Facultad de Ciencias, UNAM
Análisis de Algoritmos
Tarea 1

Rubí Rojas Tania Michelle

09 de octubre de 2020

1. ¿Cuántas comparaciones son necesarias y suficientes para ordenar cualquier lista de cinco elementos? Justifique su respuesta.

2. Dados dos arreglos ordenados A y B de longitud n y m, respectivamente. Diseña un algoritmo de tiempo $O(n+m)$ que obtenga un arreglo C que contenga los elementos entre A y B, C no debe tener elementos repetidos.

3. Consider the following sorting algorithm:

```
STUPIDSORT(A[0..n-1]):
  if n = 2 and A[0] > A[1]
      swap A[0] ↔ A[1]
  else if n > 2
      m = ⌈2n/3⌉
      STUPIDSORT(A[0..m-1])
      STUPIDSORT(A[n-m..n-1])
      STUPIDSORT(A[0..m-1])
```

   a) Prove that STUPIDSORT actually sorts its input.

   b) Would the algorithm still sort correctly if we replaced $m = \lceil \frac{2n}{3} \rceil$. Justify your answer.

   c) Show that the number of swaps executed by STUPIDSORT is at most $\binom{n}{2}$.

4. Supongamos que tenemos que ordenar una lista L de n enteros cuyos valores están entre 1 y m. Pruebe que si m es $O(n)$ entonces los elementos de L pueden ser ordenados en tiempo lineal. ¿Qué pasa si m es de $O(n^2)$? ¿Se puede realizar en tiempo lineal? ¿Por qué?

5. Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a...b]$ in $O(1)$ time. Your algorithm should use $\Theta(n+k)$ preprocessing time.

   SOLUCIÓN:

6. Sea A un arreglo de n elementos, tal que cada elemento se encuentra a lo más a k posiciones de su posición ordenada. Diseñe un algoritmo que ordene A en $O(n \log k)$.

7. An abs-sorted array is an array of numbers in which $|A[i] \leq A[j|$ whenever $i < j$. For example, the array $A = [-49, 75, 103, -147, 164, -197, -238, 314, 348, -422]$, though not sorted in the standard

sence, is abs-sorted. Design and algorithm that takes an abs-sorted array A and a number k, and returns a pair of indices of elements in A that sum up to k. For example, if $k = 167$ your algorithm should output $(3, 7)$. Output $(-1, -1)$ if the is no such pair.

8. **The Hogwarts Sorting Hat**

   Every year, upon their arrival at Hogwarts School of Witchcraft and Wizardry, new students are sorted into one of four houses (Gryffindor, Hufflepuff, Ravenclaw, or Slytherin) by the Hogwarts Sorting Hat. The student puts the Hat on their head, and the Hat tells the student which house they will join. This year, a failed experiment by Fred and George Weasley filled almost all of Hogwarts with sticky brown goo, mere moments before the annual Sorting. As a result, the Sorting had to take place in the basement hallways, where there was so little room to move that the students had to stand in a long line. After everyone learned what house they were in, the students tried to group together by house, but there was too little room in the hallway for more than one student to move at a time. Fortunately, the Sorting Hat took Algorithms many years ago, so it knew how to group the students as quickly as possible. What method did the Sorting Hat use? More formally, you are given an array of n items, where each item has one of four possible values, possibly with a pointer to some additional data. Design and analyze an algorithm that rearranges the items into four clusters in $O(n)$ time using only $O(1)$ extra space.

9. Pruebe que el segundo elemento más chico de una lista de n elementos distintos puede encontrarse con $n + \lceil \log n \rceil - 2$ comparaciones.

Demostración. Sea T el árbol binario, en particular un *min-Heap*, que contiene a los n elementos de nuestra lista en sus hojas. Por lo discutido en clase, sabemos que para encontrar al elemento más pequeño necesitamos realizar $n - 1$ comparaciones, ya que al ir comparando los elementos desde las hojas, vamos formando nuestros nodos internos (éstos serán los elementos más pequeños de las comparaciones que se vayan haciendo), los cuales siempre son $n - 1$.

Ahora bien, para encontrar al segundo elemento más pequeño debemos tener en cuenta una observación importante: como siempre vamos *subiendo* a los elementos más pequeños, eso quiere decir que el segundo elemento más pequeño ya fue comparado con la raíz del árbol T, así que sólo queda ubicar a todos los elementos que *perdieron* contra la raíz de T y compararlos entre sí. Al ubicar estos elementos, obtendremos que hay uno de ellos (a lo más) en cada nivel del árbol, y como la altura del árbol es $\log_2 n$ entonces hacer esta última pasada al árbol nos tomará $\lceil \log_2 n \rceil - 1$ comparaciones. Por lo tanto, encontrar al segundo elemento más pequeño de una lista de n elementos distintos nos toma

$$(n-1) + (\lceil \log_2 n \rceil - 1) = n + \lceil \log_2 \rceil - 2$$

comparaciones.

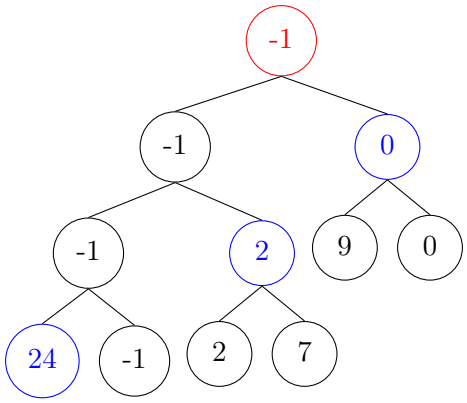

Figura 1: Ejemplo de la explicación con la lista $[24, -1, 2, 7, 9, 0]$. Los elementos en azul son aquellos que *perdieron* contra el elemento más pequeño.

□