Backtrack (vuelta atrás)

3	a	4	b
c	1	d	2
e	4	f	3
2	g	1	h

1 al 4

Para a de 1 a 4
Para b de 1 a 4
Para c de 1 a 4
...
Para h de 1 a 4.

{ Intentar abcdefgh y ver si cumple las reglas.

$4^8 = 2^{16}$

$2^{10} \approx 1000$

$1000 \cdot 2^6 = 64{,}000$

a=2 b=1 c=4 d=3

En muchos problemas, podemos ir dando una solución parcial mediante un vector factible. Backtrack se trata de ir aumentando este vector sistemáticamente hasta que se vuelva infactible, o bien una solución.

5	3	a_1	a_2	7	a_3	a_4	a_5	a_6
6	a_7		1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Para $a_1 \in \{1, 2, 4\}$.

→ Para $a_2 \in \{2, 6\} \setminus \{a_1\}$

→ Para a_3 6 ... si el siguiente ya no tiene donde estar

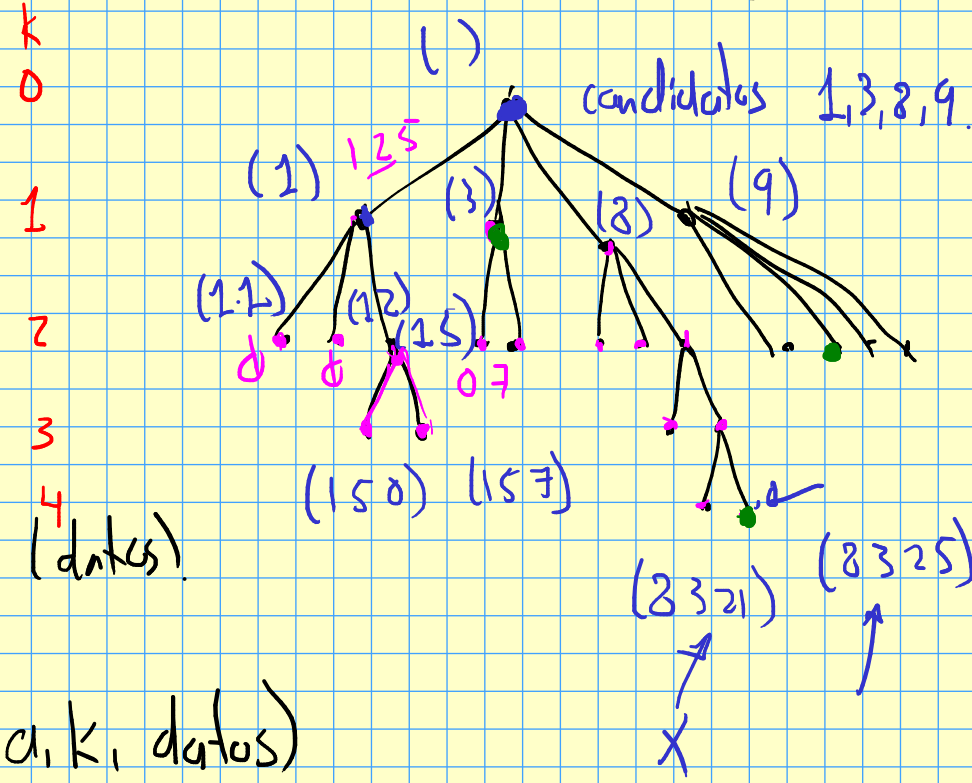
Para $a_n \in \{\dots\}$

→ Verificar solución.

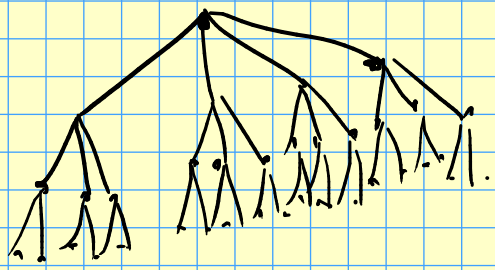
Típicamente, backtrack se implementa usando recursión de la siguiente manera:

vector parcial de sol. — profundidad.

```
backtrack (a, k, datos):
    Si k=0:
        a = inicializar (datos)
        candidatos = primeros candidatos (datos)
    si k>0:
        candidatos = crear candidatos (a, k, datos)
    preprocesar (a, k, datos)
    Para j en candidatos
        agregar j a a
        backtrack (a, k+1, datos)
        quitar último elemento a a
    postprocesar (a, k, datos)
```



más rápido que



Ejemplo Pasar por todas las permutaciones usando backtrack.

