Universidad de Granada

Divide y Vencerás

Algorítmica

Lukas Häring García 2º D

Tabla de contenidos

Ejercicio 1

Sea $a[1..n]$, $n \geq 1$, un vector de enteros diferentes y ordenados crecientemente, tal que algunos de los valores pueden ser negativos. Diseñar un algoritmo que devuelva un índice natural k, $1 \leq k \leq n$, talque $a[k] = k$, siempre que tal índice exista. El coste del algoritmo debeser $O(\log n)$ en el caso peor. Se pide diseñar el algoritmo dando todo lujo de detalles. Justificar que funciona correctamente y el coste. Repetir el problema pero para un vector de n enteros posiblemente repetidos y ordenado decrecientemente.

```
1  int abs(int k){ return (k < 0 ? -k : k); }
2  int sign(int k){ return(k == 0 ? 0 : (k / abs(k))); }
3  int k_posicion_k(int* v, int a, int b){
4    if(b < a) return -1;
5
6    int m = (a + b) / 2, l = v[m];
7    // Miramos en uno de los dos lados.
8    switch(sign(m - l)){
9      case 1:  m = k_posicion_k(v, m + 1, b); break;
10     case -1: m = k_posicion_k(v, a, m - 1); break;
11   }
12   return m;
13 }
```

Ejercicio 2

Dados n enteros cualesquiera $a_1, a_2, ..., a_n$, necesitamos encontrar el valor de la expresión:

$$\max_{0 \leq i \leq j < n} \sum_{k=i}^{j} a_k$$

que calcula el máximo de las sumas parciales de elementos consecutivos. Deseamos implementar un algoritmo Divide y Vencerás de complejidad $n \log n$ que resuelva el problema. ¿Existe algún otro algoritmo que lo resuelva en menor tiempo?

Ejercicio 4

Se dispone de un conjunto de n tornillos de diferente tamaño y sus correspondientes n tuercas, de forma que cada tornillo encaja perfectamente con una y sólo una tuerca. Dado un tornillo y una tuerca, uno es capaz de determinar si el tornillo es menor que la tuerca, mayor, o encaja exactamente. Sin embargo, no hay forma de comparar dos tornillos o dos tuercas entre ellos para decidir un orden. Se desea ordenar los dos conjuntos de forma que los elementos que ocupan la misma posición en los dos conjuntos emparejen entre sí.

```
1  // Realiza el swap de dos valores en un vector.
2  void s(int* v,int i,int j){int h=*(v+i);*(v+i)=*(v+j);*(v+j)=h;}
3  void shift(int* v, int i, int n, int k){
4    for(int j=n-1;j>i;--j){*(v+j)=*(v+j-1);}*(v+i)=k;}
5  void Tornillos_Tuercas(int* tornillos, int* tuercas, int n){
6    if(n <= 1){ return; }
7    int i = 0;
8    for(i = 0; *(tuercas + i) != *(tornillos + n - 1); ++i);
9    s(tuercas, i, n - 1);//SWAP
10   int m = -1, w = 0;
11   while(w < n){ /* MOVEMOS LAS TUERCAS */
12     if(*(tuercas + w) < *(tornillos + n - 1)){
13       ++m;
14       s(tuercas, m, w);//SWAP
15     }++w;
16   }
17   shift(tuercas, ++m, n, *(tornillos + n - 1));
18   m = -1; w = 0;
19   while(w < n){ /* MOVEMOS LOS TORNILLOS */
20     if(*(tornillos + w) < *(tuercas + n - 1)){
21       s(tornillos, m++, w);//SWAP
22     }++w;
23   }
24   shift(tornillos, ++m, n, *(tuercas + n - 1));
25   Tornillos_Tuercas(tornillos, tuercas, m);
26   Tornillos_Tuercas(tornillos + m, tuercas, n - m);
27 }
```

Ejercicio 5

Dado un vector de números enteros positivos y negativos, encontrar el subvector de casillas contiguas que contenga la secuencia en orden creciente no monótono de longitud máxima. Por ejemplo, dado el vector {2, -1, 3, 3, 8, 7, 6, -2, 5}, la secuencia creciente no monótona de longitud máxima es $\{-1, 3, 3, 8\}$. En caso de que haya dos secuencias distintas con el mismo tamaño, bastaría con obtener una de ellas

```
1  int* mayor_serie(int* v, int a, int b){
2    int diff = b - a + 1;
3    int* res = NULL;
4    switch(diff){
5      case 1: res = new int[1]{ v[a] }; break;
6      case 2:
7        if(v[a] >= v[b]){ res = new int[2]{ v[a], v[b] }; }
8      break;
9      default:
10       int m = (a + b) / 2;
11       int* s1 = mayor_serie(v, a, m - 1);
12       int* s2 = mayor_serie(v, m, b);
13       if(s1 != NULL && s2 != NULL){
14         if(s1[m - 1 - a] <= s2[0]){
15           res = new int[diff];
16           for(int i = 0; i < (m - 1 - a); ++i){
17             res[i] = s1[i];
18           }
19           for(int i = 0; i < (b - m); ++i){
20             res[m + i] = s2[i];
21           }
22         }
23         delete[] s1, s2;
24       }else if(s1 != NULL){ delete[] s2; res = s1;
25       }else(s2 != NULL){ delete[] s1; res = s2; }
26     break;
27   }
28   return res;
29 }
```

Especificaciones

1. Windows 10.0.14393
2. Procesador Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz, 3504 Mhz
3. 6 procesadores principales.
4. 12 procesadores lógicos.
5. Memoria física instalada (RAM) 8,00 GB x 2
6. Compilador MinGW.