ISIS 1105 Diseño y Análisis de algoritmos
Semestre 2014-10 Parcial 2
Abril 10, 2014
Prof. Rodrigo Cardoso

1 (30 puntos)

Desarrolle un algoritmo para calcular una búsqueda binaria en un arreglo de números enteros ordenados ascendentemente. Considere la siguiente especificación:

```
Ctx C: 0<n  ∧  b[0..n-1]:int ∧ b↑ ∧ x:int ∧ b[0]≤x<b[n-1]
Pre Q: true
Pos R: 0≤i<n-1 ∧ b[i]≤x<b[i+1]
```

1a (10/30) Describa un invariante y una función cota para un algoritmo de búsqueda binaria.

```
inv  P: 0≤i≤j<n ∧ b[i]≤x<b[j]
cota t: j-i-1
```

[10/30]

1b (5/30) Explique qué técnica ha usado para desarrollar el invariante.

Técnica: Cambiar constante por variable. Más exactamente: cambiar la expresión dependiente `i+1` por la variable `j`.

[5/30]

1c (5/30) Explique el significado de la cota `t`.

La cota mide la longitud del intervalo de incertidumbre `b[i..j]`.

[5/30]

1d (10/30) Considere como operación básica la comparación con el elemento `x`. Estime y explique los costos en tiempo y en espacio del algoritmo propuesto.

Sea `T(n)` el tiempo, en el peor caso, para ejecutar el algoritmo con un dato dado. Al iniciar, la cota vale t_0 = `n-2`. En cada iteración, `k` o `j` cambian al punto medio (división entera) del intervalo de incertidumbre, de modo que el intervalo se cambia por uno que mide la mitad de lo que mide el original. Esto sucede hasta que el intervalo mide `1`. Es decir, se puede decir que:

```
T.1 = 0
T.n = 1 + T(n/2),   n≥1

Así:
   T(n) = θ(log n).
```

El costo en espacio se refiere a 3 variables para índices. Es decir:

```
   S.n = θ(1).
```

[10/30]

A manera de información, el algoritmo que se tiene en mente es como sigue:.

```
[   Ctx C: 0<n  ∧  b[0..n-1]:int ∧ b↑ ∧ x:int ∧ b[0]≤x<b[n-1]

    {Pre Q : true}

    i,j:= 0,n-1;
```

```
    {inv P :  0≤i≤j<n ∧ b[i]≤x<b[j] }
    {cota t:  j-i-1}

    do i+1≠j    →     m:= (i+j)÷2;
                      if    b[m]≤x      →     i:= m
                      []    x<b[m]      →     j:= m
                      fi
    od

    {R: 0≤i<n-1 ∧ b[i]≤x<b[i+1] }
]
```

2 (40 puntos)

Dentro de un grupo de `n` personas, una *celebridad* es una a quien todos conocen, pero que no conoce a nadie diferente de ella. Representando el grupo de personas con el conjunto `1..n`, supóngase que se tiene una representación matricial de las relaciones de conocimiento entre las personas, así:
`c[i,j]` ≡ "la persona `i` conoce a la persona `j`", con `1≤i,j≤n`.

Suponga que en el grupo hay una celebridad. Se quiere determinar quién en el grupo puede serlo.

2a (10/40) Defina formalmente notaciones para ser celebridad y para que exista una celebridad en una parte del grupo, y utilice sus definiciones para especificar formalmente el problema que se quiere resolver.

El concepto de *celebridad* se puede formalizar así:
`clb.x ≡ (∀j| j≠x : ¬c[x,j] ∧ c[j,x])` , para `x∈1..n`

[3/10]

La existencia de una celebridad en una parte del grupo se puede definir así:
`hayclb(i,j) ≡ (∃x| x∈i..j : clb.x)`

[3/10]

A manera de observación, que puede ser útil al razonar sobre la corrección del algoritmo, se puede afirmar que el grupo no puede tener dos celebridades diferentes. La razón es que si esto pasara, cada una debería conocer a la otra, y se violaría la condición de que las celebridades no conocen a nadie diferente de ellas.

Especificación del problema:

```
Ctx C:  c:[1..n,1..n]:boolean  ∧  hayclb(1,n)
Pre Q:  true
Pos R:  1≤x≤n ∧  clb.x
```

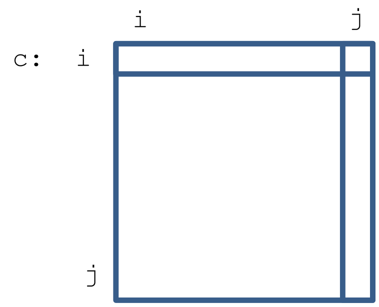
[4/10]

2b (10/40) Plantee un invariante `P` y una cota `t`, *con la técnica de apretar el cerco*.
AYUDA: *use un dibujo e indague cómo se puede avanzar rápidamente a terminación manteniendo el invariante.*

Con la técnica de apretar el cerco, se quiere un invariante de la forma:
`Inv:  1≤i≤j≤n ∧  hayclb(i,j)`

[8/10]

Variante: Gráficamente, se podría ilustra así (la parte blanca cumple `hayclb(i,j)`):



[8/10]

La cota se define con:
`Cota t: j-i`

[2/10]

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

```
[Ctx C
 {Pre Q}

   i,j:= 1,n;

 {Inv P : 1≤i≤j≤n ∧  hayclb(i,j)}
 {Cota t: j-i}

 do i≠j     →    if    c[i,j]     →    i:= i+1
                 []    ¬c[i,j]    →    j:= j-1
                 fi
 od

 {Pos R: 1≤i≤n ∧  clb.i}
]
```

[5/10]

Explicación de `{P ∧ i≠j}` ***if*** … ***fi*** `{P}`

Una celebridad `i` se reconoce porque la fila `i` es toda `false` (excepto, quizás, en `[i,i]`) y la columna `i` es toda `true`.

