.UBA fiuba
FACULTAD DE INGENIERÍA

Backtracking: Problema de la mochila

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Problema de la mochila

- **Contamos con:**
  - una mochila con una capacidad de K kilos
  - un subconjunto del conjunto E de “n” elementos
- **Cada elemento i tiene:**
  - un peso de k_i kilos
  - un valor de v_i.
- **Queremos seleccionar un subconjunto de E**
  - con el objetivo de maximizar la ganancia.
  - el peso total seleccionado no puede superar la capacidad de la mochila.

Problema de la mochila – identificación de los elementos

- Podemos asignarle a cada elemento identificad[illegible] único (un valor entero)
- Asignar un orden a los elementos según su identificador

1	2	3	4	5	6	7	8	9	10	11	12	13
k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}
v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}

Problema de la mochila – Representación de una solución

- Podemos expresar una posible solución como una lista de como mucho "n" elementos
- La lista estará ordenada de menor a mayor según el identificador del elemento
- La lista vacía corresponde a la mochila vacía

1	2	3
3	7	10

→

¿ $k_3 + k_7 + k_{10} \leq K$?

Valor = $v_3 + v_7 + v_{10}$

Árbol de estados del problema

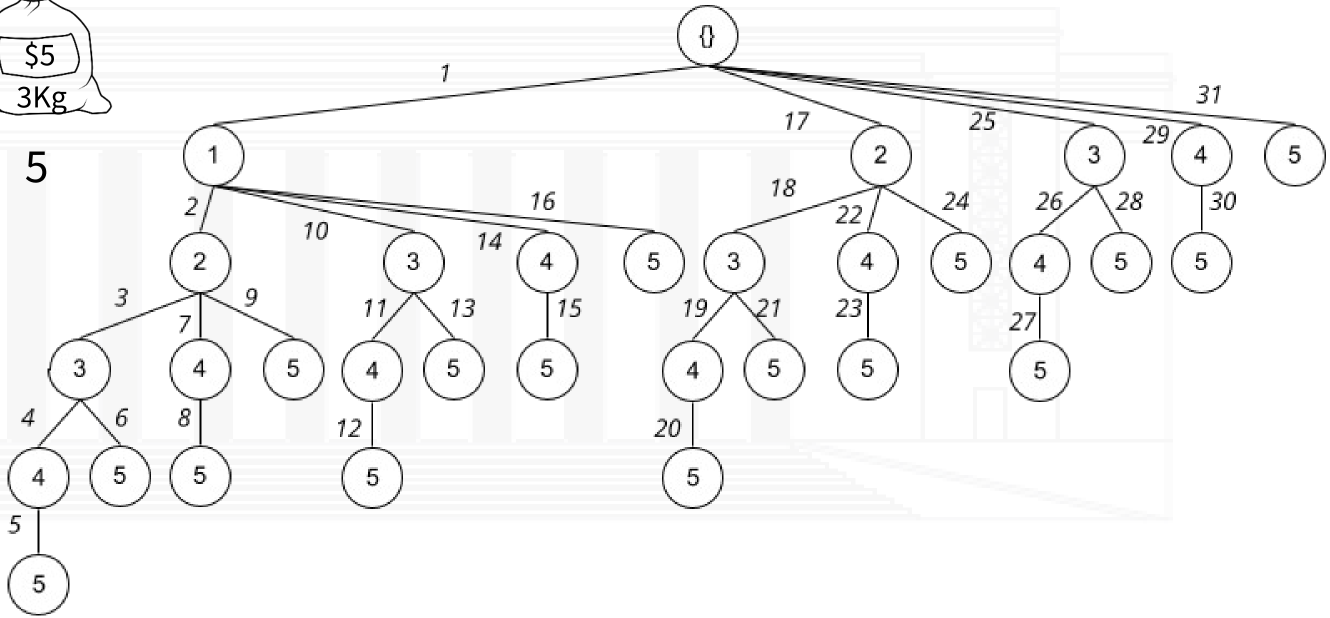
- **La raíz representa la mochila vacía**
- **Un nodo corresponde a ingresar un determinado elemento a la mochila**
  - Se agrega a los anteriores incluidos
- **Los descendientes de cada nodo**
  - corresponden a los posibles elementos a elegir posteriores (según su identificador) al agregado en el mismo
  - Al incluir un elemento ocupa su peso en la mochila y brinda su valor
- **Llamaremos al árbol resultando como árbol combinatorio.**
  - Todos los estados del problema corresponden a estados solución
  - El total de estado del problema corresponde a 2^n

