Backtracking

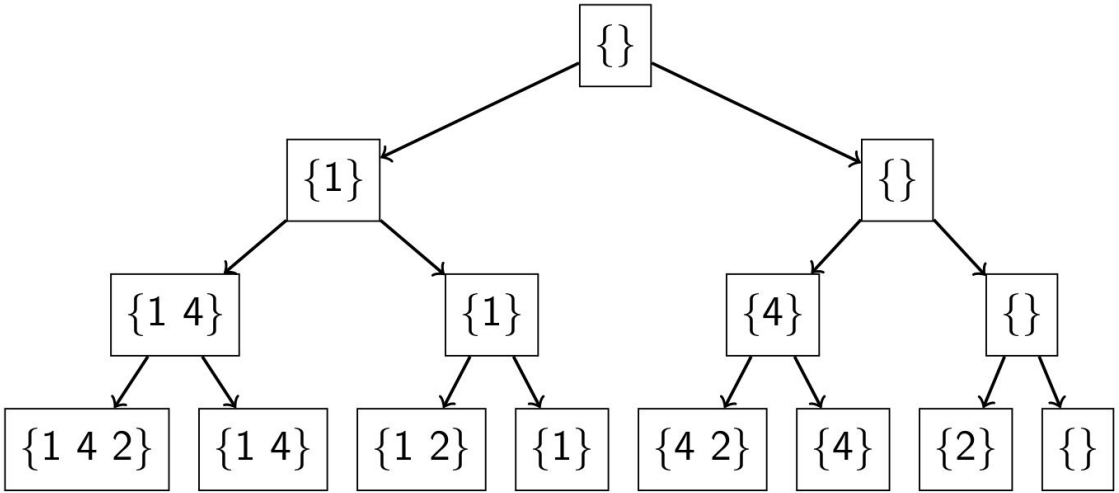
Ejercicio 1

Enunciado

Tenemos un CD que soporta hasta P minutos de música, y dado un conjunto de N canciones de duración p_i (con $1 \leq i \leq m$, y $p_i \in \mathbb{N}$) queremos encontrar la mayor cantidad de minutos de música que podemos escuchar.

Ejercicio 1 - Espacio de búsqueda

$$subsets(c : C) = c \times subsets(C) \cup subsets(C)$$



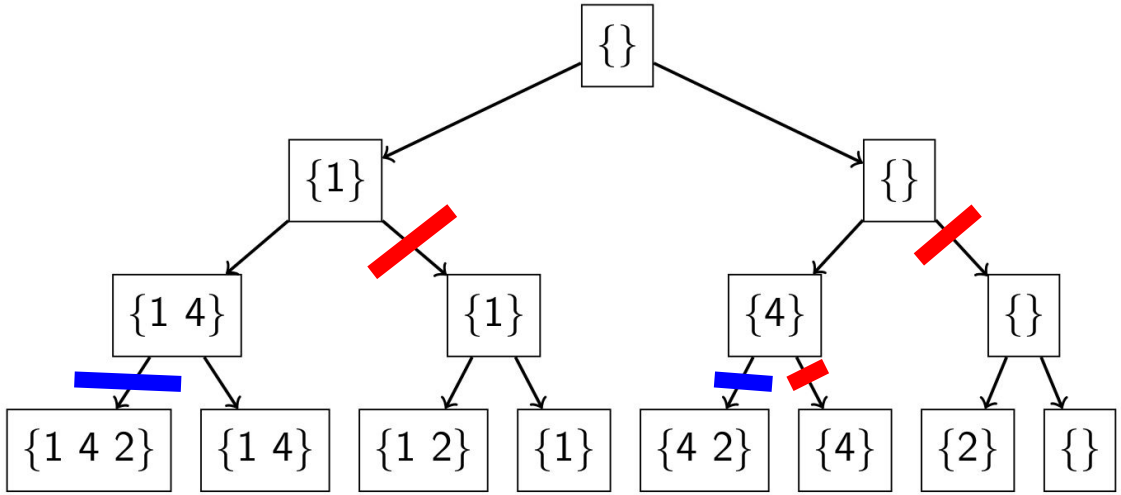
Ejercicio 1 - Algoritmo

Algorithm $BT_{CD}(a, i)$ // a es una solución parcial

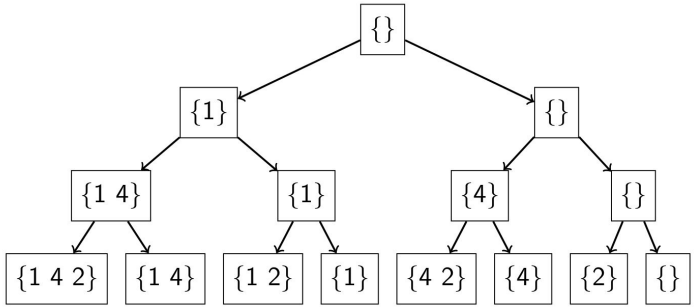
1: **if** $i = N$ **then**
2: **if** $suma(a) \leq P$ & $suma(a) > mejorSuma$ **then**
3: $mejorSuma \leftarrow suma(a)$
4: **end if**
5: **else**
6: $BT_{CD}(a \cup p_i, i + 1)$
7: $BT_{CD}(a, i + 1)$
8: **end if**=0

- La respuesta es $BT_{CD}(\{\}, 0)$

Ejercicio 1 - Podas



Ejercicio 1 - Complejidad



Tiene $\sum_{i=0}^{N} 2^i = O(2^N)$ nodos.

¿Cuántas operaciones hacemos en cada nodo?

En todos los nodos internos realizamos una cantidad constante de operaciones.

La complejidad final entonces es $O(2^N)$.

Ejercicio 2

Enunciado

Dado un tablero de Sudoku de $N \times N$ con algunas casillas ocupadas hay que decidir si se puede completar de forma que sea el resultado final sea un tablero válido.

Cuáles son las soluciones parciales?
Cuál es la función de extensión?
Qué verificamos en las hojas?
Podemos usar podas?
Qué complejidad tienen los nodos?
Cómo podemos acotar la complejidad final?

5	3			7				
6			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

Ejercicio 2

Qué puede hacer ChatGPT?

```
bool SolveSudoku(int grid[N][N]) {
  int row, col;
  if (!FindUnassignedLocation(grid, row, col)) {
    return true; // Sudoku solved successfully
  }
  for (int num = 1; num <= 9; num++) {
    if (isSafe(grid, row, col, num)) {
      grid[row][col] = num; // assign num to unassigned location
      if (SolveSudoku(grid)) {
        return true; // Sudoku solved successfully
      }
      grid[row][col] = UNASSIGNED; // undo the assignment
    }
  }
  return false; // Sudoku cannot be solved
}
```

Ejercicio 2