Si `c[i,j]` vale, se puede afirmar que `i` no es una celebridad, porque su fila tiene una valor `true`, en `i≠j`. En este caso, `i:=i+1` mantiene el invariante.

Si `¬c[i,j]`, `j` no puede ser celebridad, porque su columna no es toda `true`. En este caso, `j:= j-1` mantiene el invariante.

En los dos casos se mantiene el invariante y se rebaja la cota.

[5/10]

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

El cuerpo del ciclo tiene costo $\theta(1)$. En cada iteración se procesa una fila o una columna, hasta llegar a exactamente una fila y una columna. Es decir, hay exactamente $(n-1)+(n-1) = 2n-2 = \theta(n)$ iteraciones.

[5/10]

Para el costo del espacio solo se usan contadores e índices (un número constante).

[1/10]

En resumen:

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

[2/10]

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

[2/10]

3 **(40 puntos)**

Suponga que un sistema monetario tiene billetes de n denominaciones $d_1, d_2, \ldots, d_n$. Una cantidad de dinero X es *expresable* en el sistema monetario si X se puede construir con billetes del sistema. Por ejemplo, si las denominaciones son 3 y 7, se pueden construir 9 y 10, ya que $9=3+3+3$ y $10=3+7$; pero 2 y 11 no son expresables.

Se quiere decidir, mediante un algoritmo que utilice programación dinámica, si la cantidad X es expresable con las denominaciones $d_1, d_2, \ldots, d_n$. Se puede suponer que $0<d_1<d_2< \ldots <d_n$.

Como lenguaje, considere la notación:

$exp(x,i) \approx$ "x es expresable con billetes de denominaciones $d_1, d_2, \ldots, d_i$", $0\leq x\leq X,\ 0\leq i\leq n$.

3a **(30/40)** Usando la notación explicada, continúe con los pasos de la metodología de desarrollo de algoritmos de programación dinámica que se estudió en el curso. No es necesario que se escriba el algoritmo.

Recurrencia

Para $0\leq x\leq X,\ 0\leq i\leq n$:

$exp(x,0) \equiv x=0$

$exp(x,i) \equiv exp(x,i-1) \vee (\exists k|\ k>0 \wedge x-k*d_i\geq 0 : exp(x-k*d_i,i))$

[8/10]

La funcionalidad de `exp` es

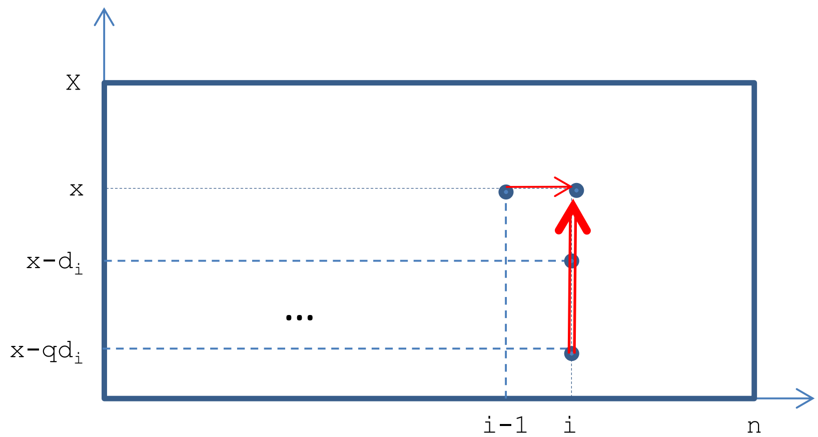
$$exp: 0..X \times 0..n \rightarrow \textbf{bool}$$

[1/10]

La pregunta es: ¿ $exp(X,n)$?

[1/10]

Diagrama de necesidades

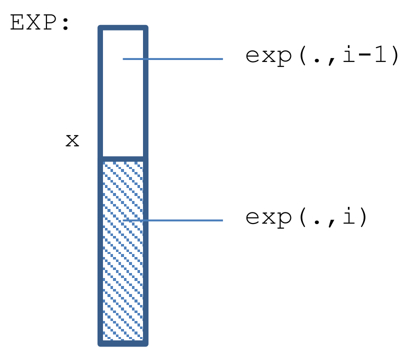


[10/10]

Estructura de datos + Invariante

Basta un vector columna que se llena de abajo arriba:

`Inv:  0≤x≤X ∧ 0≤i≤n ∧`



[10/10]

OJO: Variantes más costosas en espacio / tiempo son penalizadas

3b **(10/40)** Estime los costos temporal y espacial correspondientes (como θ(…)). Explique su respuesta.

Para calcular `T(X,n)` es necesario llenar una matriz de tamaño `Xn`. Cada elemento de la matriz se calcula en θ(X), ya que hay que calcular $(\exists k \mid k>0 \wedge x-k*d_i \geq 0: exp(x-k*d_i,i))$, y esto demanda máximo `X` cálculos, en el peor caso.

[5/10]

Es decir:

$T(X,n) = \theta(X^2 n)$

[2/10]

El espacio es el ocupado por el vector `EXP`:

[1/10]

$S(X,n) = \theta(X)$.

[2/10]