Ejemplo: Árbol de estados del problema

Supongamos la siguiente instancia

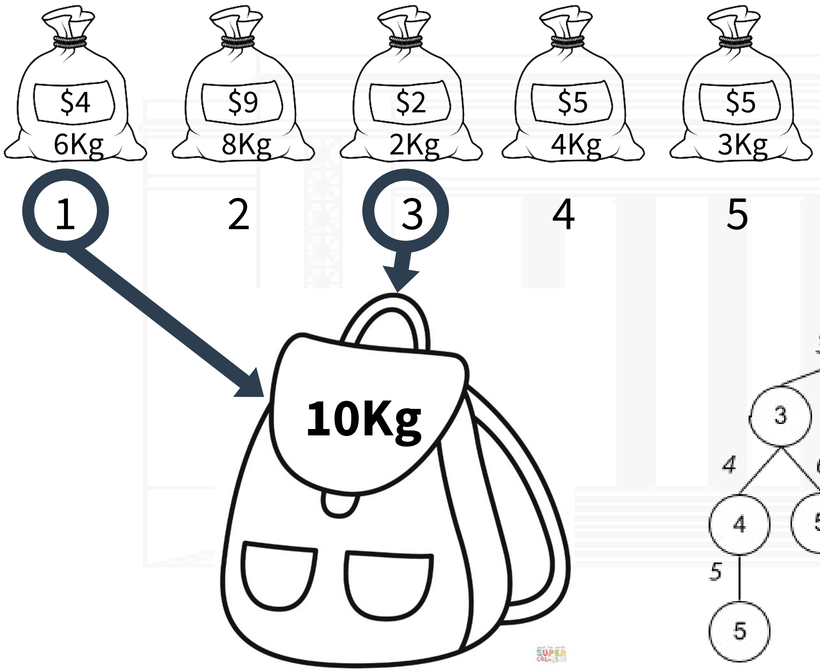
1 2 3 4 5

Los estados se pueden representar como un árbol combinatorio:

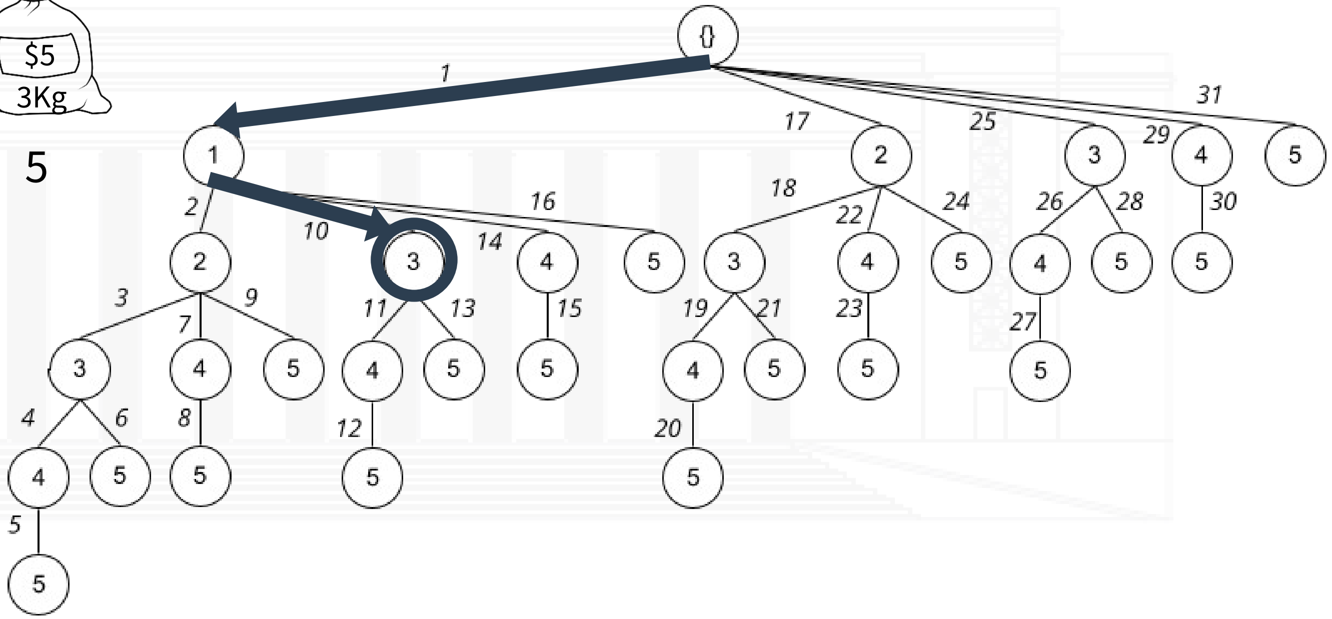


Ejemplo: Árbol de estados del problema

Posible solución:



