ISIS 1105 Diseño y Análisis de algoritmos
Semestre 2014-10 Parcial 2
Octubre 21, 2014
Profs. Rodrigo Cardoso / Germán Sotelo

1 (30/100)

Dos arreglos `f,g[0..n-1]:int`, se pueden *comparar lexicográficamente* y establecer si

$f \prec g \equiv (\exists k \mid 0 \le k < n : f[k] < g[k] \wedge (\forall i \mid 0 \le i < k : f[i] = g[i]))$

$f \succ g \equiv g \prec f$

$f = g \equiv \neg(f \prec g \vee f \succ g)$

Considere la siguiente especificación:

Ctx C: `f,g[0..n-1]:int`
Pre Q: `true`
Pos R: $(a \equiv f \prec g) \wedge (b \equiv f = g) \wedge (c \equiv f \succ g)$

1a (20/30) Escriba y anote (i.e., incluya aserciones, invariantes, cotas, etc.) un algoritmo que solucione el problema.

```
⌈
   k:= 0;

   {Inv  P: 0≤k≤n ∧ (∀i| 0≤i<k : f[i]=g[i])}
   {Cota t: if f[k]=g[k] then 0 else n-k fi}

   do k≠n ∧ f[k]=g[k] → k:= k+1 od;

   {R1: 0≤k≤n ∧ (∀i| 0≤i<k : f[i]=g[i]) ∧ (k=n ∨ (k≠n ∧ f[k]≠g[k])}

   if    k=n   →    a,b,c:= false,true,false
   []    k≠n   →    if    f[k]<g[k]   → a,b,c:= true,false,false
                    []    f[k]>g[k]   → a,b,c:= false,false,true
                    fi
   fi

   {Pos R: (a ≡ f≺g) ∧ (b ≡ f=g) ∧ (c ≡ f≻g)}
⌋
```

[Inv: 10/20]
[Cota: 3/20]
[Guarda: 2/20]
[if : 5/20]

Variante:

Paradigma BLI para buscar un índice $k \in 0..n-1$ tal que $f[k] \neq g[k]$:

Al terminar este paradigma se procede como en la solución ya explicada desde `R1` en adelante:

```
⌈
   k:= 0;
   kcent:= n;

   do k≠kcent  →    if    f[k]=g[k]   → k:= k+1
```

```
                    []    f[k]≠g[k]   → kcent:= k
                    fi
    od

    {R1: 0≤k=kcent≤n ∧ (∀i| 0≤i<k : f[i]=g[i]) ∧ (k=n ∨ (k≠n ∧ f[k]≠g[k])}

    if    k=n   →    a,b,c:= false,true,false
    []    k≠n   →    if    f[k]<g[k]   → a,b,c:= true,false,false
                     []    f[k]>g[k]   → a,b,c:= false,false,true
                     fi
    fi

    {Pos R: (a ≡ f<g) ∧ (b ≡ f=g) ∧ (c ≡ f>g)}
⌋
```

[Anuncio de Paradigma BLI: 5/20]
[Código BLI: 10/20]
[if : 5/20]

1b (5/30) Si ha usado invariantes y cotas, explique qué técnica(s) ha usado para desarrollarlo(s). Si ha usado paradigmas, explique cuáles y justifique su aplicabilidad.

El desarrollo de P responde a la técnica de eliminar una conjunción (la que se usa como guarda negada).

[3/5]

La cota `t` corresponde a lo que falta por procesar. Si se encuentra un elemento diferente se frena inmediatamente.

[2/5]

Variante con Paradigma BLI:

Espacio de búsqueda inicial: `E = 0..n-1`
Primer elemento: `0`
Sucesor: `s.k = k+1`
Último elemento: `n (= s(n-1))`
Predicado de búsqueda: `f[k]≠g[k]`

[5/5]

1c (5/30) Considere como operación básica la comparación con el elemento x. Estime y explique los costos en tiempo y en espacio del algoritmo propuesto.

La búsqueda de un elemento diferente cuesta $\theta(n)$ comparaciones. La determinación de a,b,c c uesta $\theta(2)$ comparaciones. En resumen:

$T(n) = \theta(n+2)$
$= \theta(n).$

[3/5]

No se usan estructuras de datos auxiliares de tamaño dependiente de n, es decir:

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

[2/5]

2 (40/100)

Dado `b[0..n-1]:int`, se quiere determinar la longitud del subarreglo más largo de `b` que contiene solo `0`'s.