Rec. Las permutaciones de 3 elementos abc son abc bac cab
acb bca cba

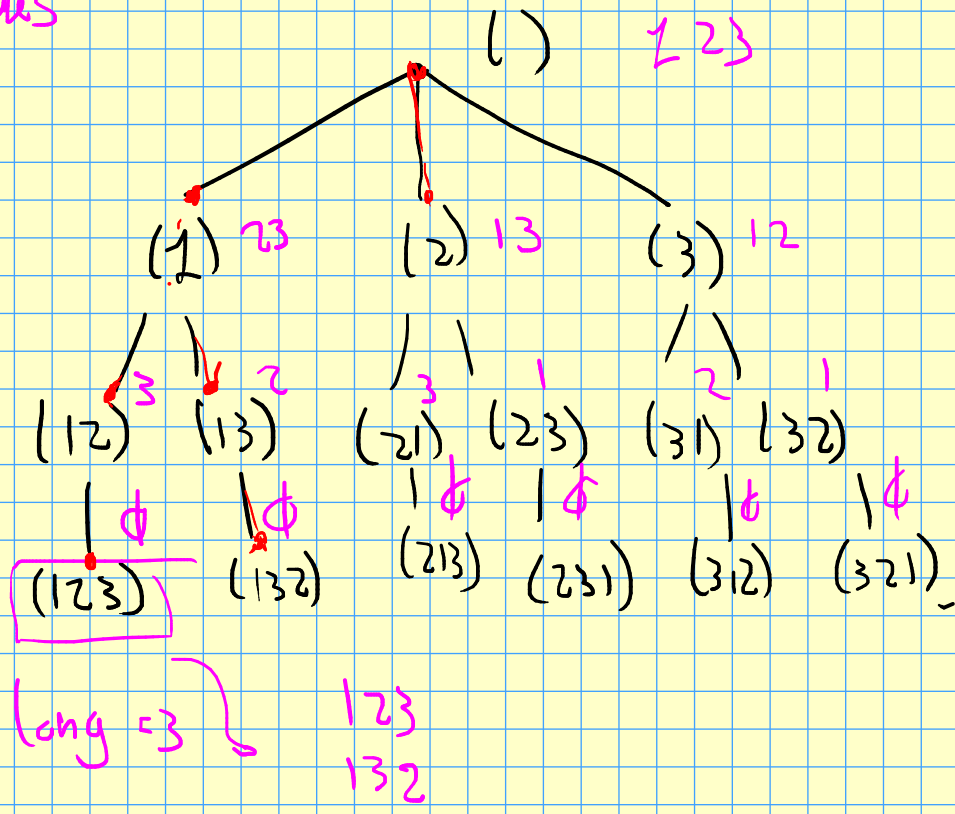
Si tenemos n elementos, son n!

abcdef.

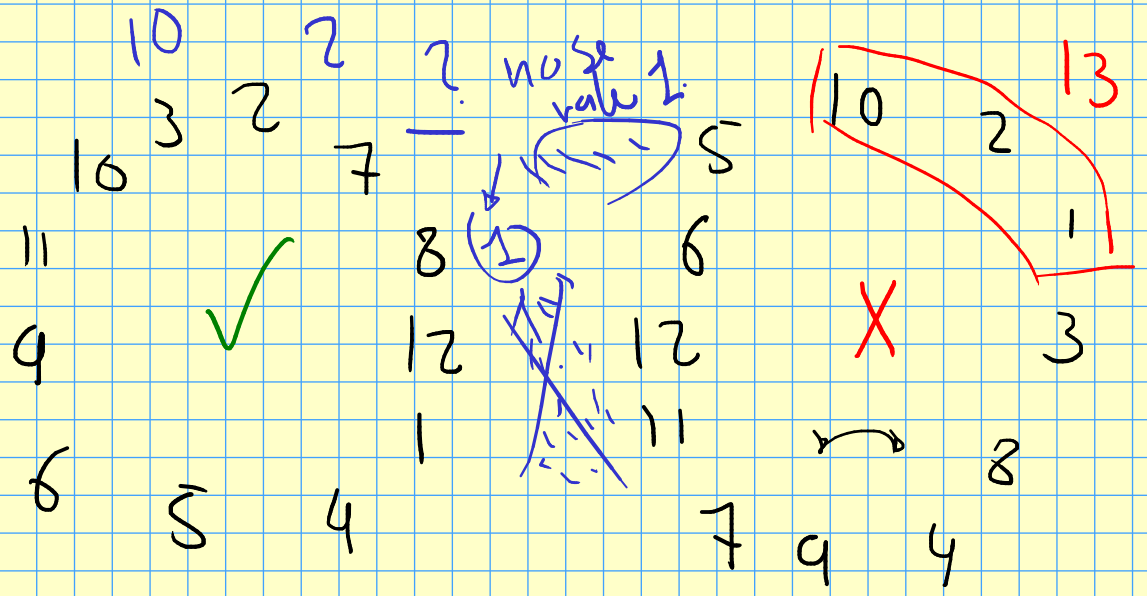
```
def perms(n,a=[],candidates=[],sols=0):
    if len(a)==0:
        candidates=list(range(n))
    if len(a)==n:
        sols+=1
        print(a)
        return(sols)
    for j in candidates:
        a.append(j)
        new_candidates=candidates.copy()
        new_candidates.remove(j)
        sols=perms(n,a,new_candidates,sols)
        a.remove(j)
    return sols
```

candidatos iniciales

pre-procesamiento.



