ISIS 1105 Diseño de Algoritmos
Semestre 2014-2
Prof. Rodrigo Cardoso
Tarea 2

Para entregar por Sicua+, antes de Octubre 20, 10:00

1 Subarreglo ascendente más largo (25/100)

Se quiere construir un programa que reciba un arreglo de números naturales y encuentre la longitud del subarreglo ascendente más largo. Use la siguiente notación:

`lsaml(i)` ≈ "longitud del subarreglo ascendente más largo en `b[0..i-1]`", `0≤i≤n`.

`lsamlf(i)` ≈ "longitud del subarreglo ascendente más largo en `b[0..i-1]`, que termina en `b[i-1]`", `0≤i≤n`.

```
[Ctx C: b[0..n-1]:nat

 INIC;

 {Inv P : 0≤i≤n ∧ p = lsaml(i) ∧ q = lsamlf(i)}
 {Cota t: n-i}

 do … od

 {R: p = lsaml(n)}
 ]
```

1a Explique qué técnica pudo haberse utilizado para proponer el invariante `P`.

Dos posibles respuestas:

(1) Cambiar una constante por una variable (`P' ≡ R[n:= i] ∧ 0≤i≤n`) más un fortalecimiento
`P ≡ P' ∧ q=lsamlf(i)`

(2) Programación dinámica (recurrencia para `lsaml` basada en la de `lsamlf`)

```
lsaml(0)    = 0
lsaml(i+1)  = max(lasml(i),lasmlf(i+1))   , 0<i<n
lasmlf(0)   = 0
lasmlf(i+1) = if b[i]≥b[i-1] then 1+lsamlf(i) else 1 fi, 0<i<n
```

[10/10]

1b Desarrolle un algoritmo que satisfaga la especificación indicada (operación básica: suma).

```
[ Ctx C: b[0..n-1]:nat

  i,p,q:= 0,0,0;

  {Inv P : 0≤i≤n ∧ p = lsaml(i) ∧ q = lsamlf(i)}
  {Cota t: n-i}

  do  i≠n   →    if b[i]≥b[i-1] then q:= 1+q
                 else q:= 1
                 fi;
                 p = max(p,q);
                 i:= i+1
  od
```

```
  {R: p = lsaml(n)}
 ]
```

[10/10]

1c Estime las complejidades temporal y espacial de su solución.

El arreglo se procesa de izquierda a derecha, pasando una vez por cada elemento. El costo del proceso por elemento es θ(1). Es decir:

$$T(n) = \theta(n)$$

Como espacio adicional se usan variables enteras i,p,q. Por tanto:

$$S(n) = \theta(1).$$

[5/5]

2 Búsqueda de preimagen (25/100)

Considere la función f: a..b × c..d → **int** definida como f(x,y) = x^2-y.
Sea H un valor entero dado.
Escriba un programa que decida si existe una pareja (i,j) en el dominio de f, tal que f(i,j)=H.

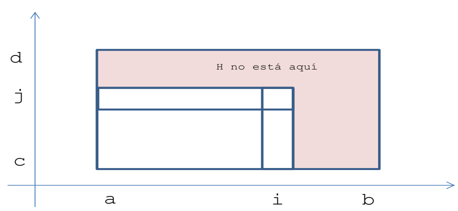
2a Especifique el problema (Contexto, Pre-, Poscondición).

```
Ctx: f: a..b × c..d → int ∧ f(x,y) = x²-y ∧ H:int
Pre: true
Pos: existePareja ≡ (∃i,j| (i,j)∈ a..b × c..d: f(i,j)=H)}
```

[5/5]

2b Proponga un invariante P y una cota t para desarrollar el programa solución con un ciclo.

```
Inv P: a≤i≤b ∧ c≤j≤d
       ∧ ¬(∃i1,j1| (i1,j1)∈ (i+1..b×c..d)∪(a..b×j+1..d): f(i,j)=H)
```




```
Cota t: (i-icent)*(j-jcent)
```

[5/5]

2c Escriba código que satisfaga lo anotado en **2a** y **2b**.

```
[ Ctx: f: a..b × c..d → int ∧ f(x,y) = x²-y ∧ H:int
  Pre: true
  Pos: existePareja ≡ (∃i,j| (i,j)∈ a..b × c..d: f(i,j)=H)}
⎡
  i,j:= d,b; icent:= c-1; jcent:= a-1;

  {Inv P}

  do i≠icent ∧ j≠jcent  →     u:= i*i-j;
                              if   u > H then i:= i-1
                              elsf u < H then j:= j-1
```

```
                                  else  icent:= i
                                  fi
  od
⌋
```

[5/5]

2d Calcule la complejidad temporal de su solución (operación básica: comparación).

Es una búsqueda en silla en la que, en cada iteración se elimina una fila o una columna del dominio. El peor caso es cuando el valor no está o cuando `H` está en el último `(i,j)` que se chequea.

$T(a,b,c,d,H) = \theta((b-a)+(d-c))$

[5/5]

2e Estime la complejidad espacial de su solución.

Solo se usan las variables `i,j,u,icent,jcent`.

$S(a,b,c,d,H) = \theta(1)$

[5/5]

3 Cambio de monedas (35/100)

Sea $A_n = \{a_1,a_2,\ldots,a_n\}$ un conjunto de valores de monedas, todos diferentes. Suponga que $a_i \in$ **nat**, $1 \le i \le n$, y que $1=a_1<a_2<\ldots<a_n$. Dado $C \in$ **nat**, el problema del *cambio de monedas* consiste en encontrar el mínimo número de monedas de valores en A_n cuyo valor total sea C. Se supone que hay una cantidad ilimitada de monedas de cada valor.

