C** tal que todo elemento del conjunto es menor o igual a **2n** y que **2 no** divide al **mínimo común múltiplo** de cualesquiera **2 números** del conjunto.

Input:

- Un único entero n.

Salida:

- La sumatoria de todos los elementos del conjunto.

Límite:

- Para para un conjunto agrupado del 30% de los casos

$$2 \leq n \leq 45 \times 10^3$$

- Para para un conjunto agrupado del 70% de los casos

$$45 \times 10^3 \leq n \leq 10^9$$

Análisis de cómo atacar
Preguntas clave:

- ¿Cuál es ese grupo C?
  - ¿Hasta dónde va 2*n?
  - ¿Cuál es el mínimo común múltiplo?

Fuerza bruta:

- **Encontrar el grupo C**
- **Sumarlo**

Ejemplo:

n = 3
2*n = 1,2,3,4,5,6.
C = {1,3,5}

Observaciones:

- Va hasta el doble de n.
- El 2 no divide al grupo.

Optimización:

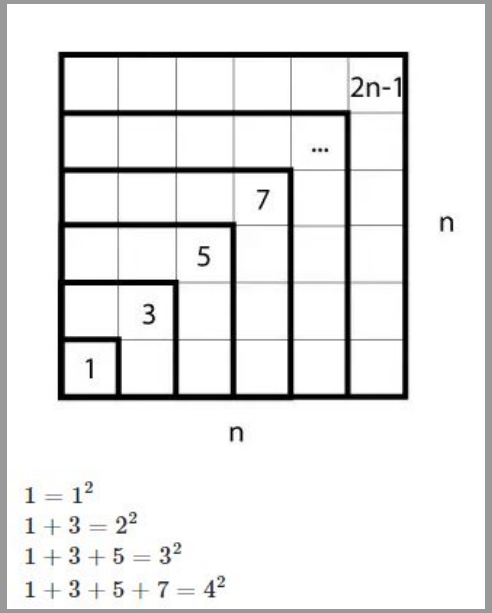
- **Veamos otros ejemplos:**

n = 4
2*n = 1,2,3,4,5,6,7,8.
C = {1,3,5,7}.

n = 5
2*n = 1,2,3,4,5,6,7,8,9,10.
C = {1,3,5,7,9}.

- parece existir un patrón...

∑=1+3+5...n



$1 = 1^2$
$1 + 3 = 2^2$
$1 + 3 + 5 = 3^2$
$1 + 3 + 5 + 7 = 4^2$