

- ① a) El criterio que usaremos será elegir la moneda que más va a subir de precio entre hoy y mañana.
- b) Para representar las criptomonedas usaremos un arreglo bidimensional, donde la primera dimensión estarán las monedas $1..n$ y en la segunda el precio de cada una los días $0..m$.
- c) Lo que haremos será cada día elegir la moneda que más nos hará ganar (con el criterio mencionado) y gastar todo nuestro dinero en esa moneda. Luego, al día siguiente venderemos todo y repetiremos el proceso.
- d) (En página siguiente)

Federico Virgelini

```
fun trading-magico(valor: array[1..n, 0..m] of float,  
                  D: float)  
    ret gan: float  
  
    var moneda: nat nat  
    var cantidad: float  
  
    moneda := elegir-moneda(valor, 0)  
    cantidad := D / valor[moneda, 0]  
  
    for i := 1 to m-1 do  
        gan := valor[moneda, i] * cantidad  
        moneda := elegir-moneda(valor, i)  
        cantidad := gan / valor[moneda, i]  
    od  
  
    gan := valor[moneda, m] * cantidad  
  
end fun
```

Federico Virgolini

fun elegir_moneda (valor: array[1..n, 0..m] of float,
dia: int) ret mon: nat

var max: float

max := valor[1, dia+1] / valor[1, dia]

mon := 1

for i := 2 to n do

if valor[i, dia+1] / valor[i, dia] > max then

max := valor[i, dia+1] / valor[i, dia]

mon := i

fi

od

end fun

Virgolini Federico

② ① N días

M pesos en total

Ciudades $1..N$ C_i

↳ k_i costo de hotel

↳ P_i puntuación de hotel

Puedo dormir en ~~casa~~ por \$0

Definiremos la siguiente función recursiva:

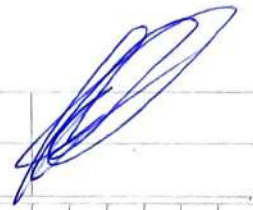
$\text{mochilero}(i, j) =$ "máxima puntuación obtenible eligiendo en que ciudades de $1..i$ se dormirán en hotel con un presupuesto j "

Por lo que la llamada principal que resolverá el problema es $\text{mochilero}(N, M)$

Definiendo la función en notación matemática:

$$\text{mochilero}(i, j) = \begin{cases} 0 & \text{si } i=0 \\ \text{mochilero}(i-1, j) & \text{si } j < k_i \\ \max\left(\text{mochilero}(i-1, j), P_i + \text{mochilero}(i-1, j - k_i)\right) & \text{si } j \geq k_i \end{cases}$$

Virgolini Federico



- ⑥ La tabla que llenará el algoritmo tendrá dos dimensiones; la primera representará las ciudades (u hoteles) que visitará que irá desde 0 hasta N . Y la segunda estará dada por el presupuesto, que irá desde 0 hasta M .

Por la forma de la función matemática, podemos ver que el algoritmo buscará resultados ya calculados siempre en la fila superior, por lo que la tabla debe ser llenada de ~~de~~ arriba hacia abajo. En la implementación que haremos, se llenará la tabla de derecha a izquierda, pero se podría hacer al revés sin problemas por lo mencionado anteriormente.

Virgolini Federico

```
fun mochilero_din(costo: array[1..N] of nat,  
                  puntos: array[1..N] of nat,  
                  M: nat) ret pun_max: nat
```

```
var table: array[0..N, 0..M]
```

```
for j := 0 to M do  
    table[0, j] = 0  
od
```

```
for i := 1 to N do  
    for j := M down to 0 do  
        if costo[i] > j then  
            table[i, j] := table[i-1, j]  
        else  
            table[i, j] := max( table[i-1, j],  
                               puntos[i] + table[i-1, j - costo[i]] )  
        fi  
    od  
od  
pun_max := table[N, M]
```

```
end fun.
```


* Desde $v+1$ hasta w

Virgolini, Federico

③ Analicemos la función s :

a) Esta función devuelve la posición del elemento mínimo del arreglo P entre las posiciones v y w (inclusive)

{PRE: $1 \leq v \leq w \leq n$ }

b) Supone que la posición mínima está en v , luego ~~recorre el arreglo~~ va recorriendo el arreglo* y si encuentra un elemento menor al de la posición almacenada la reemplaza.

