15-10-2021

Bakantracking

Backtracking Goes Brrrr

cparedesr@uc.cl - matamalaappels@uc.cl

Backtracking 101

- Si un problema tiene ciertas condiciones y restricciones. Es posible de resolver con métodos iterativos
- Supongamos un problema super simple, tenemos 3 listas de números A, B y C. y queremos determinar si dado un numero x si lo siguiente se cumple

$$a \in A, b \in B, c \in C \quad a + b + c = x$$

Backtracking 101

La siguiente solución iterativa seria valida...

```
for a in A
    for b in B
        for c in C
            return true if x==a+b+c
```

Backtracking 101

Pero muy muy ineficiente, la complejidad base es $\mathcal{O}(n^3)$

```
for a in A
    for b in B
        for c in C
            return true if x==a+b+c
```

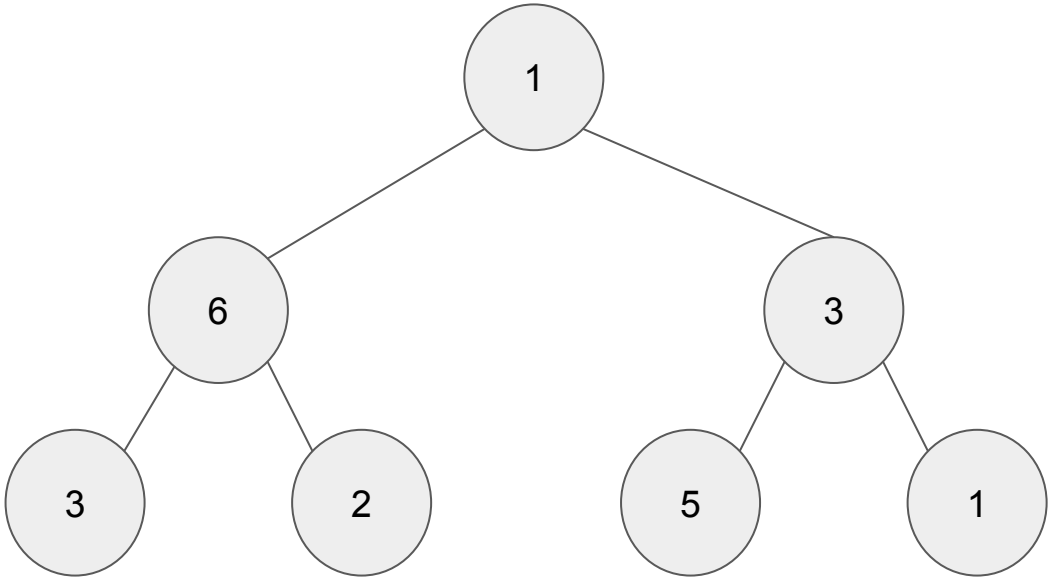
Backtracking 101

- Backtracking es la idea de poder **ir hacia atrás** esto es por ejemplo ser capaces de detectar que **a+b** en cierto momento ya es mayor a **x**, lo que implica que no existe c que cumpla con las condiciones.
  En dicho momento el Algoritmo *Se da cuenta* y puede retroceder y continuar con los siguientes casos válidos.

- A esto se le llama contradicción, cuando es *imposible* que cualquier rama bajo el estado actual cumpla con las condiciones