2a (10/40) Especifique formalmente el problema que se quiere resolver.

```
Ctx: b[0..n-1]:int
Pre: true
Pos: r = xz(n)
```

donde:

`xz(k) =` "longitud del subarreglo de `0`'s más largo en `b[0..k-1]`"

`= (max p,q| 0≤p<q<k ∧ z(p,q): q-p)`

`z(p,q) ≡ (∀i| p≤i<q : b[i]=0)  , 0≤p<q<n.`

[10/10]

2b (10/40) Plantee un invariante `P` y una cota `t`.

`Inv P : 0≤k<n ∧ r = xz(k) ∧ s = xzf(k)`

[8/10]

`Cota t: n-k`

[2/10]

donde:

`xzf(k)` = "longitud del subarreglo de `0`'s más largo en `b[0..k-1]`, que incluye `b[k-1]`"

`= (max i| 0≤i<k ∧ z(i,k) : k-i)`

2c (10/40) Desarrolle una solución de acuerdo con su especificación. Explique por qué se mantiene el invariante.

```
⌈
   k,r,s:= 0,0;

   {Inv P : 0≤k<n ∧ r = xz(k) ∧ s = xzf(k) }
   {Cota t: n-k}

   do k≠n   →    if    b[k]=0      → s:= s+1
                 []    b[k]≠0      → s:= 0
                 fi;
                 if    s>r         → r:= s
                 []    s≤r         → skip
                 fi;
                 k:= k+1
   od

  {Pos R: r = xz(n)}
⌋
```

[8/10]

El invariante se mantiene porque el cuerpo del ciclo actualiza `s` y calcula el nuevo `r` como el más largo entre el que se tenía calculado y el nuevo `s`, avanzando hacia terminación.

[2/10]

2d (10/40) Estime costos de tiempo y espacio. Operación básica: asignación. Explique sus respuestas.

El arreglo de longitud n se recorre una vez, y en cada elemento se usan $\theta(1)$ asignaciones. Por tanto $T(n) = \theta(n)$.

[5/10]

Se usan $\theta(1)$ variables auxiliares de tamaño 1. Así:
$S(n) = \theta(1)$.

[5/10]

3 (40/100)

Un repaso sobre composición funcional: dadas dos funciones $g:A \to B$, $h:B \to C$, la función compuesta $h \bullet g:A \to C$ se define de manera que $(h \bullet g)(x) = h(g(x))$, para $x \in A$. El operador $\bullet$ suele omitirse, de modo que $gh = g \bullet h$. La composición funcional es asociativa.

Suponga un conjunto de n funciones de valor entero $f_i:0..m_i-1 \to 0..m_{i+1}-1$, $0 \le i < n$, representadas con arreglos $F_i[0..m_i-1]:\mathbf{int}$, con $F_i[x] = f_i(x)$, $0 \le x < m_i$.

Considere el problema de calcular de manera óptima la función resultante de componer las n funciones

$$f_{n-1}f_{n-2} \ldots f_1f_0 : 0..m_0-1 \to 0..m_n-1$$

representando ésta con un arreglo $F[0..m_0-1]:\mathbf{int}$, con $F[x] = f_{n-1}f_{n-2} \ldots f_1f_0(x)$, $0 \le x < m_0$. Como operación básica considere $\min(.,.)$, el cálculo del mínimo de dos valores.

3a Establezca una notación para plantear una solución que utilice la metodología de desarrollo de algoritmos de programación dinámica estudiada en clase.
AYUDA: Diseñe una notación para componer las funciones entre dos índices i y j.

Se puede definir:
$T(i,j) \approx$ "Tiempo mínimo para calcular $f_jf_{j-1} \ldots f_{i+1}f_i$", $0 \le i \le j < n$.

[5/10]

$T(0,n-1) = ?$

[3/10]

$T: 0..n-1 \times 0..n-1 \to \mathbf{nat}$.