c) Calcularemos el orden del algoritmo calculando el número de comparaciones que ~~hace~~ realiza.
~~el algoritmo~~

$$\begin{aligned} \text{ops}(s) &= \\ \text{ops}(y := v) + \text{ops}(\text{for } \dots \text{od}) &= \\ \text{ops}(\text{for } i := v+1 \text{ to } w \text{ do } \dots \text{od}) &= \\ \sum_{i=v+1}^w \text{ops}(\text{if } P[i] < P[y] \text{ then } \dots \text{fi}) &= \end{aligned}$$

$$\begin{aligned} \sum_{i=v+1}^w 1 &= (\text{ya que el cuerpo del if no tiene comparaciones}) \\ w - (v+1) + 1 &= \\ w - v & \end{aligned}$$

Virgo Iní Federico

~~Vemos que la complejidad del algoritmo~~

Vemos que el número de operaciones (comparaciones) que realiza el algoritmo depende de la distancia entre v y w . Por lo que el algoritmo tiene orden lineal.

d) Reemplazará el nombre de la función s y su variable y :
de retorno

~~Reemplazará~~ $s \rightarrow \text{min_pos_from_to}$
 $y \rightarrow \text{min_pos}$

Analicemos la función t :

a) Esta función devuelve la posición del elemento máximo del arreglo p entre las posiciones v y w (inclusive)
{PRE: $1 \leq v \leq w \leq n$ }

b) Supone que la posición máxima está en la posición v , luego va recorriendo el arreglo desde $v+1$ hasta w y si encuentra un elemento mayor al de la posición almacenada la reemplaza.

c) El algoritmo es análogo al anterior (solo se invierte la comparación) por lo que el orden del algoritmo es igual.

Virgilio Federico



d) Reemplazaría el nombre de la función t y su variable de retorno y por:

$t \rightarrow \text{max_pos_from_to}$

$y \rightarrow \text{max_pos.}$

Analizamos el procedimiento F :

a) Este algoritmo ordena de menor a mayor todos sus elementos; sin importar que array le pasemos (siempre y cuando sea de nat como dice la función) por lo que su precondition es True.

b) El algoritmo recorre el arreglo "de afuera hacia adentro" tomando en primer lugar el extremo izquierdo y colocando allí el menor elemento, luego toma el extremo derecho y allí coloca el mayor elemento. Esos dos elementos quedan ordenados en su lugar correspondiente, y luego repite el proceso con el arreglo $[2..n-1]$ y así hasta llegar al centro del arreglo (en algún sentido \therefore). ~~Notas~~

Virgolini Federico

Notar que si el arreglo tiene un número de elementos impar, el elemento resultante en el centro tendrá a su izquierda elementos menores a él y a su derecha elementos mayores a él por lo que también resulta ordenado.

c) Veremos el orden del algoritmo calculando el número de comparaciones que realiza

$$\text{ops}(r) =$$

$$\text{ops}(\text{for } i=1 \text{ to } n/2 \text{ do... od}) =$$

$$\sum_{i=1}^{n/2} \text{ops}(\text{swap}(P, i, s(P, i, n-i+1)), \text{swap}(P, n-i+1, t(P, i+1, n-i+1))) =$$

Swap no usa comparaciones, pero si los argumentos que usa, por lo que debemos sumarlos (y ya los calculamos antes)

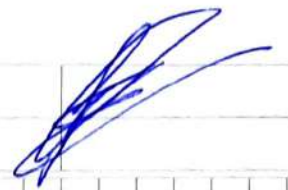
$$\sum_{i=1}^{n/2} \text{ops}(s(P, i, n-i+1)) + \text{ops}(t(P, i+1, n-i+1)) =$$

$$\sum_{i=1}^{n/2} (n-i+1-i) + (n-i+1-i-1)$$

$$\sum_{i=1}^{n/2} 2n - 4i - 1 =$$

(siguiente pag)

Virgolini Federico



$$2n \sum_{i=1}^{n/2} 1 - 4 \sum_{i=1}^{n/2} i - \sum_{i=1}^{n/2} 1 =$$

$$2n \cdot \frac{n}{2} - 4 \frac{n/2 (n/2 + 1)}{2} - n/2 =$$

$$n^2 - \frac{n^2}{2} - n - n/2 =$$

$$\frac{n^2}{2} - \frac{3}{2}n$$

Por lo que el orden del algoritmo es de n^2

d) Reemplazaría el nombre del procedimiento 'r' por 'ord. afuera-adentro'.

