ISIS 1105 Diseño de Algoritmos
Semestre 2011-10 - Tarea No. 2
Prof. Rodrigo Cardoso
Fecha de entrega: Abril 8, 2011, 12:00 (por Sicua+)

---

1 **[30/100]**

Considere el problema de calcular la raíz cúbica entera de un número natural `n` de acuerdo con la siguiente especificación:

```
[  Ctx C: n: nat
   {Q: true}
        ...
        {Inv P :  a = (b+1)^2  ∧  c= (b+1)^3  ∧  b^3 ≤ n }
        {cota t:  n-c}
   do  ...  od
      {R1: b^3 ≤ n < (b+1)^3 }
   {R2: b ≤ ∛n < b+1 }
   {R : b = ⌊∛n⌋ }
]
```

1a **[15/30]** Escriba código GCL que siga la especificación indicada. Dentro del código use únicamente operaciones de suma, resta y multiplicación.

Variante 1a-1:

```
[ Ctx C: n: nat
  {Q: true}
       a,b,c:= 1,0,1;
                                                              [3/15]
       {Inv P :  a = (b+1)^2  ∧  c = (b+1)^3  ∧  b^3 ≤ n }
       {cota t:  n-c}
   do n ≥ c
                                                              [2/15]
        →    a, b, c:= a+2*b+3, b+1, c+3*a+3*b+4
                                                              [10/15]
   od
  {R1: b^3 ≤ n < (b+1)^3 }
  {R2: b ≤ ∛n < b+1 }
  {R : b = ⌊ ∛n ⌋ }
]
```

Variante 1a-2:

```
[ Ctx C: n: nat
  {Q: true}

       a,b,c:= 1,0,1;
                                                              [3/15]
       {Inv P :  a = (b+1)^2  ∧  c = (b+1)^3  ∧  b^3 ≤ n }
       {cota t:  n-c}
   do n ≥ c
                                                              [2/15]
        →    c:= c+3*a+3*b+4;
             a:= a+2*b+3;
```

```
                b:= b+1
```
[10/15]

```
    od
  {R : b = ⌊ ∛n ⌋ }
]
```

Variante 1a-3:

Otro orden de actualización de las variables `a,b,c`, pero cuidando que el invariante se mantenga, por ejemplo:

```
[ Ctx C: n: nat
  {Q: true}
      a,b,c:= 1,0,1;
```
[3/15]

```
      {Inv P :  a = (b+1)^2  ∧  c = (b+1)^3  ∧  b^3 ≤ n }
      {cota t:  n-c}
   do n ≥ c
```
[2/15]

```
        →     a:= a+2*b+3;
              b:= b+1;
              c:= c+3*a-3*b-2;
```
[10/15]

```
    od
  {R : b = ⌊ ∛n ⌋ }
]
```

1b [8/30] Indique qué técnica(s) puede(n) explicar la definición del invariante `P` a partir de la especificación dada.

Hay dos técnicas involucradas:

- Eliminar una conjunción de `R1`. Se elimina la conjunción `n < (b+1)^3`

  `P1 : b^3 ≤ n`

  [4/8]
- `P1` se fortalece hasta `P`, para evitar el cálculo de potencias dentro del código.

  [4/8]

1c [7/30] Cuente asignaciones como operaciones básicas y estime el orden de complejidad de su solución (como `θ(...)`).

El algoritmo calcula las potencias cubicas desde `0` hasta `n`. Así: `T(n) = θ(∛n)`.

[7/7]

2 [30/100]

Una función `f:[a,b] → real` es *unimodal* si es continua y existe un punto `m`, con `a≤m≤b`, tal que `f` es creciente en `[a,m]` y decreciente en `[m,b]`. El predicado `unimodal(f,a,m,b)` denotará esta situación. Dado un número real `eps>0`, se quiere escribir un programa que determine un valor de `x∈[a,b]` que aproxime a `c` con un error menor que `eps`.

Estime la complejidad de su solución, contando asignaciones.

AYUDA: El problema es una búsqueda. Use la técnica de apretar el cerco para proponer un invariante. Asegúrese de aprovecharse de las hipótesis para mejorar la eficiencia.

Se propone el invariante:

`Inv P : a ≤ p ≤ c ≤ q ≤ b  ∧  x = (p+q)/2`

[10/30]

que establece un cerco para `c` definido por el intervalo `[p,q]` (números reales).

La cota debe ser tal que, al final, `x` y `c` estén a una distancia menor que `eps`. Es natural proponer
`Cota t: |x-c| - eps`

[5/30]

Nótese que `p≤c≤q` y por tanto, `-q≤-c≤-p`. También `p≤x≤q`, de modo que, sumando miembro a miebro las dos últimas desigualdades, el invariante dice que siempre debe valer

$$p-q \le x-c \le q-p$$
$$\equiv$$
$$|x-c| \le q-p$$

De modo que si se propone como guarda `q-p≥eps`, si esto deja de cumplirse se tendrá la condición deseada.

[5/30]

Para mantener el invariante se condideran los puntos a 1/3 y 2/3 del intervalo de incertidumbre y se toma una decisión de refinir `p` o `q` según los valores de la función en esos puntos.

