

# IIC-2133 — Estructuras de Datos y Algoritmos Backtracking

**Jorge Baier**  
(@jorgebaier)

Departamento de Ciencia de la Computación  
Pontificia Universidad Católica de Chile



**Backtracking** es una técnica recursiva para resolver problemas combinatoriales que se pueden resolver mediante toma de decisiones secuencial.

Un problema de satisfacción de restricciones es una tupla  $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ , donde

- $\mathcal{X} = \{X_1, X_2, \dots, X_n\}$  es un conjunto de *variables*,
- $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$  es un conjunto de *dominios* asociados a variables ( $X_i \in D_i$ ).
- $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$  es un conjunto funciones que reciben una asignación  $\sigma$  de valores para las variables en  $\mathcal{X}$  y retornan  $\top$  o  $\perp$

El objetivo es encontrar una asignación  $\sigma$  de valores para las variables  $\mathcal{X}$  tal que  $C_i(\sigma) = \top$ , para todo  $i \in \{1, \dots, n\}$



# Un Pseudo-Código para Backtracking

```
1 function Backtrack( $P, i, sol$ )  
    Input: problema de satisfacción de restricciones  $P$ , un  
           contador  $i$  para la variable a asignar, un vector  $sol$  para  
           almacenar la solución ( $sol[i] \in D_i \cup \{undef\}$ )  
2 if  $i = n + 1$  then  
3     if  $c(sol) = \top$  para todo  $c \in \mathcal{C}$  then  
4         return True  
5     else  
6         return False  
7 if para algún  $c \in \mathcal{C}$ ,  $c(sol) = \perp$  then return False  
8 for each  $valor \in D_i$  do  
9      $sol[i] \leftarrow valor$   
10    if Backtrack( $P, i + 1, sol$ ) then  
11        return True  
12 return False
```

En el llamado inicial,  $Backtrack(P, 1, sol)$ ,  $sol$  es tal que  $sol[i] = undef$  para todo  $i \in \{1, \dots, n\}$



# Un Ejemplo

Dado un conjunto de números  $C$  y un número  $n$ , escriba una función que encuentre un subconjunto de  $S$  de  $C$  tal que

$$\sum_{x \in S} x = n.$$

[Escribimos un pseudo-código en la pizarra y luego lo implementamos en C]



# Backtracking y Problemas de Optimización

Backtracking también se puede utilizar en problemas de optimización con restricciones.

Un problema de satisfacción de restricciones es una tupla  $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, U \rangle$ , donde

- $\mathcal{X}, \mathcal{D}, \mathcal{C}$  definidos como antes.
- $U$  es una función que recibe una asignación  $\sigma$  para variables en  $\mathcal{X}$  y retorna un número real.

Encontrar  $\sigma$  que maximiza  $U(\sigma)$ , en el conjunto  $\{\sigma' \mid C_i(\sigma') = \top, \text{ para cada } i \in \{1, \dots, n\}\}$ .



# Pseudo-Código Backtracking con Optimización

```
1 function Backtrack( $P, i, sol$ )  
    Input: problema de satisfacción de restricciones  $P$ , un  
            contador  $i$  para la variable a asignar, un vector  $sol$  para  
            almacenar la solución ( $sol[i] \in D_i \cup \{undef\}$ )  
2 if  $i = n + 1$  then  
3     if  $c(sol) = \top$ ,  $\forall c \in \mathcal{C}$  y  $U(bestsol) < U(sol)$  then  
4          $bestsol \leftarrow sol$   
5 if para algún  $c \in \mathcal{C}$ ,  $c(sol) = \perp$  then return  
6 for each  $valor \in D_i$  do  
7      $sol[i] \leftarrow valor$   
8     Backtrack( $P, i + 1, sol$ )
```

En el llamado inicial,  $Backtrack(P, 1, sol)$ ,  $sol$  es tal que  $sol[i] = undef$  para todo  $i \in \{1, \dots, n\}$ .  $bestsol$  es una variable global.