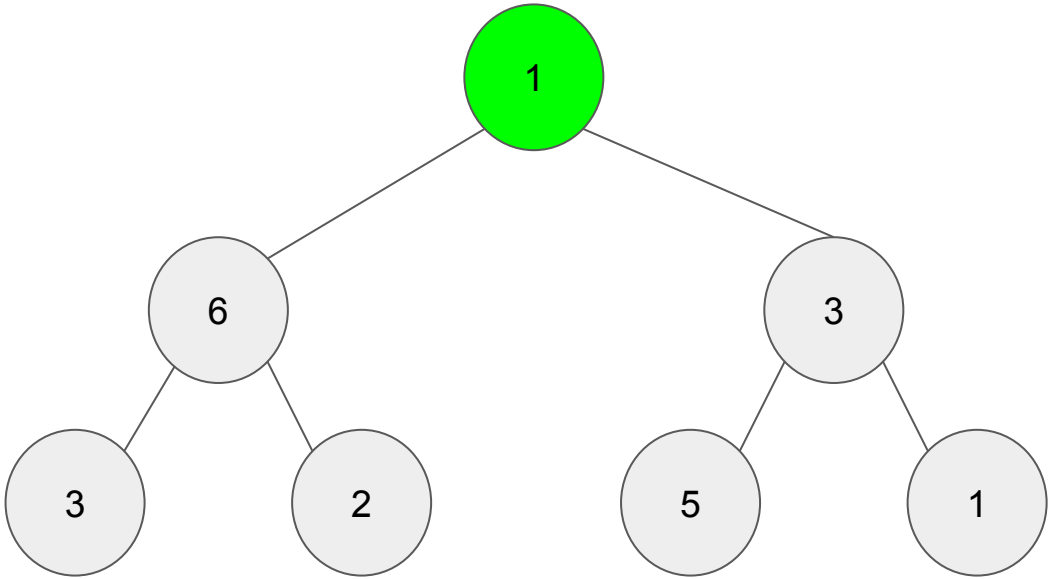
Backtracking 101

- Veamos como Ejemplo X = 5



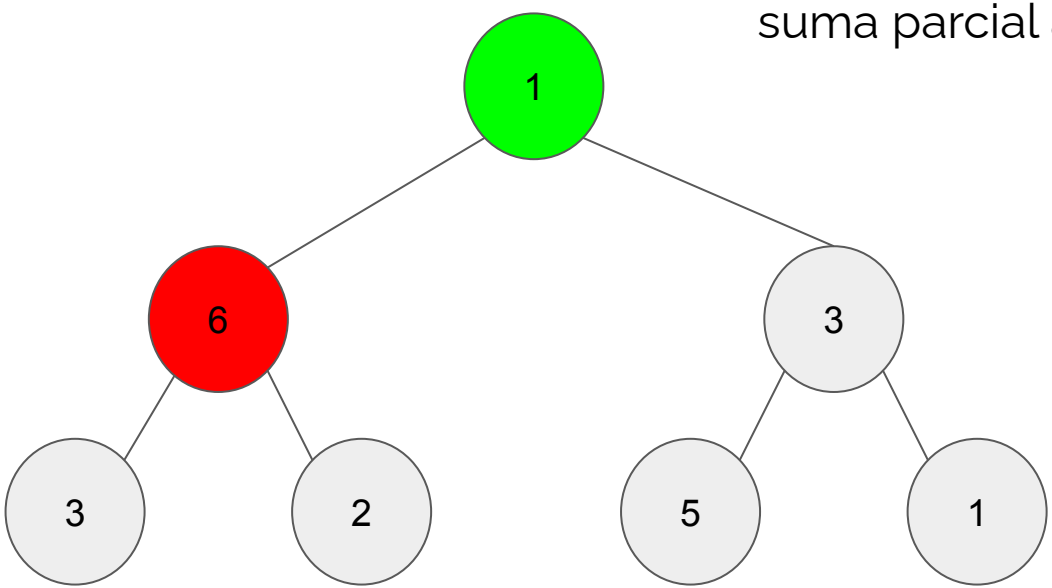
Backtracking 101

- Primera iteracion, a=1

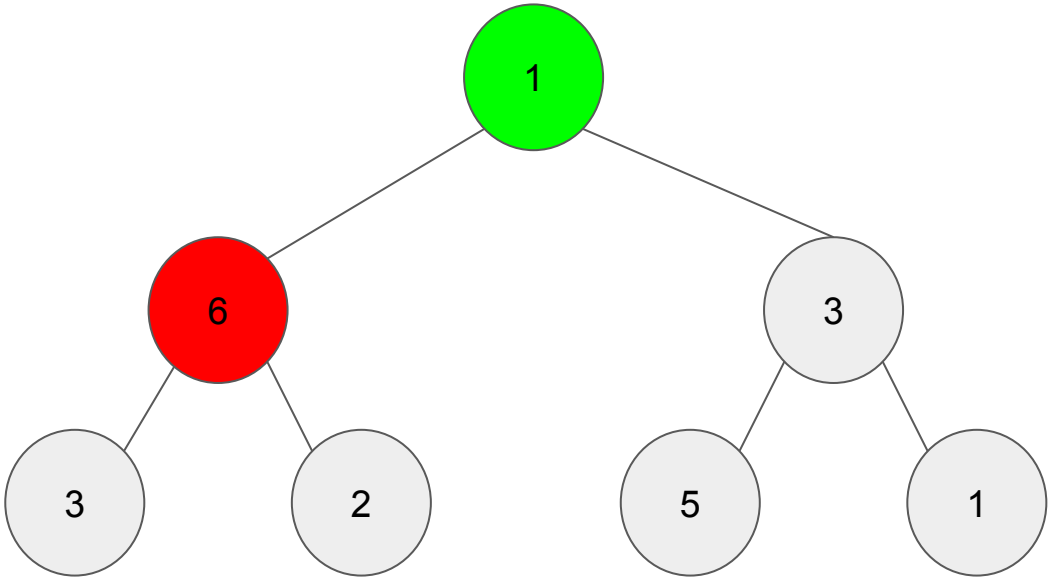


Backtracking 101

- Dentro del for, b=6, por ende suma parcial aux=6+1=7

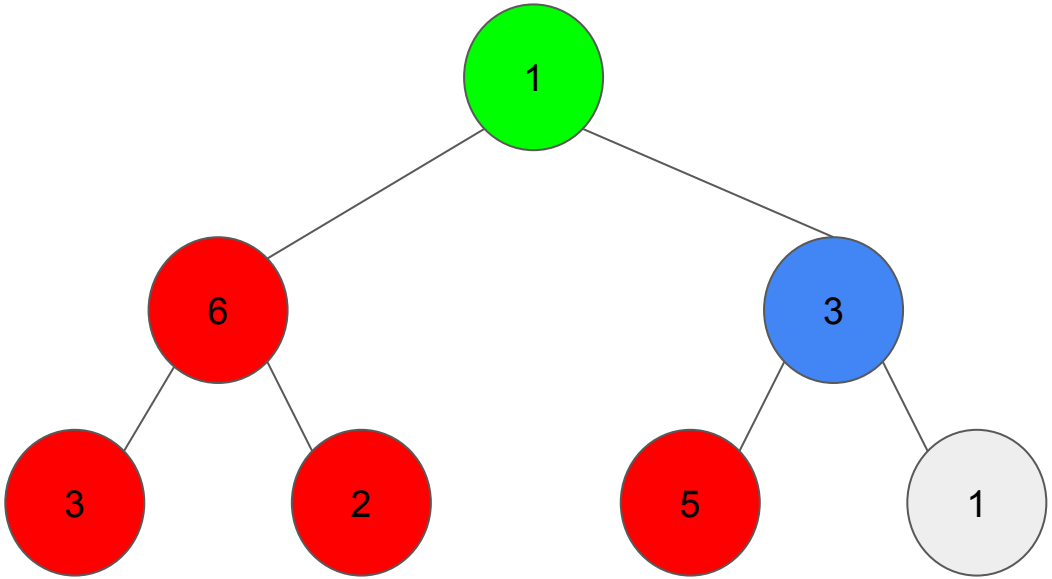


Backtracking 101



