Backtracking y programación dinámica

Máximo conjunto independiente

```
S0 = 0
S1 = w1
for i=2 to N:
   Si = MAX(wi + Si-2, Si-1)
print(SN)
```

Con el algoritmo anterior se obtiene la máxima sumatoria de pesos, sin embargo no se obtiene el conjunto de nodos como tal.

Para obtener dicho conjunto hay varias alternativas. La más evidente es modificar el algoritmo para que además de ir calculando la sumatoria en cada llamado se vaya acumulando el nodo correspondiente. Esta alternativa requiere memoria O(*N*) adicional.

Backtraking para obtener los elementos

```
i = N
while i > 0:
    if w_i + S_{i-2} >= S_{i-1}:
        print(i)
        i -= 2
    else:
        i--
```

¿Cuál es la eficiencia de este algoritmo?

O(*N*), es decir, no afecta la eficiencia del anterior.

Ejemplo: *w* = 5, 6, 7, 8

i	4	3	2	1	0
w_i	8	7	6	5	0
S_i	14	12	6	5	0
print	**4**	**-**	**2**	**-**	**-**

Programación dinámica

1. Un problema debe poder definirse a partir de subproblemas más pequeños, cuya cantidad sea pequeña (ojalá lineal o a lo sumo polinómica), y cuya solución tenga una complejidad razonable (ojalá constante o a lo sumo lineal). Dicha relación se puede expresar mediante recurrencia y establece que la solución óptima de un subproblema es igual a una función de las soluciones optimas de subproblemas más pequeños.

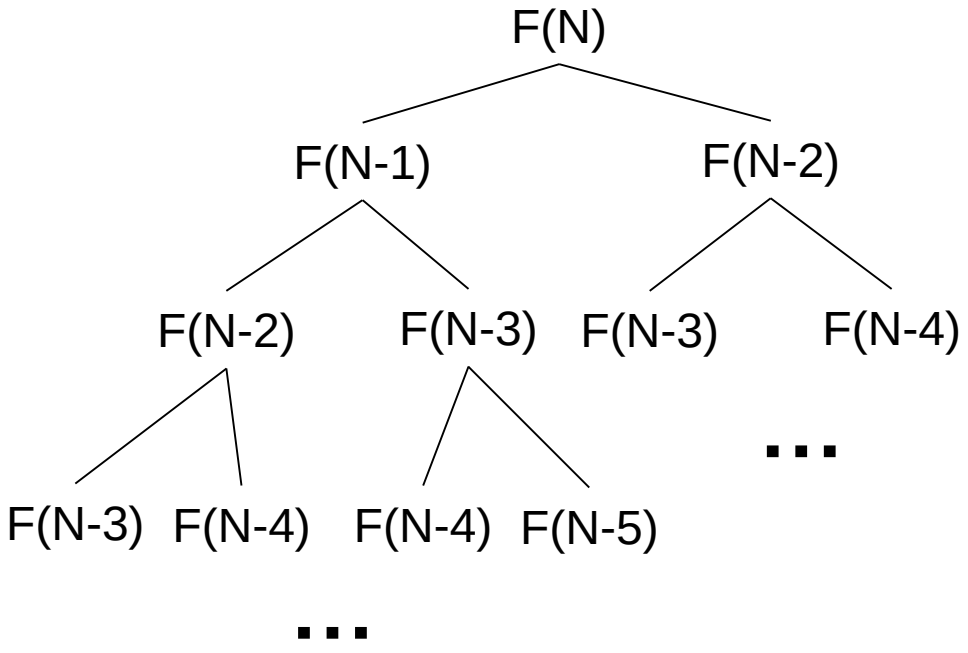
2. Los subproblemas deben poder solucionarse de manera sistemática desde los más pequeños a los más grandes. Durante este proceso tales soluciones se almacenan en memoria para poder accederlas cuando se necesiten sin necesidad de recalcularlas. A este procedimiento se le conoce como “memoización”. Luego de resolver todos los subproblemas, la solución final al problema original es la del subproblema mayor.

3. Una vez obtenida la solución final, elementos complementarios a esta pueden obtenerse vía backtraking.

Programación dinámica

¿Qué otro algoritmo que
conozcamos sigue esta receta?

fib(*N*) = fib(*N-1*) + fib(*N-2*), con f(0) = 0, f(1) = 1



1) `fib(i) = fib(i-1) + fib(i-2)`

2)
$f_0 = 0$
$f_1 = 1$
for i = 2 to N:
    $f_i = f_{i-1} + f_{i-2}$
print(f_N)