Universidad de Chile

Departamento de Ciencias de la Computación

Control 1 CC4102

Semestre: Augusto 2010

Solution

Problema 1 (1.5 puntos)
Sea un arreglo $A[1,\ldots,n]$ ordenado de n elementos, un elemento x, y la tarea de encontrar $p \in \{1,\ldots,n+1\}$ tal que $A[p-1] < x \leq A[p]$ si $p \leq n$ y $A[n] < x$ sino.

1. Cual es la definición del algoritmo de busqueda por interpolación? Cual es su complejidad en promedio si los elementos del arreglo estan selectionados uniformamente en un universo grande?

——————begin solution————————

```
interpolationSearch(A, k, n)
  i ← 1
  j ← n
  while i + 1 < j do
    guess ← i + ⌊(j − i) ∗ (k − A[i])/(A[j] − A[i])⌋
    if  k ≤ A[guess] then
      j ← guess
    else
      i ← guess
    end if
  end while
  return j
```

Si los elementos del arreglo estan selectionados uniformamente en un universo grande, su compljidad en promedio es $\mathcal{O}(\lg\lg n)$.

———————end solution————————

2. Cual es la definición del algoritmo de “doubling search”? Cual es su complejidad en el peor caso?

——————begin solution————————

```
doublingSearch(A, k, n)
  i ← 1
  j ← n
  gap ← 1
  while i + gap ≤ j and k > A[i + gap] do
    i ← i + gap
    gap ← 2 ∗ gap
  end while
  return binarySearch(A, k, i, mín{i + gap, j})
```

Si $A[p-1] < k \leq A[p]$, el algoritmo encontro p despues de $2\lg\lg p \in \mathcal{O}(\lg\lg p)$ comparaciones.

——————————end solution——————————

3. De un algoritmo de busqueda por *extrapolación* inspirado de ambos algoritmos de busqueda por interpolacion y de "doubling search", que adivina una posición g basada en las dos ultimas posiciones comparadas a x (no se necesita la complejidad).

——————————begin solution——————————

Note: it is normal that the extrapolation algorithm is exactly the same as the interpolation algorithm.

```
extrapolationSearch(A, k, n)
  i ← 1
  j ← n
  while i + 1 < j do
    guess ← i + ⌊(j − i) ∗ (k − A[i])/(A[j] − A[i])⌋
    if  k ≤ A[guess] then
      j ← guess
    else
      i ← guess
    end if
  end while
  return j
```

Si los elementos del arreglo estan selectionados uniformamente en un universo grande, su compljidad en promedio es $\mathcal{O}(\lg \lg n)$.

——————————end solution——————————

Problema 2 (1.5 puntos)

—————————begin solution———————————

Note that to see the edges of the tree in the solution, one must compile to dvi and then obtain the ps file through dvips.

——————————end solution————————————

Un de los numerosos lagos del sur de Chile es contaminado con una bacteria muy poderosa. No es claro cual lago, pero la contaminación biológica se va a propagar a los otros lagos en algunos meses. El procedimiento para limpiar un lago es muy caro y muy destructivo para el lago, entonces es preferible de aplicarlo solamente al lago contaminado.

La asociación local para la protección de la naturaleza organizó una recolección de 16 extractos de agua de diferente lagos. Es posible probar si una pequeña cantidad de agua es contaminada con una incubadora en dos semanas. Desafortunadamente, la asociación puede utilizar solamente cuatro incubadoras de la universidad. Ellos saben que es posible examinar cuatros extractos de agua en dos semanas, o ocho en un mes, que necesitaria dos meses para probar 16 extractos. El miedo es que la contaminación biológica se propagage a todos los lagos antes que el análisis de los extractos sea terminada.

