Parcial-3-2017-Tema-6-y-7-ADA.pdf

ApuntesDeConfi_

Análisis y diseño de algoritmos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Problema 1 (6 ptos). En un país determinado se emiten n sellos diferentes de valores naturales positivos $s_1, s_2, \ldots, s_n$. Se quiere enviar una carta y se sabe que la correspondiente tarifa postal es T. ¿De cuántas formas diferentes se puede franquear exactamente la carta, si el orden de los sellos no importa y tenemos un número ilimitado de cada tipo?

Ejemplo: para $s_1 = 1$, $s_2 = 2$ y $T = 2$ tenemos 2 formas distintas: utilizar 0 sellos s_1 y 1 sello s_2, o bien, utilizar 2 sellos s_1 y 0 sellos s_2.

Se desea resolver el problema mediante **Backtracking**. Se pide:

a) Diseñar el algoritmo indicando claramente, la estructura de la solución, el estado inicial de la misma, la condición que determina cuándo una solución es final, la política de ramificación y las condiciones de validez de los estados del árbol de exploración.

b) Implementar en Java o pseudocódigo el algoritmo anterior. Indicar cuál sería la llamada inicial al algoritmo.

c) Resolver el problema para el ejemplo con $T = 6$ y los siguientes tipos de sellos $s_1 = 3$, $s_2 = 2$ y $s_3 = 1$, indicando el orden en el que los nodos del árbol de exploración se van procesando.

a.) Estructura de la solución: Lista de listas $L = [L_1, \ldots]$: $L_i = [L_{i_1} \ldots L_{ij}]$

↳ Estado inicial: $L = []$

Política de ramificación:

Restricciones explícitas: $S_i : 1 \leq i \leq n$

Restricciones implícitas: $S_i : 1 \leq i \leq n$, $S_i \leq T - \text{suma}(aux)$

para no repetir convinaciones → $S_i < aux_j$ siendo j el último elemento de aux

↳ precondición S tiene que estar ordenado crecientemente.

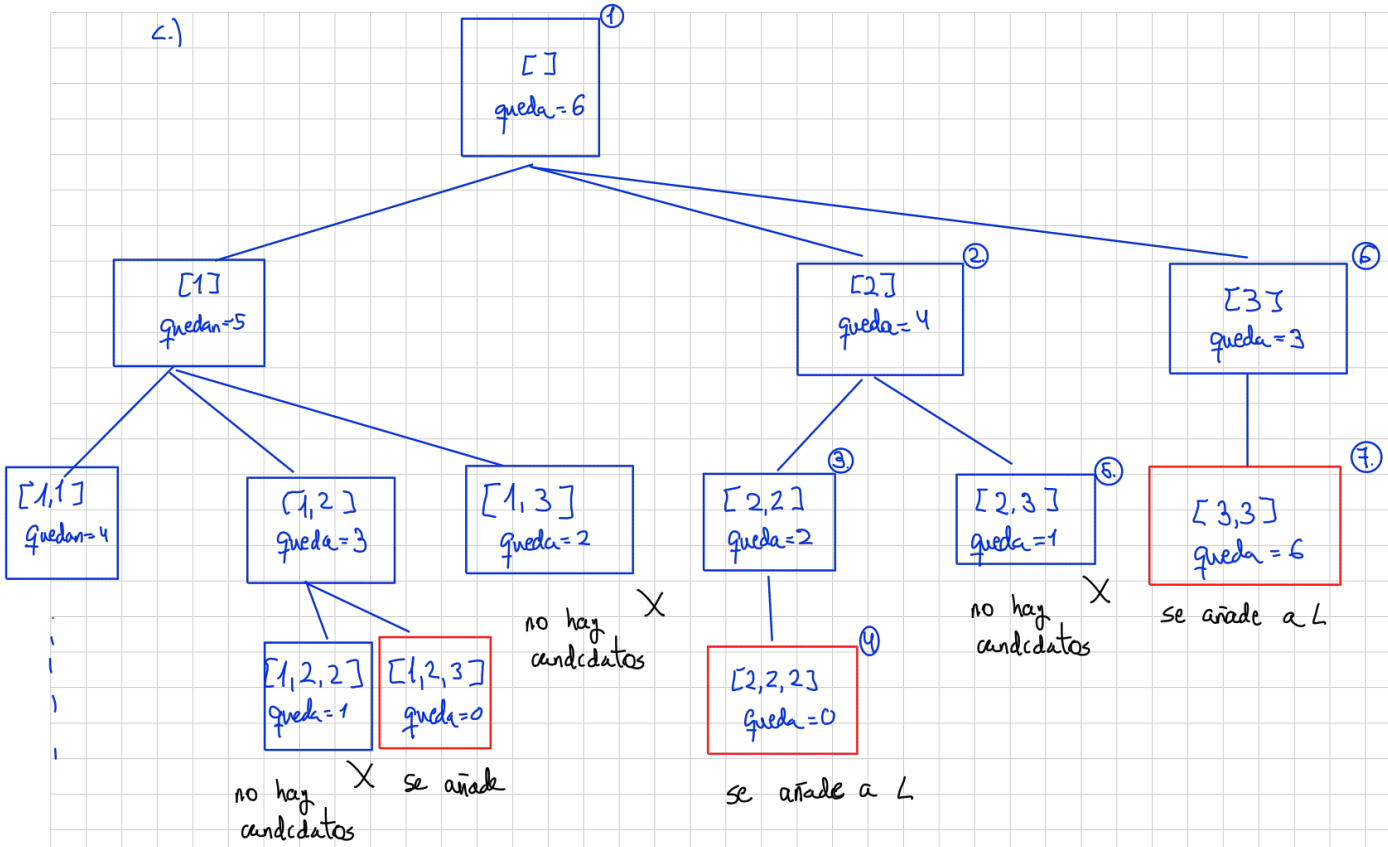
* aux es la posible convinación que estamos estudiando.

$$\text{Suma}(aux) = \sum_{i=1}^{aux.length} aux_i$$

Función de terminación: Si $\text{Suma}(aux) = T$ aux se añade a L y se prueba para el resto de candidatos.