Problema 10. Se cayeron los 12 números de un reloj. Se quieren volver a poner, uno en cada posición. No importa tanto saber la hora, pero es muy importante que la suma de tres de esos números consecutivos (en orden cíclico) no sea 13, porque da mala suerte. ¿De cuántas formas es posible hacer esto?



Mala suerte.

Ideas:
- No considerar todas las rotaciones ni reflexiones.

¿cuántas que sí se valen hay?

$12!$ formas de poner los números.
↳ verificar si cumple o no.

$\frac{12!}{12 \cdot 2} = \frac{11!}{2}$ segundos

- Ideas
- $12!$ segundos ¿cuánto tiempo es?

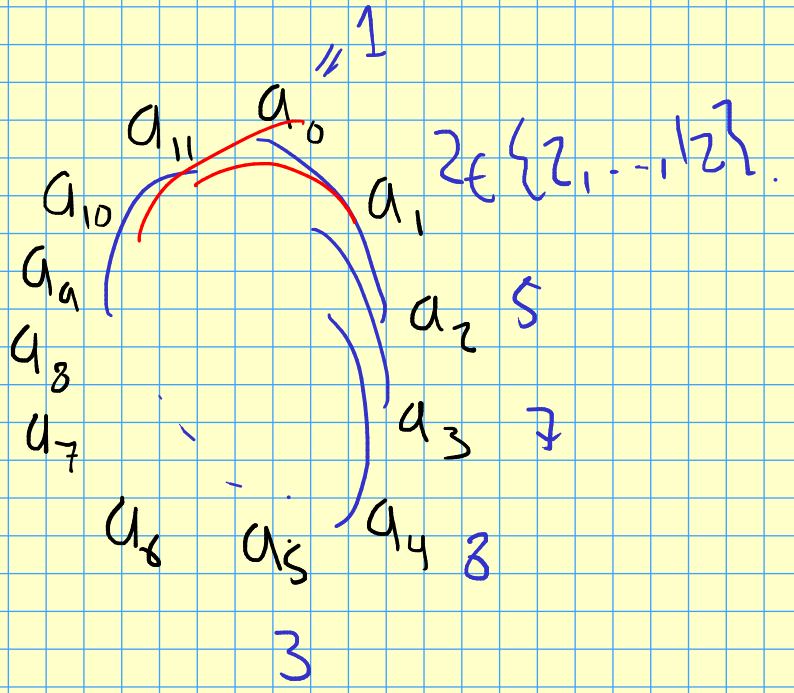
$12! = 12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$.

$12!$ segundos $= 11 \cdot 9 \cdot 8 \cdot 7$ días $= 72 \cdot 77$ días

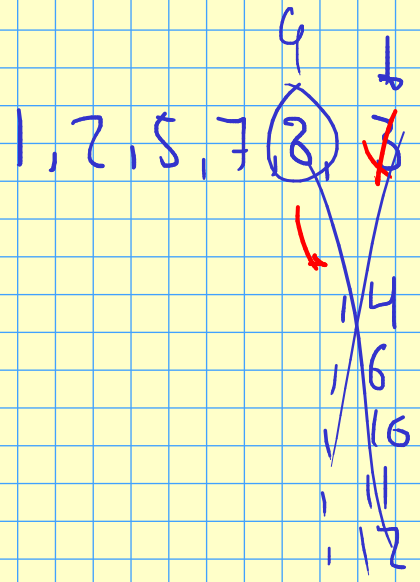
$= 5544$ días

≈ 15 años.

$$\begin{array}{r} 77 \\ 72 \\ \hline 154 \\ 539 \\ \hline 5544 \end{array}$$



$z \in \{2, \dots, 12\}$.



Candidatos

$\not{2}, \not{3}, \not{4}, \not{5}, 6, \not{7}, \not{8}, 9, 10, 11, 12$.

En las elecciones de k elementos de un conjunto de tamaño n sí importa el orden de los objetos.

Por ejemplo las elecciones de 2 objetos de $\{1,2,3,4,5\}$ son

12	21	31	41	51
13	23	32	42	52
14	24	34	43	53
15	25	35	45	54

- Las elecciones de n elementos de un conjunto de tamaño n precisamente son las permutaciones.