Practico 3.3 Backtracking

1

Modifique el codigo del algoritmo que resuelve el problema de la moneda utilizando backtracking, de manera que devuelva que monedas se utilizan, en vez de solo la cantidad.

```
fun cambio (P: Nat, M: Set of Moneda) ret res : Set of Moneda
var aux_m, cambio_aux ,cambio_aux_mas_k  : Set of Moneda
    var k : moneda
    var aux_p : Nat

    aux_m : copy_set(M)
    aux_p := P
    if(aux_p = 0)
        then
            res := empty_set()
    else
        if(P > 0 && !is_empty(aux_m))then
            ABORT
            res := { infinito }
        else
             k := get(aux_p)
             elim(aux_p, k)

            cambio_aux := cambio(P, aux_p)
            cambio_aux_mas_k :=  add_set(k, cambio(P-k, M))

            if( k > P V set_length(cambio_aux) < set_length(cambio_aux_mas_c))then
                res := cambio_aux
            else
//            ( k <= P V set_length(cambio_aux) => set_length(cambio_aux_mas_c))
                res := cambio_aux_mas_c
            fi
            destroy_set(aux_p)
        fi
    fi
end fun

Algoritmo 2020
{- devolvemos un par (n, l) a donde n : nat, l : List of nat -}
fun cambio(d:array[1..n] of nat, i,j: nat) ret r: nat x List of nat
  var r1, r2: nat x List of nat

  if j = 0 then r := (0, empty_list())
  else if i = 0 then r := (∞, empty_list())   {- cualquier lista da igual acá -}
  else if d[i] > j then
    r := cambio(d,i-1,j)
  else
    {- acá está lo interesante -}
    r1 := cambio(d,i-1,j)     {- r1 es un par -}
    r2 := cambio(d,i,j-d[i])  {- r2 es un par -}
    if r1.fst < 1 + r2.fst then
      r := r1
    else
      addr(r2.snd, d[i])
      r.fst := 1 + r2.fst
      r.snd := r2.snd
    fi
  fi
end fun
```

2

En un extrano pais las denominaciones de la moneda son 15, 23 y 29, un turista quiere comprar un recuerdo pero tambien quiere conservar el mayor numero de monedas posibles. Los recuerdos cuestan 68, 74, 75, 83, 88 y 89. Asumiendo que tiene suficientes monedas para comprar cualquiera de ellos, ¿cual de ellos elegira? ¿que monedas utilizara para pagarlo? Justificar claramente y mencionar el metodo utilizado.

- Podemos elegir `68` $:= \{29, 29, 23V29\}$ o `74` $:= \{29, 29, 29\}$ o `75` $:= \{29, 29, 29\}$.
- Si bien el problema no lo dice, si se debe tener en cuenta el valor minimo a gastar la moneda sería `68`, caso contrario si quisiera tener el recuerdo de mayor valor con la menor cantidad de monedas utilizadas `74`.
- El metodo utilizado es seleccionar de las denominaciones la menor cantidad de monedas para pagar cada uno de los recuerdos y elegir el que gaste menos monedas.

3

Una panaderia recibe n pedidos por importes $m_1, ..., m_n$, pero solo queda en deposito una cantidad H de harina en buen estado. Sabiendo que los pedidos requieren una cantidad $h_1, ..., h_n$ de harina (respectivamente), determinar el maximo importe que es posible obtener con la harina disponible.

$Resolucion:$

- Para cada pedido tenemos una variable i que nos indica costo m_i y la cantidad de harina necesaria h_i
- Tenemos cantidad de harina disponible j.
- La funcion $max\_import(i, j)$ = "Mayor cantidad de importe de i pedidos con j harina."
- La llamada principal que nos va a devolver el valor va a ser $max\_import(H, n)$ = "Maximo importe de n pedidos con H harina"

$$max\_import(j,i) = \begin{cases} 0 & , j = 0 \lor i = 0 \\ max\_import(j, i-1) & , j < h_i \land (j > 0 \land i > 0) \\ max(max\_import(j, i-1), max\_import(j - h_i, i-1) + m_i) & , j => h_i \land (j > 0 \land i > 0) \end{cases}$$

4

Usted se encuentra en un globo aerostatico sobrevolando el oceano cuando descubre que empieza a perder altura porque la lona esta levemente dañada. Tiene consigo n objetos cuyos *pesos* $p_1, ..., p_n$ y *valores* $v_1, ..., v_n$ conoce. Si se desprende de al menos P kilogramos lograra recuperar altura y llegar a tierra firme, y afortunadamente la suma de los pesos de los objetos supera holgadamente P. ¿Cual es el menor valor total de los objetos que necesita arrojar para llegar sano y salvo a la costa?