Pero observamos que independiente de nuestra elección de **c**, no obtendremos solución a nuestro problema, por lo que nos retratamos (hacemos backtrack)

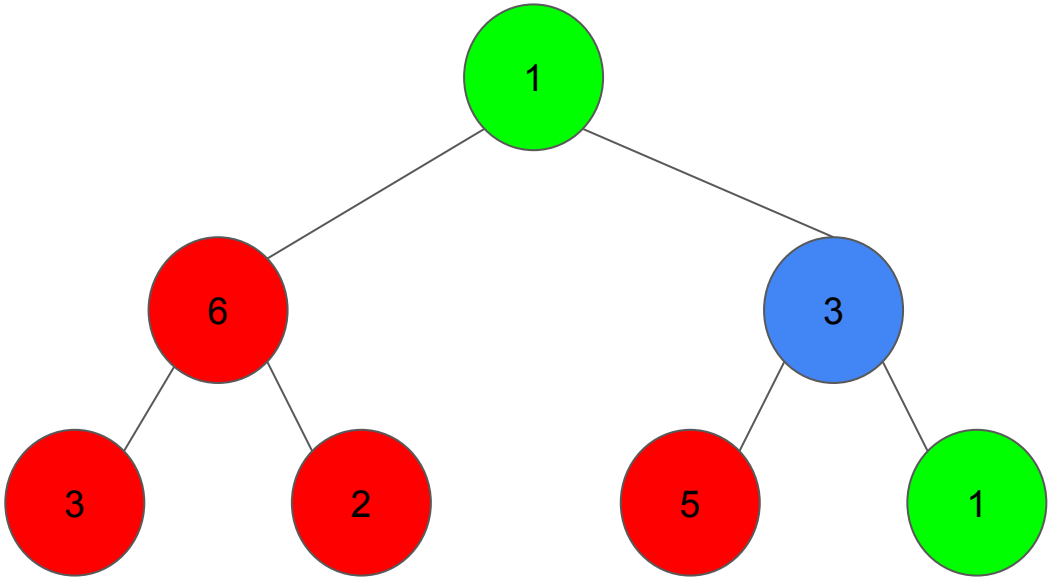
Backtracking 101



Continuamos con b=3, aux=4, por lo que aun puede existir un caso favorable. Continuamos

c=5, lo que deja aux=9, que ya no sirve. Nos retractamos

Backtracking 101



Usamos c=1, llegamos al resultado final y por ende terminamos.