④②

Implement List of T where

*type List of T = tuple

elem: array[1..N] of T

size: nat

end tuple

fun empty() ret l: List of T

l.size := 0

end fun

~~fun add~~

proc add l (in e: T, in/out l: List of T)

for i := l.size downto 1 do

l.elem[i+1] := l.elem[i]

od

l.elem[1] := e

l.size := l.size + 1

end proc.

fun length(l: List of T) ret n: nat

n := l.size

end fun.

Virgolini Federico

```
proc tail (in/out l: List of T)
  for i:=1 to l.size-1 do
    l.elem[i] := l.elem[i+1]
  od
  l.size := l.size - 1
end proc.
```

```
proc concat (in/out l: List of T, in l0: List of T)
  for i:=l.size+1 to l.size+l0.size do
    l.elem[i] := l0.elem[i-l.size]
  od
  l.size := l.size + l0.size
end proc.
```

```
proc drop (in/out l: List of T, in n: nat)
  for i:=n+1 to l.size do
    l.elem[i-n] := l.elem[i]
  od
  l.size := l.size - n
end proc.
```

b) Si, la representación elegida tiene la limitación que solo podemos almacenar N elementos en nuestra lista, por lo que `add`, `concat` y el constructor `addl` deben tener pre condiciones extra para chequear que el/los elementos "quepan".

Virgolini Federico



c) La operación tail implementada tiene orden $l.tail$, es decir orden lineal. Podemos reemplazar la estructura de datos para que esta operación tenga orden constante como lo es en una lista enlazada por ejemplo, que tiene la siguiente estructura:

Implement List-en of T where

type List-en of T = Pointer to Node of T

type Node of T = tuple

elem: T

next: ~~Pointer~~ Pointer to Node of T

end tuple

y la operación tail resultaría:

proc tail (in/out l: List of T)

var p: Pointer to Node of T

p := ~~l~~

l := l.next

free (p)

end proc

Notar que esta nueva versión es de orden constante.

d)

```
fun combiner ( l1: List of int,  
               l2: List of int )  
  ret l3: List of int.
```

```
var l1-aux, l2-aux: List of int
```

```
l1-aux := copy-list(l1)
```

```
l2-aux := copy-list(l2)
```

```
while not is_empty(l1-aux) do
```

```
  addr(l3, head(l1-aux))
```

```
  tail(l1-aux)
```

```
  addr(l3, head(l2-aux))
```

```
  tail(l2-aux)
```

```
od
```

```
end fun
```

```
  destroy(l1-aux)
```

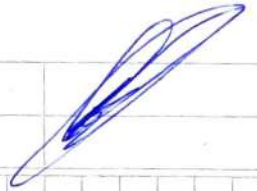
```
  destroy(l2-aux)
```

observaciones

- como precondition, ambas listas deben tener el mismo tamaño

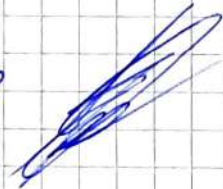
- El algoritmo es para listas cuya primera posición es cero (En caso que sea 1 se debe invertir el orden en el que se agregan los elementos)

Virgolini Federico



Por la presente declaro que la resolución de este examen es obra de mi exclusiva autoría y respetando las pautas y criterios fijados en los enunciados. Asimismo declaro conocer el régimen de infracción de los estudiantes cuyo texto ordenado se encuentra en el apéndice de la Res. Rec 1554/2018

Virgolini Federico
42 692 460



REPUBLICA ARGENTINA - MERCOSUR
REGISTRO NACIONAL DE LAS PERSONAS
MINISTERIO DEL INTERIOR Y TRANSPORTE

Apellido / Surname
VIRGOLINI

Nombre / Name
FEDERICO NICOLÁS

Sexo / Sex
M

Nacionalidad / Nationality
ARGENTINA

Ejemplar
A

Fecha de nacimiento / Date of birth
06 JUL 2000

Fecha de emisión / Date of issue
08 FEB 2015

Fecha de vencimiento / Date of expiry
08 FEB 2030

Documento / Document
42.692.460

FRM IDENTIFICADOR SIGNATURE