La solución propuesta es:

```
[Ctx C: unimodal(f,a,c,b) ∧ eps>0
   p,q:= a,b;
   {Inv P : a ≤ p ≤ c ≤ q ≤ b  ∧  x = (p+q)/2 }
   {Cota t: |x-c| - eps }
   do q-p ≥ eps →   u,v:= p+(q-p)/3,q-(q-p)/3;
                    if f(u)≤f(v) → p:= u
                    [] f(u)>f(v) → q:= v
                    fi;
                    x:= (p+q)/2
   od
   {Pos R: |x-c| < eps}
]
```

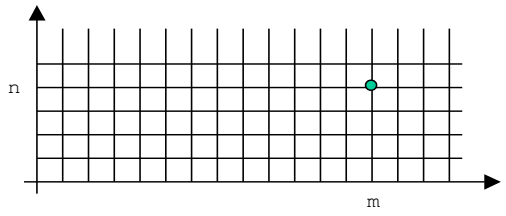
[10/30]

Para la complejidad, cada iteración reduce el intervalo a 2/3 del tamaño antes de iterar. Es decir:
$T(b,a) = \theta(\log_{3/2}(b-a))$

[+10/30]

3 [40/100]

Un agente secreto se encuentra en el origen (posición `(0,0)`) de una ciudad cuadriculada y desconocida, cuyo plano tiene la forma:



La ciudad tiene muchas calles y carreras ("muchas" ≈ infinitas!). El agente sabe que su contacto se encuentra en una cierta posición `(m,n)`. Aunque ignora las coordenadas exactas, su contacto le ha comunicado que desde el origen a su posición hay exactamente `r` maneras de llegar desde el origen,

caminando siempre hacia la izquierda o hacia arriba, y que, además, $m \geq n > 1$. Por ejemplo, si el contacto está en $(3,2)$, le ha comunicado que está en un punto al que se puede llegar de 10 maneras.

3a [30 puntos] Utilice programación dinámica para explicar cómo puede el agente averiguar (m,n), las coordenadas de su contacto. Para empezar, use como lenguaje

$nc(i,j)$ = "maneras de ir desde el origen a (i,j)", para $i \geq j \geq 0$.

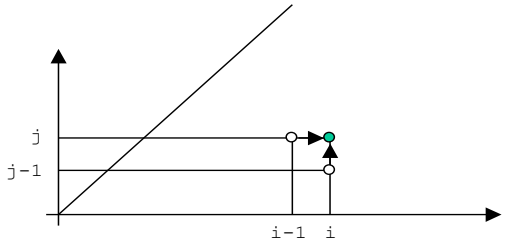
Complete los demás pasos para construir su solución (recurrencia, diagrama de necesidades, invariante). No es necesario que escriba su algoritmo.

Recurrencia:

$$nc(i,j) = 1 \quad \text{, si } i \geq j = 0$$
$$= nc(i-1,j) + nc(i,j-1) \quad \text{, si } i \geq j > 0$$

[10/30]

Diagrama de necesidades:

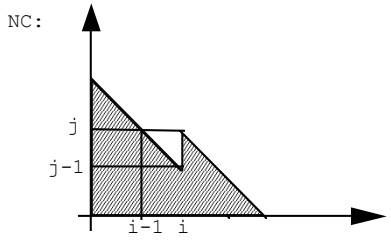


El diagrama sugiere un recorrido del cuadrante por diagonales de pendiente -1 que se alejan del origen. Las diagonales pueden ser recorridas ascendentemente, i.e., empezando la diagonal i-sima en $(i,0)$. En algún momento se tendrá $nc(i,j)=r$, para ciertos $i \geq j > 0$, según lo prometido.

Entonces: $(m,n) = (i,j)$.

[10/30]

Invariante: (se requiere una estructura de datos NC, no acotada, pero finita)



[10/30]

Variaciones

El cuadrante se puede recorrer, por ejemplo, con cuadrados de lado $1, 2, \ldots$, cuya esquina inferior izquierda esté en el origen. Para calcular un cuadrado a partir del anterior se calculan los lados izquierdo (excepto la esquina derecha) y superior. En este caso el contacto se encuentra en el cuadrado de lado $\max(m,n)=m$. Las complejidades son $O(m^2)$. La espacial puede mejorarse, guardando los lados derecho y superior del cuadrado, i.e., $O(m+m) = O(m)$.

Como en la primera solución, se puede aprovechar la simetría de nc para reducir a la mitad los cálculos necesarios.

3b [10 puntos] Estime las complejidades espacial y temporal de su algoritmo en términos de `m` y `n`. Para lo temporal, use la asignación como operación básica.

La posición del contacto se encuentra en la diagonal `m+n`. La diagonal `i` tiene `i+1` puntos en los que debe ser calculada la función `nc`.

$$S(m,n) = O(1+2+\ldots+(m+n)) = O((m+n)^2/2) = O(m^2)$$

Algo mejor: bastan dos vectores que guarde las dos últimas diagonales. Así:

$$S(m,n) = O(m+n) = O(m)$$

El tiempo es proporcional al área barrida:

$$T(m,n) = O((m+n)^2/2) = O(m^2)$$

Hay mejoras posibles. Observando que `nc(x,y)=nc(y,x)`, la mitad superior de cada diagonal puede determinarse conociendo la mitad inferior. Más aun: no debería calcularse esa parte superior, porque la respuesta está en la parte inferior de la diagonal. Sin embargo, debe calcularse hasta la diagonal. Puede aprovecharse el hecho de que

$$nc(i,i) = nc(i-1,i)+nc(i,i-1) = 2\ nc(i,i-1).$$

[10/30]