Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática, de Computadores y del Software (grupo A)

Examen Parcial, 10 de Febrero de 2014.

1. **(3 puntos)** Especifica, diseña y verifica o especifica y deriva, un algoritmo iterativo de coste lineal que dado un vector v de enteros, un valor entero positivo m tal que $0 \leq m \leq longitud(v)$ y un valor s igual a la suma de las m primeras posiciones del vector, devuelva el valor máximo que se puede obtener sumando m elementos consecutivos del vector.

Por ejemplo, sea $m = 3$, $s = 10$ y el vector

5	-4	9	7	-5	6	-1	10	0	3

la suma máxima de 3 elementos consecutivos es 15 = 6 -1 + 10.

Justificar el coste de la función implementada.

2. **(3 puntos)** Un vector de enteros mayores que 0 de longitud 2^n (donde n es un número natural) es *caucásico* si el valor absoluto de la diferencia entre el número de elementos pares de sus mitades es, a lo sumo, 2 y cada mitad también es *caucásica.*

Algunos ejemplos:

- $\{2, 4, 6, 8 \parallel 1, 3, 5, 7\}$ No es *caucásico*, porque su primera mitad tiene 4 elementos pares y la segunda 0.
- $\{2, 4, 6, 8 \parallel 2, 8, 5, 10\}$ Es *caucásico.*
- $\{2, 4, 8, 12, 3, 7, 9, 21 \parallel 10, 20, 30, 1, 3, 5, 7, 40\}$ No es *caucásico* ya que la primera mitad no lo es.

Diseña un algoritmo *recursivo* que determine si un vector de longitud 2^n es *caucásico.* El algoritmo debe de ser eficiente. Determina **justificadamente** la complejidad.

3. **(4 puntos)** Deseamos organizar un festival de rock al aire libre para lo cual vamos a contratar exactamente a N artistas de entre M disponibles ($N < M$). No todos los artistas aceptan tocar juntos en el festival. Los "vetos" entre artistas son conocidos de antemano. Para cada artista $i \in \{1..M\}$ conocemos el beneficio o pérdida generado por dicho artista $B[i]$, es decir si $B[i] > 0$, dicho artista genera beneficio mientras que si $B[i] < 0$ genera pérdida. Diseñar un algoritmo de vuelta atrás que resuelva el problema de planificar el festival que maximice la suma de los beneficios/pérdidas de los artistas participantes.

Febrero 2014

① v vector de enteros

$0 \leq m \leq long(v)$

$$s = \sum_{k=0}^{m-1} v[k]$$

Valor máximo que se puede obtener sumando m enteros consecutivos.

Especificación

$$\{ 0 \leq n \wedge n = long(v) \wedge 0 \leq m \leq n \wedge s = \sum_{k=0}^{m-1} v[k] \}$$

fun vMax (int v[], int n, int m, int s)

return int sMax;

$$\{ sMax = max\ w : 0 \leq w \leq n-m \cdot (\Sigma t : 0 \leq t < m : v[t+w]) \}$$

w	t	v[t+w]
0	0	v[0]
	1	v[1]
	2	v[2]
1	0	v[1]
	1	v[2]
	2	v[3]

w	t	v[w+t]
n-3	0	v[n-3]
	1	v[n-2]
	2	v[n-1]

Diseño

```
int vMax (int v[], int n, int m, int s){
    int sMax, i;
    sMax = s;
    i=0;
    while(i < n-m){
        if (sMax < s)
            sMax = s;
        s = s - v[i] + v[i+m];
        i++;
    }
    return sMax;
}
```

Verificación

Invariante $I \equiv (0 \le i \le n-m) \wedge \left(s = \sum_{k=i}^{i+m-1} v[k]\right) \wedge \left(sMax = \max_{0 \le l \le i} \sum_{k=l}^{l+m-1} v[k]\right)$

(I.1) $\{P\}\ sMax = s;\ i = 0\ \{I\}$

$pmd(sMax = s,\ i=0;\ I) \Leftrightarrow \left((I)_i^0\right)_{sMax}^{s}$

$\Leftrightarrow (0 \le 0 \le n-m) \wedge \left(s = \sum_{k=0}^{m-1} v[k]\right) \wedge \left(s = \max_{0 \le l \le 0} \sum_{k=l}^{l+m-1} v[k]\right)$

$\Leftrightarrow (0 \le n-m) \wedge \left(s = \sum_{k=0}^{m-1} v[k]\right)$

$P \equiv (0 \le n) \wedge (n = long\ v) \wedge \left(s = \sum_{k=0}^{m-1} v[k]\right) \wedge \underbrace{(0 \le m \le n)}_{\Downarrow\ (0 \le n-m)}$

$\Rightarrow pmd$ ✓

I.2 $\{I \wedge i < n-m\}\ \underbrace{A_1}_{\text{cuerpo del bucle (IF)}}\ \{I\}$

(a) $\{I \wedge (i < n-m) \wedge (sMax < s)\}\ sMax = s;\ s = s - v[i] + v[i+m];\ i{+}{+};\ \{I\}$