Como se ve ahora, solo revisamos 2 hojas, a diferencia de 4 como hubiera sido en el caso original

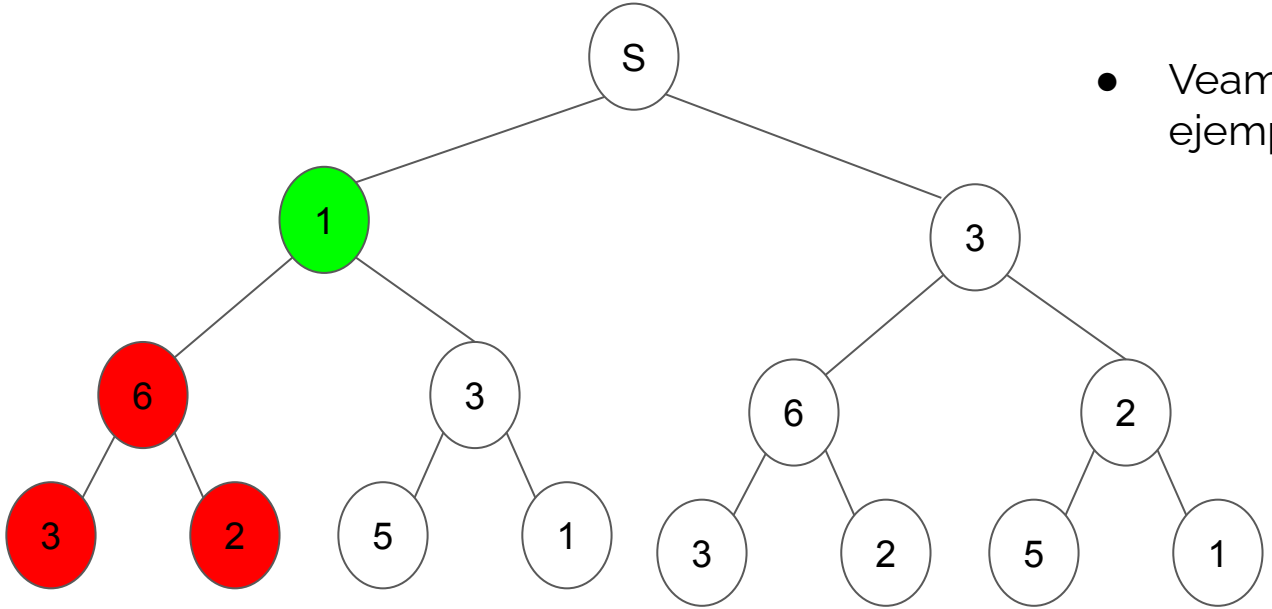
Bakantracking

- A la técnica que vimos ahora se le llama "**Poda**", y tal como indica el nombre, significa no probar ciertas ramas del *árbol de ejecución.*