Terminará cuando no queden candidatos.

Función objetivo: ———



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

Que te den **10 € para gastar** es una fantasía. ING lo hace realidad.

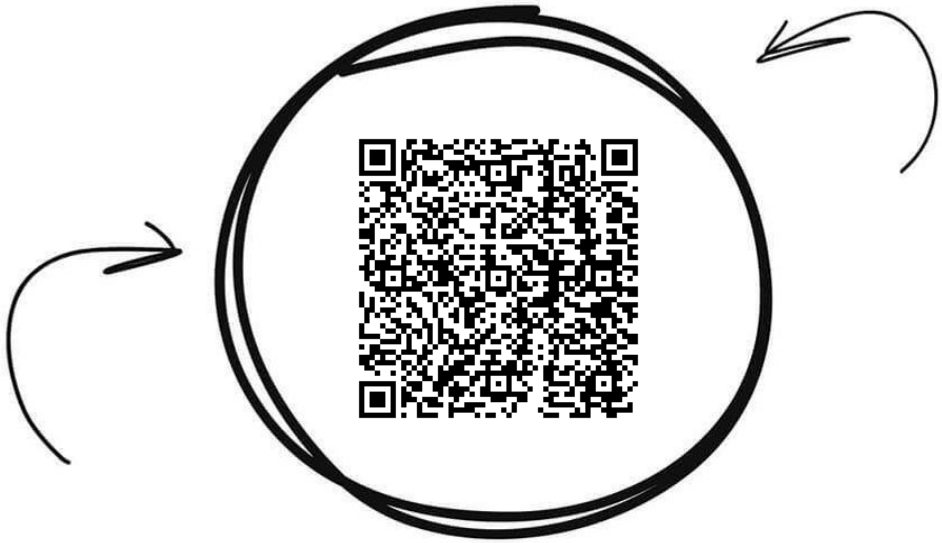
Abre la **Cuenta NoCuenta** con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Quiero el cash

Consulta condiciones **aquí**

do your thing

Análisis y diseño de algoritmos



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

```
import java.util.ArrayList;
import java.util.List;

public class ex17 {
    public static List<List<Integer>> franquear (int[] s, int t) {
        List<List<Integer>> sol = new ArrayList<>();
        List<Integer> aux = new ArrayList<>();
        procesarFranqueo(s,t,aux,0,sol);
        return sol;
    }

    private static void procesarFranqueo(int[] s,int t, List<Integer> aux,int suma,
List<List<Integer>> sol) {
        if (suma == t) {
            sol.add(new ArrayList<>(aux));
        } else {
            List<Integer> candidatos = buscaCandidatos(s,t,aux,suma);
            for (Integer candidato : candidatos) {
                aux.add(candidato);
                suma += candidato;
                procesarFranqueo(s,t,aux,suma,sol);
                suma -= candidato;
                aux.remove(candidato);
            }
        }

    }

    private static List<Integer> buscaCandidatos(int[] s, int t, List<Integer> aux, int suma) {
        List<Integer> candidatos = new ArrayList<>();
        for (int i = 0; i<s.length; i++) {
            if (aux.isEmpty() || aux.get(aux.size()-1) >= s[i] && s[i] <= (t - suma))
                candidatos.add(s[i]);
        }
        return candidatos;
    }

    public static void main(String[] args) {
        int[] s = {2, 3};
        System.out.println(franquear(s,6));
    }
}
```

Se desea utilizar la técnica de Ramificación y Poda para encontrar la solución del problema anterior que minimiza el número de sellos utilizados para franquear la carta.

a) Diseñar el algoritmo indicando claramente, la estructura de la solución, el estado inicial de la misma, la condición que determina cuándo una solución es final, la política de ramificación y las condiciones de validez de los estados del árbol de exploración, y la función de estimación.

b) Dibujar el árbol de exploración para la misma instancia del problema con T = 10 y los tipos de sello s_1 =3, s_2 = 2 y s_3 = 1, indicando el orden en el que los nodos del árbol de exploración se van procesando, cuándo se podan las ramas y cómo va evolucionando la mejor solución encontrada.

Estructura de la solución: Lista $L = [L_1, \dots L_j]$ donde se indicará que sellos hay que usar.

↳ **Estado inicial:** $L = []$

Política de ramificación

Restricciones explícitas: Si : $1 \leq i \leq n$

Restricciones implícitas: Si : $1 \leq i \leq n$, $Si \geqslant L_j$, $Si \leq T - \sum_{i=1}^{L.size} L_i$

siendo L_j el ultimo sello que se ha añadido a L

*Precondición S tiene que estar ordenado de menor a mayor.

Función de terminación: cuando no queden mas candidatos o $\sum_{i=1}^{L.size} L_i = T$

Función objetivo: L. size() sea mínimo.

Función de cota: $f(L) = g(L) + h(L)$

$g(L) = L.\text{sizer}()$

$h(L) = \left(T - \sum_{i=1}^{L.size} L_i \right) / Si$ redondeado hacia arriba

siendo Si el sello de mayor tarifa que no supere $\left(T - \sum_{i=1}^{L.size} L_i \right)$ si no hay candidato $h(L)$ es ifinito