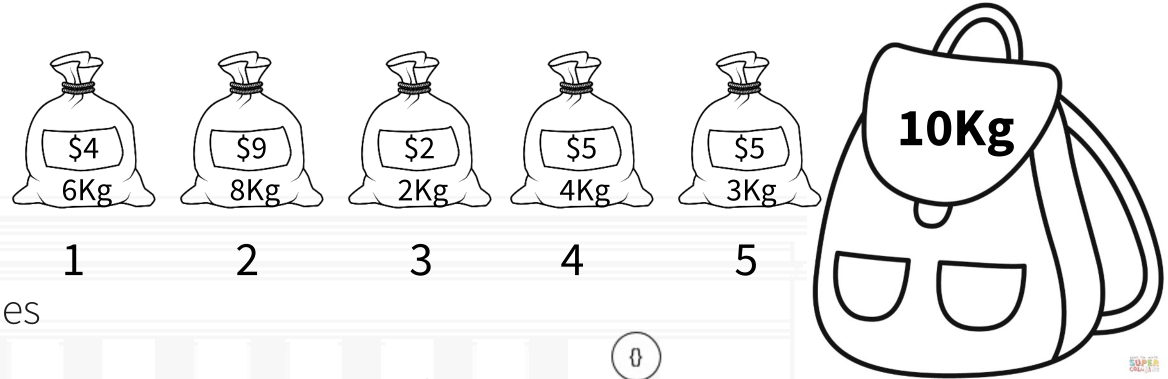
Se corresponde al recorrido en el árbol:



Poda del árbol

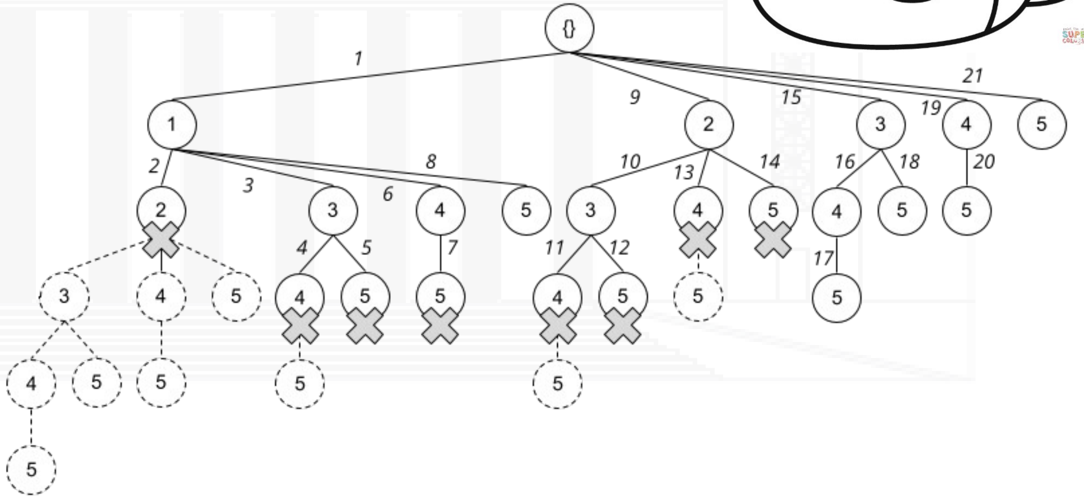
No todos las combinaciones de elementos son posibles

Si un conjunto de elementos supera la capacidad de la mochila, agregar adicionales también lo hará



La función límite verifica la superación de la capacidad

En caso de ausencia, poda el nodo y desestima la inspección de todas las ramas del árbol que las contiene



Backtracking – Pseudocódigo

```
Sea elementos un vector de los elementos disponibles
Sea mochila una lista inicialmente vacía con los elementos seleccionados.
Sea maximaGanancia la cantidad máxima obtenida en la mochila
Sea maximaCombinacion los elementos seleccionados para obtener la máxima ganancia

mochila={}.
maximaGanancia=0
maximaCombinacion=mochila

Backtrack(mochila)
```

Backtracking – Pseudocódigo (II)

```
Backtrack (mochila):
    Si mochila no supera la capacidad disponible
        Sea ganancia la suma de los elementos en la mochila
        si ganancia > maximaGanancia
            maximaGanancia = ganancia
            maximaCombinacion = mochila

    Si mochila tiene capacidad disponible
        Sea ultimoElemento el ultimo elemento añadido en la mochila
        Por cada elemento posterior a ultimoElemento en elementos
            Agregar elemento a mochila
            Backtrack(mochila)

            Quitar elemento de mochila
```

Complejidad Temporal

En el peor de los casos no es posible podar nodos del arbol

- Debemos recorrer cada uno de los nodos del árbol con una complejidad $O(2^n)$

En los nodos en el peor de los casos debemos hacer un trabajo O(n)

- Para resguardar una mejor solución encontrada
- Para calcular el peso y ganancia acumulado (se podría realizar en O(1))

La multiplicación de ambas complejidades nos brinda la complejidad temporal del algoritmo.

Complejidad Espacial

Por la implementación recursiva

Por cada llamado en profundidad en el árbol incluimos el consumo de memoria adicional

La memoria utilizada

Es proporcional a la profundidad máxima de la recursión generada

Para el árbol combinatorio la profundidad máxima es “n”.

En cada nivel de profundidad

Se agrega un elemento a la mochila

Se realizan cálculos que requieren O(1) de almacenamiento

Por lo que la complejidad espacial es O(n)