- Existen más técnicas, las **heurísticas** y la **propagación**

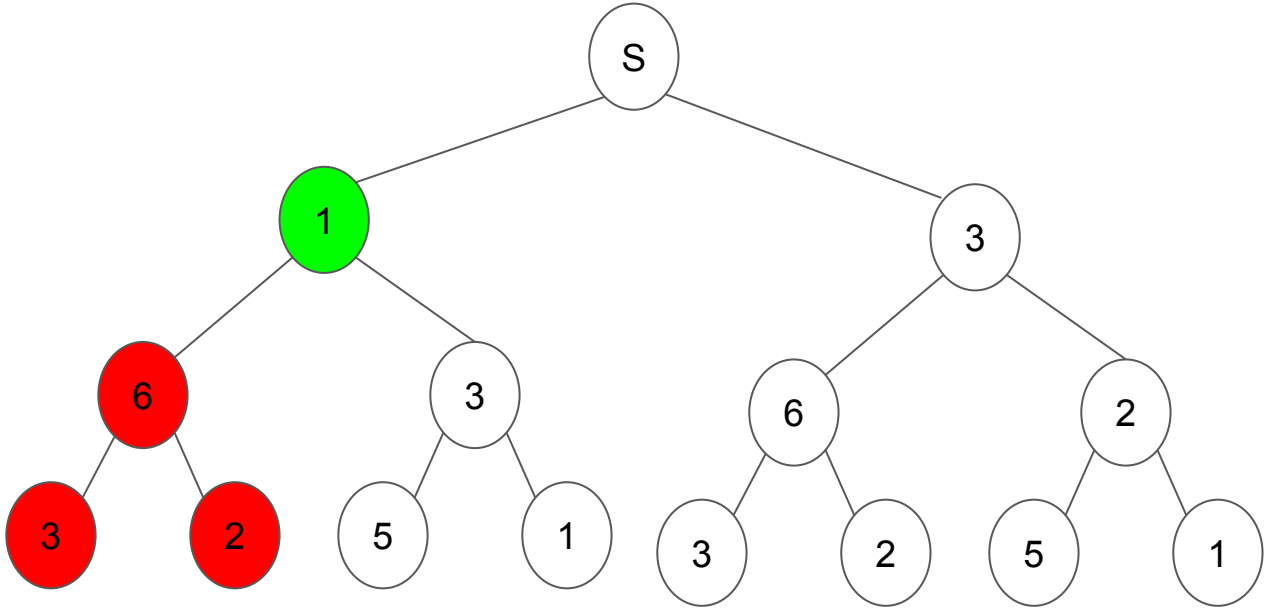
Arbol de ejecución: Similar al árbol de probabilidad, es una herramienta para visualizar todos los estados posibles del programa.

Bakantracking - Propagacion



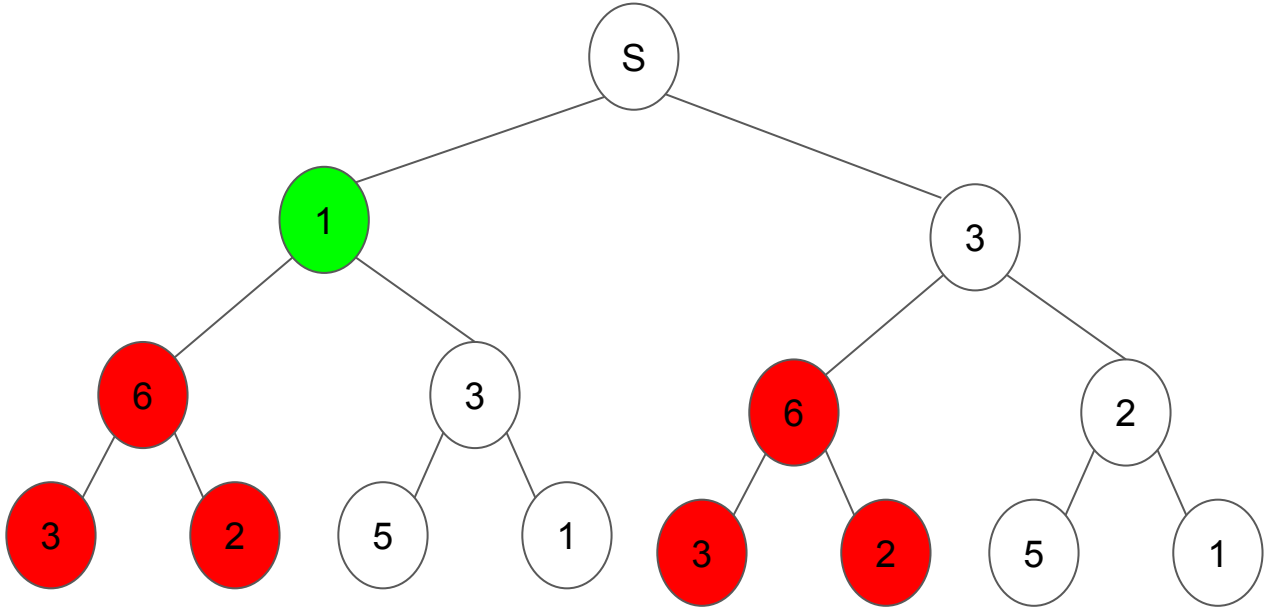
- Veamos una extensión del ejemplo anterior

Bakantracking - Propagacion



En este caso, sabemos que en B=6 y todo ese subárbol se invalida con a=1. Pero no solamente es así, si no que se invalida en cualquier caso, porque dicho subárbol si o si será mayor a x=5

Bakantracking - Propagación



Nos interesa hacer conocer este resultado al resto de las ramas que compartan esto

Bakantracking - Heurísticas

- A veces el orden de cómo recorremos el árbol de ejecución también puede ayudarnos a reducir el tiempo que tarda nuestro programa. En el ejemplo anterior se puede observar que nos convendría partir por la rama de la derecha.