3a Construya su solución de acuerdo con la "receta para programación dinámica" que se vio en clase (lenguaje, recurrencia, …) para resolver el problema.

Lenguaje

$mnm(x,i) \approx$ "mínimo número de monedas para construir x con A_i", $0 \le x \le C$, $1 \le i \le n$.

$mnm: 0..C \times 1..n \rightarrow$ **nat**

$mnm(C,n) = ?$

[5/5]

Recurrencia

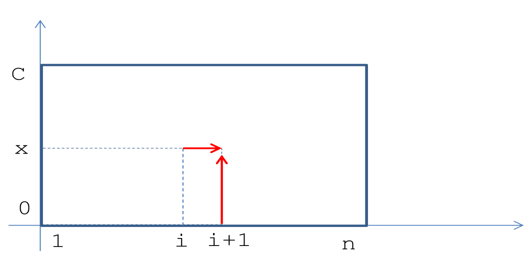
$mnm(x,1) = x$, si $0 \le x \le C$

$mnm(x,i+1) = mnm(x,i)$, si $0 \le x < a_{i+1}$, $1 < i \le n$

$mnm(x,i+1) = \min\{mnm(x,i), 1+mnm(x-a_{i+1},i+1)\}$, si $0 < a_{i+1} \le x \le C$, $1 < i \le n$

[5/5]

Diagrama de necesidades



[5/5]

Estructura de datos + Invariante

```
MNM[0..C]:nat

Inv: 2≤i≤n+1 ∧ 0≤x≤C ∧ MNM[0..x-1]=mnm(0..x-1,i) ∧ MNM[x..C]=mnm(x..C,i-1)
```

[5/5]

Solución

```
for x:=0 to C → MNM[x] = x rof;

i,x:= 2,0;

{Inv: 2≤i≤n+1 ∧ 0≤x≤C ∧ MNM[0..x-1]=mnm(0..x-1,i) ∧ MNM[x..C]=mnm(x..C,i-1)}
{Cota: n-i}

do i≠n+1 → if    a_i>x  then skip
           else  MNM[x]:= min(MNM[x],1+MNM[x-a_i])
           fi;
           if x≠C        then x:= x+1
                         else i,x:=i+1,0
           fi
od
```

[5/5]

3b Estime complejidades temporal (operación básica: asignación) y espacial de su solución para 3a.

El dominio de mnm se recorre totalmente (área: nC). Cada iteración cuesta $\theta(1)$ asignaciones. Es decir:
$T(A_n,C) = \theta(nC)$

Se usa un arreglo de tamaño C, más variables i,x. O sea:
$S(A_n,C) = \theta(C)$.

[5/5]

3c Indique qué añadir a su solución de **3a** para encontrar cómo se consigue el valor óptimo. Calcule los incrementos en costos asociados a este requerimiento.

Deben recordarse las decisiones. Si se define una función
d: 0..C × 1..n → **bool**
tal que
d(x,i) ≡ "el óptimo para lograr mnm(x,i) requiere una moneda a_i"

Se usa una matriz

D[0..C,1..n]: **bool**

tal que, el invariante se fortalezca exigiendo que
(∀xy,j| 0≤y≤C ∧ 1≤j<i: D[y,j] ≡ d(y,j)) ∧ (∀y| 0≤y<x: D[y,i] ≡ d(y,i))

Al final, se puede calcular recursivamente el conjunto de monedas que dan el óptimo. Sea b[1..n]:**nat**.El siguiente algoritmo calcula en b[i] cuántas monedas de a_i deben incluirse:

```
for i:=1 to n → b[i]:= 0 rof;
```

```
i,x:= n,C;

do i≠0      →     if D[i,x]   then b[i]:= b[i]+1;
                                    x:= x-a_i
                  else i:= i-1
                  fi
od
```

El costo adicional es

$T'(A_n,C) = \theta(nC) + \theta(C)$ // construir D + construir b

$= \theta(nC)$

$S'(A_n,C) = \theta(nC) + \theta(n)$ // matriz D + arreglo b

$= \theta(nC)$

[5/5]

4 Modelaje con grafos (15/100)

Modele con un grafo el proceso de construcción paso a paso de una solución para el problema de *cambio de monedas* planteado en **3**. Estime la complejidad temporal de un tal proceso (operación básica: incluir en el cambio una moneda de valor $a \in A$).

Sea $G(V,\rightarrow)$ grafo tal que

$$V = \{(x,b_1,b_2,\ldots,b_n) \mid x \in \mathbf{nat},\ b_i \in \mathbf{nat},\ 1 \le i \le n\} = \mathbf{nat}^{n+1}$$

$$(x,b_1,b_2,\ldots,b_i,\ldots,b_n) \rightarrow (x-a_i,b_1,b_2,\ldots,b_i+1,\ldots,b_n) \quad , \text{ si } 0<a_i \le x,\ 1 \le i \le n.$$

[5/5]

En realidad, interesa el subgrafo de G que contiene un vértice inicial
$v_0 = (x,0,0,\ldots,0)$

más todos los vértices alcanzables desde este vértice, i.e., vértices v tales que $v_0 \rightarrow^* v$.

Este subgrafo no es infinito, porque al avanzar en un camino el valor de la primera componente va disminuyendo y no puede ir más allá de 0.

[5/5]

Como problema de grafos, se quiere encontrar una ruta de longitud mínima desde el vértice inicial hasta un vértice de la forma $(0,b_1,b_2,\ldots,b_n)$.

[5/5]