$Resolucion:$

- Para cada objeto n tenemos la variable i que tiene un peso p_i con valor v_i
- Por otro lado tenemos $weigth$ que es el peso actual que se necesita tirar para llegar a tierra firme
- El algoritmo $min\_value(weigth, i)$ = "Menor valor posible que se puede tirar para perder $weigth$ usando i objetos"
- La funcion principal seria $min\_value(P, n)$ = "Menor valor posible al tirar P kilogramos usando n objetos."

$$min\_value(weigth,i) = \begin{cases} 0 & , weigth = 0 \land i > 0 \\ \infty & , weigth => p_i \land i = 0 \\ min(min\_value(weigth, i-1), min\_value(weigth - p_i, i-1) + v_i) & , (weigth => p_i \land i > 0) \end{cases}$$

5

Sus amigos quedaron encantados con el telefono satelital, para las proximas vacaciones ofrecen pagarle un alquiler por el. Ademas del dia de partida y de regreso (p_i y r_i) cada amigo ofrece un monto m_i por dia. Determinar el maximo valor alcanzable alquilando el telefono.

$Resolucion:$
-Cantidad de amigos total a,
-Por otro lado una variable k que es la cantida de amigos actual que tengo para prestar, que tiene un p_k (dia de partida) ,(dia de regreso) r_k y un m_k que es monto que paga por dia
-El algoritmo recursivo $max\_money(k, p_k)$ = "Mayor valor alcanzable alquilando el telefono desde el día p_k a k amigos"
-La llamada principal es $max\_money(a, p_a)$ = "Mayor valor alcanzable alquilando el telefono desde el día (p_i) a (a) todos los amigos"

$$max\_money(k, p_k) = \begin{cases} 0 & , k = 0 \\ max\_money(k-1, p_k) & , p_k > r_k \\ max(max\_money(k-1, p_k), max\_money(k-1, r_k) + (r_k - p_k) * m_k) & , p_k <= r_k \land k > 0 \end{cases}$$

6

Un artesano utiliza materia prima de dos tipos: A y B. Dispone de una cantidad MA y MB de cada una de ellas. Tiene a su vez pedidos de fabricar n productos $p_1, ..., p_n$ (uno de cada uno). Cada uno de ellos tiene un valor de venta $v_1, ..., v_n$ y requiere para su elaboracion cantidades $a_1, ..., a_n$ de materia prima de tipo A y $b_1, ..., b_n$ de materia prima de tipo B. ¿Cual es el mayor valor alcanzable con las cantidades de materia prima disponible?

$Resolucion:$

- Maxima cantida de materia prima total MA y MB
- Cantidad actual de materia prima A_k y B_k
- Tenemos n productos totales
- Tenemos i productos actuales tal que v_i es su valor de venta, a_i (de materia prima A) y b_i (de materia prima B)
- El algoritmo $max\_value(i, A_k, B_k)$ = "Maximo valor posible de i pedidos con la cantidad disponible A_k y B_k de materia prima"
- La llamada principal es $max\_value(n, MA, MB)$ = "Maximo valor posible de n pedidos con la cantidad disponible MA y MB de materia prima"

$$max\_value(i, A_k, B_k) = \begin{cases} 0 & , i = 0 \\ max\_value(i-1, A_k, B_k) & , i > 0 \land (A_k < a_i \lor B_k < \\ max(max\_value(i-1, A_k, B_k), max\_value(i-1, A_k - a_i, B_k - b_i) + (v_i)) & , i > 0 \land (A_k >= a_i \lor B_k \end{cases}$$

7

En el problema de la mochila se buscaba el maximo valor alcanzable al seleccionar entre n objetos de valores $v_1, ..., v_n$ y pesos $w_1, ..., w_n$, respectivamente, una combinacion de ellos que quepa en una mochila de capacidad W. Si se tienen dos mochilas con capacidades $W1$ y $W2$, ¿cual es el valor maximo alcanzable al seleccionar objetos para cargar en ambas mochilas?