- Si bien acá puede parecer rara la implementación, supongamos que A, B y C vienen ordenados y son números entre 0 y 10. Claramente no nos conviene recorrer ningún número que está a la derecha del primer numero >=5

- Es más, podemos definir una estrategia que dado que se elige un numero cercano a 5, supongase a=3, sabemos que b y c deberían estar a la izquierda del array. Por lo que nos conviene continuar la asignación en dicha dirección

Bakantracking

http://eightqueen.becher-sundstroem.de/

Bakantracking

- Veamos una aplicación en la practica (Rescatado de una tarea personal)

Bakantracking - Problema Propuesto

Chile al Mundial

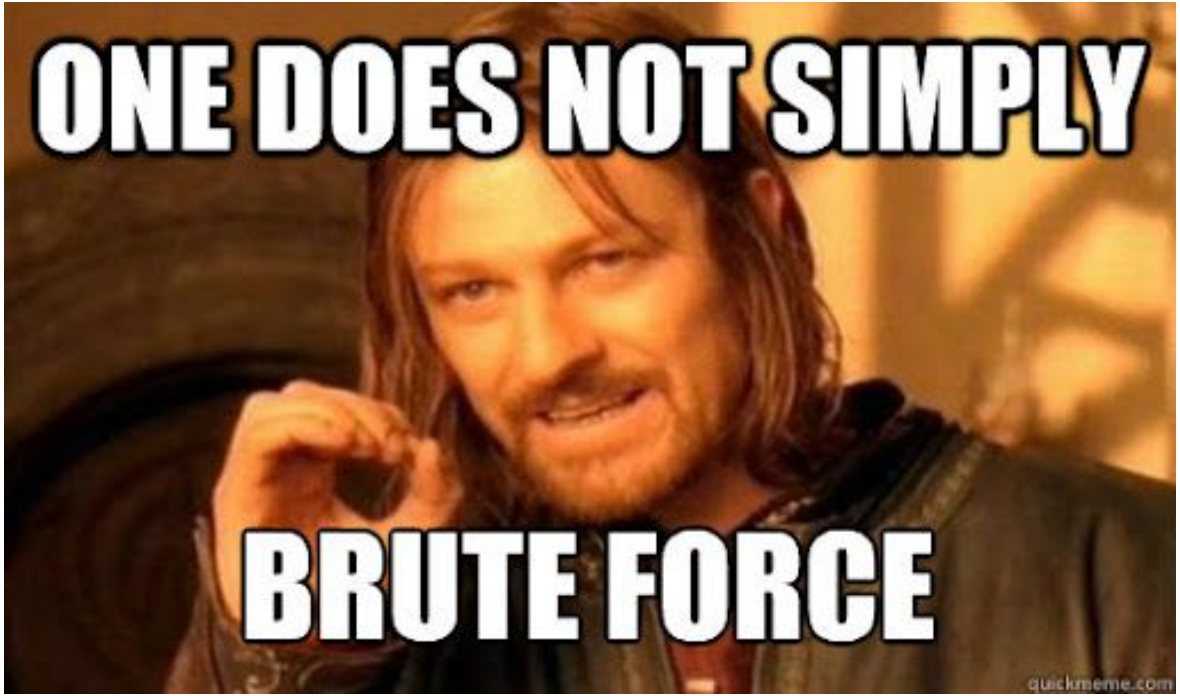
Chile al mundial

Chile al mundial

```
Input: room = [ [1,1,1,1,1,0,1,1],
                [1,1,1,1,1,0,1,1],
                [1,0,1,1,1,1,1,1],
                [0,0,0,1,0,0,0,0],
                [1,1,1,1,1,1,1,1]],
                 row = 1, col = 3
Output : True
```

Chile al mundial

```
Input: room = [ [1,1,1,1,1,0,1,1],
                [1,1,1,1,1,0,1,1],
                [1,0,1,1,1,1,1,1],
                [0,0,0,0,0,0,0,0],
                [1,1,1,1,1,1,1,1]],
                 row = 1, col = 3
Output : False
```



memegracioso.jpeg