Ayude la asociación y explique a ellos el procedimiento para analizar los 16 extractos en solamente dos semanas, suponiendo que solamente un extracto contiene la bacteria, y utilizando el hecho que la incubadora puede detectar muy pequeñas cantidades de bacterias, como por ejemplo en una mezcla de extractos. Su explicación puede ser muy corta, pero necesita clarificar los puntos siguientes:

1. cuales experimentos son ejecutados;

2. como analizar los resultados de los experimentos;

3. una prueba de que la determinación sera terminada en dos semanas (o menos);

4. presente un ejemplo de solución con un diagrama.

—————————begin solution———————————

The solution is based on putting in the incubator a combination of several samples at once. We give it for general N, just take $N = 16$ for the solution to this particular problem.

1. Prepare some mixes, each containing water from a number of samples as follows: Number the samples from 0 to $N-1$; write these numbers in binary, and look at the bits of the binary encodings from left to right; Potion p contains some water from sample b if and only if the pth bit in the binary encoding for sample b is 1. This set of mixes is the set of tests to be run.

2. Note that with $\lceil \lg N \rceil$ bits, you can encode $2^{\lceil lgN \rceil} \geq N$ different numbers in binary. The number of mixes needed is the number of bits needed, so the number of tests is $n = \lceil \lg N \rceil$.

3. To determine which sample contains the germ from the test results, define $(x_i)_{i \leq n}$ such that $x_i = 1$ if the mix i has a positive test result, and $x_i = 0$ otherwise. The values of $(x_i)_{i \leq n}$ form a number x on n bits, and this number (between 0 and $N-1$) is the number of the contaminated sample.

4. For example, in the case where one sample contains the germ among $N = 16$ samples, the analysis of only $n = 4$ mixes composed as described in the array below permits us to find this person, as shown in the tree below. Note that the bits are arranged from the heavier 1 to the lighter n (from the left to the right): this is not important as long as the same convention is applied for encoding and decoding.

Mix 1 is a mix of the water samples	8 to 15;
mix 2 is a mix of the water samples	4 to 7 and 12 to 15;
mix 3 is a mix of the water samples	$\{2, 3, 6, 7, 10, 11, 14, 15\}$;
mix 4 is a mix of the water samples	corresponding to odd numbers.

x_1

x_2 x_2

x_3 x_3 x_3 x_3

x_4 x_4 x_4 x_4 x_4 x_4 x_4 x_4

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

——————–end solution————————–

Problema 3 (1.5 marks)
Un `min-max priority queue` es un tipo de datos abstracto que provee las operaciones `deleteMin` y `deleteMax`, muy similar al tipo de datos abstracto `min priority queue`. En este problema, n denota la cantidad maxima de elementos en la fila.

1. Describa un `min-max heap`, una estructura de datos tomando espacio $n+O(1)$ que implementa las operaciones `deleteMin` y `deleteMax` en tiempo $\mathcal{O}(\lg n)$ para cada operación, utilizando un solo árbol binario, y técnicas muy similares a las utilizadas para un `heap` usual.

   ————————begin solution——————————

   A simple solution, since no space requirement is specified, is to simply use any efficient ordered dictionary structure, such as an AVL tree, with two additional elements $-\infty$ and $+\infty$: the smallest element is the successor of $-\infty$ and the largest element is the predecessor of $+\infty$. The search for both, and the addition and removal of elements is supported in logarithmic time.

   Should one require better space than $O(n)$, one can consider a binary tree with the same shape property as a normal heap, but with the following ordering property, where "value" indicates the value stored at a node, and $p(x)$ indicates the parent of node x: for any node x at depth $2, 4, 6, \dots$

   $$\text{value(p(p(x)))} < \text{value(x)} < \text{value(p(x))}$$

   and for any node x at depth $3, 5, \dots$

   $$\text{value(p(x))} < \text{value(x)} < \text{value(p(p(x)))}$$

   and for nodes x at depth 1 (i.e. the children of the root)

   $$\text{value(p(x))} < \text{value(x)}$$

   —————————end solution———————————

2. Describa un algoritmo para agregar un nuevo elemento en un `min-max heap` en tiempo $\mathcal{O}(\lg n)$. Justifique su análisis de complejidad.