(b) $\{I \wedge (i < n-m) \wedge (sMax \ge s)\}\ s = s - v[i] + v[i+m];\ i{+}{+}\ \{I\}$

(a) $pmd(\_\_\_, I) \Leftrightarrow (I)_i^{i+1}\Big|_{s}^{s-v[i]+v[i+m]}\Big|_{sMax}^{s}$

$\Leftrightarrow (0 \le i+1 \le n-m) \wedge \left(s - v[i] + v[i+m] = \sum_{k=i+1}^{i+m} v[k]\right) \wedge$

$\wedge \left(s = \max_{0 \le l \le i+1} \sum_{k=l}^{l+m-1} v[k]\right)$

$\Leftrightarrow (0 \le i+1 \le n-m) \wedge \underbrace{\left(s = \sum_{k=i}^{i+m-1} v[k]\right) \wedge \left(s = \max_{0 \le l \le i+1} \sum_{k=l}^{l+m-1} v[k]\right)}_{s = \sum_{k=i}^{i+m-1} v[k]}$

Veamos que la precondición implica lo pmd.

$\underline{(0 \le i \le n-m)} \wedge \boxed{\left(s = \sum_{k=i}^{i+m-1} v[k]\right)} \wedge \left(sMax = \max_{0 \le l \le i} \sum_{k=l}^{l+m-1} v[k]\right)$

$\underline{\wedge (i < n-m)} \wedge (sMax < s)$

(I3) $I \wedge (i \geq n-m) \Rightarrow \left(sMax = \max_{0 \leq l \leq n-m} \sum_{k=l}^{l+m-1} V[k]\right)$

$(P \geq n-m) \wedge (0 \leq i \leq n-m) \wedge \left(s = \sum_{k=1}^{l+m-1} v[k]\right) \wedge \left(sMax = \max \sum_{k=l}^{l+m-1} v[k]\right)$

$\Rightarrow (i = n-m) \wedge \left(sMax = \max_{0 \leq l \leq i} \Sigma \ldots\right)$

$\Rightarrow sMax = \max_{0 \leq l \leq n-m} \sum_{k=l}^{l+m-1} v[k]$

Función de cota

$C(v, n, m, s, i, sMax) = n - m - i$

(C1) $I \wedge (i < n-m) \Rightarrow n-m-i > 0 \Rightarrow n-m-i \geq 0$ ✓✓

(C2) $\{(I \wedge B) \wedge T = c\}$ A1 $\{T > c\}$

Podemos hacerlo de una vez pq en el if no tocamos i.

$pmd(\Delta 1, T > c) \Leftrightarrow (T > n-m-1) \left|_{s}^{s} \right|_{sMax}^{s-v[i]+v[mm]} \Big|_{i}^{l+1}$