$Resolucion:$

- Tenemos n objetos en total.
- Sea $W1$ y $W2$ la capacidad maxima de cada mochila.
- Definimos i como la cantida de objetos actuales, siendo su valor (v_i) y (w_i) su peso.
- Teniendo w_1 y w_2 la capacidad actual disponible de la mochila.
- El algoritmo $max\_value(i, w_1, w_2)$ va a obtener el valor maximo de i objetos para w_1 y w_2 capacidad disponible en el momento.
- La llamada principal es $max\_value(i, W1, W2)$ = "Mayor valor posible de n objetos con $W1$ y $W2$ capacidad"

$$max\_value(i, w_1, w_2) = \begin{cases} 0 \\ 0 \\ max\_value(i-1, w_1, w_2) \\ max(max\_value(i-1, w_1 - w_i, w_2) + (v_i), max\_value(i-1, w_1, w_2 - w_i) + (v_i), max\_value(i-1, w_1 \\ max(max\_value(i-1, w_1, w_2), max\_value(i-1, w_1, w_2 - w_i) + (v_i)) \\ max(max\_value(i-1, w_1, w_2), max\_value(i-1, w_1 - w_i, w_2) + (v_i)) \end{cases}$$

- Preguntar porque no hacer $MAXmax(entraw1, noentra)max(entraw2, noentra)$

8

Una fabrica de automoviles tiene dos lineas de ensamblaje y cada linea tiene n estaciones de trabajo, $S1_{,1}, ..., S1_{,n}$ para la primera y $S2_{,1}, ..., S2_{,n}$ para la segunda. Dos estaciones $S1_{,i}$ y $S2_{,i}$ (para $i = 1, ..., n$), hacen el mismo trabajo, pero lo hacen con costos $a1_{,i}$ y $a2_{,i}$ respectivamente,que pueden ser diferentes. Para fabricar un auto debemos pasar por n estaciones de trabajo $S_i1_{,1}$, $S_i2_{,2}$ ..., $S_in_{,n}$ no necesariamente todas de la misma linea de montaje $(i_k = 1, 2)$. Si el automovil esta en la estacion $S_{i,j}$, transferirlo a la otra linea de montaje (es decir continuar en $S_{i'},_j + 1$ con $i' != i$) cuesta $t_{i,j}$. Encontrar el costo minimo de fabricar un automovil usando ambas lineas.

$Resolucion:$

- Tenemos n estaciones totales,siendo $S_{1,n}$ y $S_{2,n}$ las dos lineas de montaje.
- Sea i la estacion actual en la que estamos, y $S_{j,i}$ las estaciones en la que estamos. Sea j = 1,2.
- Tenemos $a_{j \cdot i}$ que es el costo de la estacion que estamos, siendo ($a_{1 \cdot i}$!= $a_{2 \cdot i}$).
- Cambiar de estacion va a costar $t_{i,j}$
- El algoritmo $min\_cost(i, S_{j,i})$="Costo minimo de fabricar un automovil recorriendo ambas ensamblaje $S_{j,i}$ por completo"
- La llamada principal $min\_cost(n, S_{j,n})$="Costo minimo de fabricar un automovil recorriendo ambas estaciones del ensamblaje $S_{j,n}$ "

$$min\_cost(i, S_{j,i}) = \begin{cases} 0 & , S_{j \cdot i} = 0 \lor i = 0 \\ min(min\_cost(i-1, S_1.i-1) + a_1 + t_{i \cdot 1}), (min\_cost(i-1, S_{j \cdot i} - _1)) & , i > 0 \land (a_1 + t_{i \cdot 1} < a_2) \\ min(min\_cost(i-1, S_2.i-1) + a_2 + t_{i \cdot 2}), (min\_cost(i-1, S_{j \cdot i} - _1)) & , i > 0 \land (a_2 + t_{i \cdot 2} < a_1) \\ min(min\_cost(i-1, S_{1 \cdot i} - _1), min\_cost(i-1, S_{1 \cdot i} - _1) + (a_{1 \cdot i})) & , i > 0 \land (a_2 + t_{i \cdot 2} >= a_1) \\ min(min\_cost(i-1, S_{2 \cdot i} - _1), min\_cost(i-1, S_{2 \cdot i} - _1) + (a_{2 \cdot i})) & , i > 0 \land (a_1 + t_{i \cdot 1} >= a_2) \end{cases}$$