   ————————begin solution——————————

   Add the element to the first leaf available, as in a normal heap.

   Then, going up the path to the root, check if each node respects the min max heap property relative to its depth:

   $$\text{if } x \text{ has value } a,\ p(x) \text{ has value } b \text{ and } p(p(x)) \text{ has value } c,$$
   $$\text{then } a \in [\text{mín}(b, c), \text{máx}(b, c)].$$

   When it does not, reorder the three nodes of the path so that they respect the condition and check the property upward.

   As the tree is of height $\lg n$ and as each reordering requires a finite number of operations, the whole insertion requires at most $O(\lg n)$ operations.

——————end solution——————

3. Describa como buscar y borrar el elemento mínimo en un `min-max heap` en un tiempo mínimo. Explique la complejidad de su algoritmo.

——————begin solution——————

The minimum element is at the root. To remove it, exchange it with the last element, as in a normal heap. Then, going down the tree, check if each node respects the min max heap property relative to its depth, its two children and its four grand-children. When it does not, reorder the three nodes of the path so that they respect the condition and check the property downward in the only modified subtree.

As the tree is of height $\lg n$ and as each reoredering requires a finite number of operations, the whole deletion requires at most $O(\lg n)$ operations.

——————end solution——————

Problema 4 (1.5 puntos)

1. Defina los términos siguientes (de las formulas): $\mathcal{O}()$, $\Omega()$, $\Theta()$, $o()$ y $\omega()$

——————begin solution————————

- $f(n) \in O(g(n))$ si

$$\exists c, n_0 > 0 tq \forall n > n_0, f(n) \leq cg(n)$$

- $f(n) \in \Omega(g(n))$ si

$$\exists c, n_0 > 0 tq \forall n > n_0, f(n) \geq cg(n)$$

  (o si $g(n) \in O(f(n))$)

- $f(n) \in \Theta(g(n))$ si

$$\exists c_1, c_2, n_0 > 0 tq \forall n > n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$f(n) \in O(g(n)) y f(n) \in \Omega(g(n))$$

- $f(n) \in o(g(n))$ si

$$\forall c \exists n > 0 tq \forall n > n_0, f(n) < cg(n)$$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) \notin \Omega(g(n))$$

- $f(n) \in \omega(g(n))$ si

$$\forall c \exists n > 0 tq \forall n > n_0, f(n) > cg(n)$$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0$$

$$f(n) \notin O(g(n))$$

$$g(n) \in o(f(n))$$

——————end solution————————

2. Rellene con las complejidades pedidas (exactas cuando se saben, no asimptóticas, y en el modelo de comparaciones):

——————begin solution————————

Problema	comparaciones	accessos a memoria externa
Buscar el minimo en arreglo desordenado	$n-1$	$\lceil n/B \rceil$
Buscar en arreglo desordenado	n	$\lceil n/B \rceil$
Buscar en arreglo ordenado	$1+\lg n$	$\lg n - \lg B + 1$
Buscar la mediana en arreglo ordenado	0	1
Ordenar un arreglo desordenado	$\mathcal{O}(n \lg n)$	$n \log_m n = N/B \frac{\lg(N/B)}{\log M/B}$

———————end solution————————

3. De el código y la complejidad exacta del algoritmo que calcula el min y el max de un arreglo desordenado $A[1..n]$?

——————begin solution————————

- Hacer $n/2$ comparaciones de pares disjuntas.
- Calcular el max de los $n/2$ gañadores en $n/2-1$ comparaciones: eso es el maximo.
- Calcular el min de los $n/2$ gañadores en $n/2-1$ comparaciones: eso es el minimo.

La complejidad total es de $\lceil 3n/2 \rceil - 2$ comparaciones

———————end solution————————