$\Leftrightarrow T > n-m-(i+1) \Leftrightarrow T > n-m-i-1$

COMPLEJIDAD

$T(n) = C(n-m-1) + \tilde{C}$

$= Cn + cte = O(n)$

(2) 2468 || 1357 ⇒ no es cancásico

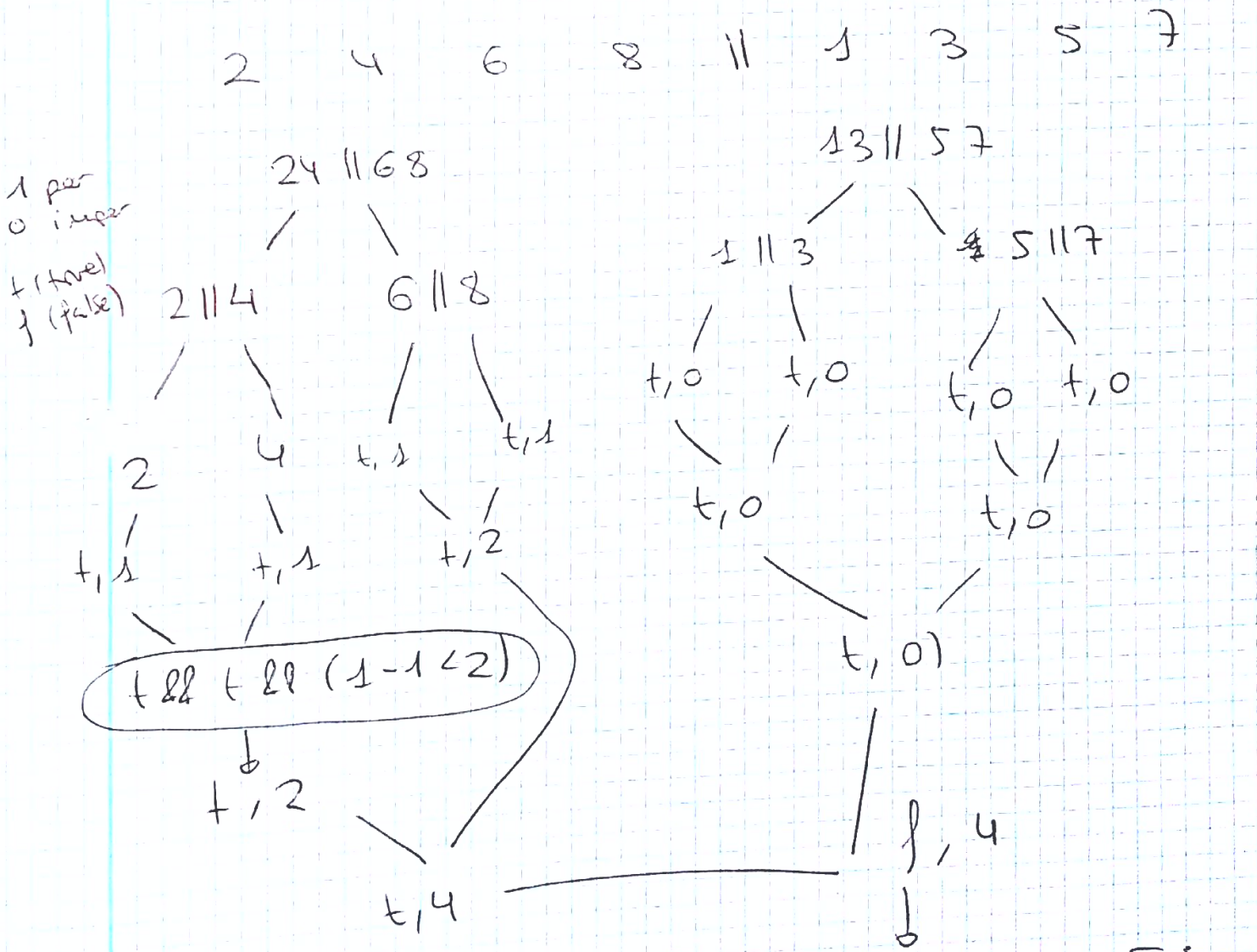
4 pares 0 pares

2|4|6 8 || 2 8|5 10 ⇒ sí es cancásico

4 3

Planteamiento

- Caso base : un elemento ⇒ es cancásico.
- Caso recursivo



```
void esCancesico (int v[], int P, int U, bool &cancs,
 int & nPares)
bool cancaI, cancaD;
int nParesI, nParesD, M;
if (P==U) {
    canca = true;
    if (v[P]%2 ==0) nPares = 1;
    else nPares = 0;
} else {
    M = (P+U)/2;
    esCancesico (v, P, M, cancaI, nParesI)
    esCancesico (v, M+1, U, cancaD, nParesD)
    canca = cancaI && cancaD && (abs(nParesI-nParesD)<=2);
}
```

Complejidad

Recursión por división

$T(n) = C$ (caso base)

$T(n) = a\,T(n/b) + cn^k$

En nuestro caso $a=2$, $b=2$, $k=0$;

$$O(n^{\log_b a}) = a > b^k$$

$$\boxed{O(n)}$$

(Vuelta Atrás)

③ M (total artistas) > N ← nº escenarios

vetos [M] [M]

beneficios

(cada artista tiene un beneficio)

Planteamiento

La solución viene dada por:
un vector de M posiciones una por cada artista de booleanos que indican si los artistas participan en el festival o no.

- Restricciones explícitas:

  Lista de M bodeanos

- Restricciones implícitas:
  - Maximizar el beneficio
  - Respetar los vetos
  - Asisten exactamente n artistas así que hay (N trues y M-N falses)

PARÁMETROS

- bool sol [M]: vector solución actual

sol
- int N: Num artistas en el festival
- int M: Num artistas totales
- bool veto [M] [M]: matriz de vetos

- solOpt [M]: vector sol optima

sol
- int benef [M]: vector beneficios
- int bAct: beneficio actual
- int bMax: " máximo
- int k: ~~Etapa~~

sol
- int nAct: Num artistas actuales

CÓDIGO:

```
void festival (bool sol[], bool solOpt[], int &bAct,
     int &bMax, int k, int &nAct){

  sol[k] = true; // Metemos al artista en el festival
  bAct += beneficios[k];
  nAct++;
  if (esValida(sol, k, nAct)){ // Que no haya vetos
                               // Es sol
     if (k == M-1 && (N==nAct)) // n artistas vienen
        if (bAct > bMax) // beneficios mejores?
           bMax = bAct;
           copiaSol (solOpt, sol);
     else{ // Si no es sol
        if (k<M-1){ // Elijo el sig. sólo si puedo elegirlo
        festival (sol, sol, bAct, bMax, k+1, nAct);
  }
  sol[k] = false; // Si no, quito al artista
  bAct -= beneficios[k];
  nAct--;
  if (k==M-1) && (nAct==N){ // Lo devuelvo a comprobar
     if (bAct > bMax)
        bMax = bAct
        copiaSol (solOpt, sol);
  else{ if(k<M-1)
        festival (sol, solOpt, bAct, bMax, k+1, nAct);
  }
```

```
bool esValido (int sol[], int k, int nAct)
    bool v = false;
    for (int i=0; i<k; i++)
        v = v || (veto[i][k]
                  && sol[i])

    return (!v && nAct <= N)
```

Una vez que los encontrado veto → true