9

El juego $\nwarrow U \uparrow P \nearrow$ consiste en mover una ficha en un tablero de n filas por n columnas desde la fila inferior a la superior. La ficha se ubica al azar en una de las casillas de la fila inferior y en cada movimiento se desplaza a casillas adyacentes que esten en la fila superior a la actual, es decir, la ficha puede moverse a:
• la casilla que esta inmediatamente arriba,
• la casilla que esta arriba y a la izquierda (si la ficha no esta en la columna extrema izquierda).
• la casilla que esta arriba y a la derecha (si la ficha no esta en la columna extrema derecha).
Cada casilla tiene asociado un numero entero $c_{i,j}$ $(i, j = 1, ..., n)$ que indica el puntaje a asignar cuando la ficha este en la casilla. El puntaje final se obtiene sumando el puntaje de todas las casillas recorridas por la ficha, incluyendo las de las filas superior e inferior.
Determinar el maximo y el minimo puntaje que se puede obtener en el juego.
Los dos ́ultimos ejercicios, tambien pueden resolverse planteando un grafo dirigido y recurriendo al algoritmo de Dijkstra. ¿De que manera? ¿Seran soluciones mas eficientes?

$Resolucion:$

- Tenemos un tablero $T_{n,n}$ que tiene n filas y n columnas;
- Sea $T_{i,k}$ la posicion actual en la que estoy, siendo i filas y k las columnas.
- El algoritmo $max\_value(T_{i,k}) =$"Minimo valor de la sumatoria de todas las casillas recorridas por la ficha desde un T_i, k incluyendo las filas superior e inferior"

$$max\_value(T_{i,k}) = \begin{cases} 0 & , k = \\ -\infty & , i = \\ max(max\_value(T_{i,k} - 1) + c_{i,j} , max\_value(T_i + 1,_k - 1) + c_{i,j} , max\_value(T_i - 1,_k - 1) + c_{i,j} & , k > \end{cases}$$

$$min\_value(T_{i,k}) = \begin{cases} 0 & , k > 0 \\ \infty & , i = 0 \lor i \\ min(min\_value(T_{i,k} - 1) + c_{i,j} , min\_value(T_i + 1,_k - 1) + c_{i,j} , min\_value(T_i - 1,_k - 1) + c_{i,j} & , k => 0 \land \end{cases}$$

Si se puede

8.2

$Resolucion:$

- Contamos con dos lineas de ensamblaje.
- Cada linea tiene n estaciones de trabajo, (S_1,...,S_n) y (S_2,...,S_n)
- Dos estaciones hacen el mismo trabajo, pero con costos distintos(a_1i y a_2i).

- Un automovil debe pasar por n estaciones de trabajo para ser terminado, y no todas de la misma linea de ensamblaje, pasar de una estacion a otra cuesta t_ij.
- automovil(n,a_x) = "Costo minimo de fabricar el automovil usando ambas lineas de ensamblaje".
- Calculo la funcion teniendo en cuenta estaciones, y sus respectivos costos.

$$automovil(r_i, r_j, c_x) = \begin{cases} 0 & , r \\ min(automovil(r_i - 1, r_j, c_x), automovil(r_i - 1, r_j, c_x) + a_x + t_x) & , (r \\ min(automovil(r_i, r_j - 1, c_x), automovil(r_i, r_j - 1, c_x) + b_x + t_x) & , (r \\ min(automovil(r_i - 1, r_j - 1, c_x), automovil(r_i - 1, r_j, c_x) + a_x, automovil(r_i, r_j - 1, c_x) + b_x) & , (r \end{cases}$$