Práctico 2021-05-26

Práctico 3.1 - Ejercicio 2

Contraejemplo propuesto:

Denominaciones: 1, 5, 12
Cambio a entregar: 15

Solución ópitma (a ojo): 3 monedas de 5.

El algoritmo voraz del problema de la moneda da como resultado: 1 moneda de 12 y 3 de 1 (total 4 monedas).

Práctico 3.1 - Ejercicio 8

1. Entender enunciado.

- n troncos.
- Cada tronco i, irradia temperatura k_i y dura prendido tiempo t_i

  Se pide encontrar el **orden** en que se utilizan la menor cantidad de troncos posible entre las 22 y las 12 hs del día siguiente, tal que:
  - Entre las 22 y las 6, la temperatura irradiada no es menor a K1.
  - Entre las 6 las 12, la temperatura irradiada no es menor a K2.
  - Asumiremos K1 > K2

2. Criterio de selección.
   Entre las 22 y las 6: elijo el tronco con tiempo t_i mayor, tal que k_i es mayor o igual a K1.
   Entre las 6 y las 12: idem, pero con K2.

3. Estructuras de datos.

```
type Tronco = tuple
                id: nat
                calor: float
                tiempo: float
              end tuple

fun estufaVoraz(S: Set of Tronco, K1: float, K2: float) ret L: List of Tronco
end fun
```

4. Implementar el algoritmo

```
fun estufaVoraz(S: Set of Tronco, K1: float, K2: float) ret L: List of Tronco
  var C: Set of Tronco
  var t: Tronco
  var h: float

  C := copy_set(S,C)
  h := 0
  L := empty_list()
  while h < 14 do
    if h < 8 then
      t := elegirTronco(C,K1)
    else
      t := elegirTronco(C,K2)
    fi
    elim_set(C,t)
    addR(L,t)
    h := h + t.tiempo
  od
  destroy_set(C)

end fun

fun elegirTronco(C: Set of Tronco, K: float) ret t: Tronco
  var B: Set of Tronco
  var max_tiempo : float
  var t' : Tronco

  max_tiempo := -infinito
  B := copy_set(C)

  while not is_empty_set(B) do
    t' := get(B)
    if t'.tiempo > max_tiempo && t'.calor >= K then
      max_tiempo := t'.tiempo
      t := t'
    fi
    elim(B,t')
  od
  destroy_set(B)

end fun
```