[2/10]

3b (20/40) Usando la notación establecida, continúe con los pasos de la metodología de desarrollo de algoritmos de programación dinámica estudiada. No es necesario que se escriba el algoritmo.

Considérense las funciones $g:0..p-1 \to 0..q-1$, $h:0..q-1 \to 0..r-1$ representadas con arreglos G y H como en el enunciado. La composición hg debe calcularse de modo que

$$(hg)(x) = h(g(x)) = H[G[x]] \quad , 0 \le x < q$$

de modo que cada elemento del dominio $0..p-1$ se calcula en $\theta(1)$. Puesto que son p elementos, la composición gasta $\theta(p)$. Resumiendo: la composición de dos funciones es $\theta(p)$, siendo p el tamaño del dominio de la primera función.

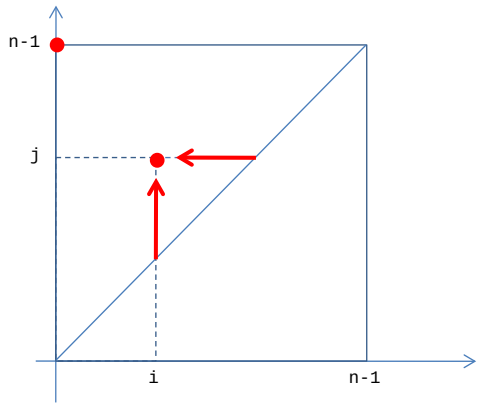
[3/20]

Recurrencia

$$
\begin{aligned}
T(i,i) &= 0 && , \ 0 \le i < n \\
T(i,j) &= (\min k \mid i \le k \le j : T(i,k) + T(k+1,j) + m_i) && , \ 0 \le i < j \le n \\
&= m_i + (\min k \mid i \le k \le j : T(i,k) + T(k+1,j)) && , \ 0 \le i < j \le n
\end{aligned}
$$

[5/20]

Diagrama de necesidades



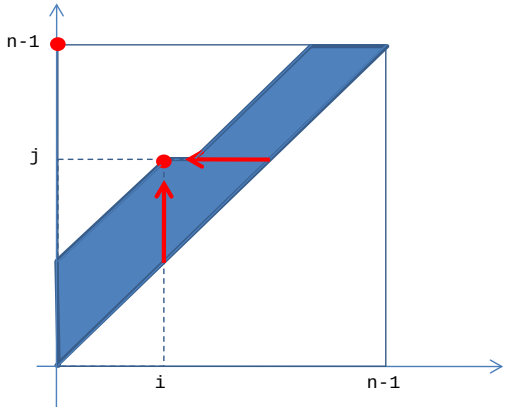
[5/20]

Estructura de datos + Invariante

Como estructura de datos adicional se define una matriz `M[0..n-1,0..n-1]:`**`nat`** que se usa -en su diagonal superior- para guardar los valores de `T`, de modo que `M[i,j]=T(i,j), 0≤i≤j<n`.

El invariante `P` corresponde a un paso intermedio del proceso de llenar la parte superior de `M` por diagonales, del centro hacia la esquina superior izquierda, y cada diagonal de abajo arriba:

`Inv P:`



[4/20]

`Cota t: "Tamaño del área blanca en el dibujo, en el triángulo superior"`

El área blanca en el triángulo superior corresponde a los valores del dominio aún no calculados. Las iteraciones terminan cuando se haya calculado `M[0,n-1]`.

[3/20]

3c (10/40) Estime los costos temporal y espacial correspondientes (como $\theta(\ldots)$). Explique su respuesta.

El costo temporal corresponde al tamaño original del área triangular que mide la cota, el cual es $\theta(n^2/2)$ multiplicado por el costo del cálculo correspondiente a un elemento `(i,j)`. En general, `M[i,j]` se puede calcular con $\theta(j-i)$ operaciones `min`. En el peor caso `M[i,j]` gastará $\theta(n)$ operaciones `min`.

Resumiendo:

$$T(0,n-1) = \theta(n^2/2) * \theta(n)$$
$$= \theta(n^3)$$

[5/10]

Claramente, el espacio adicional necesario requiere la matriz `M`:

$$S(0,n-1) = \theta(n^2/2)$$
$$= \theta(n^2)$$

[5/10]