Algoritmos
TAREA 3

Compilación

4 de diciembre de 2013

Propón un algoritmo de ordenamiento que haga uso de una cola de prioridades. Explica en detalle su complejidad y correctez.

El algoritmo que a continuación se presenta supone que se tiene una cola de prioridades Q implementada eficientemente como un heap y que dicha cola siempre tiene al inicio el número mas pequeño. También supongo por sencillez que los elementos del arreglo L son distintos, aunque el algoritmo funciona si fueran iguales, pero facilita el análisis.

Algorithm 1 Ordenamiento con cola de prioridades

```
1: Q ← ∅
2: L_o ← 0
3: for j ← 0 upto length(L) do
4:     Q.insertWithPriority(L[j],L[j])
5: end for
6: for i ← 0 upto length(L) do
7:     L_o[i] ← Q.removeMin()
8: end for
9: return L_o
```

1. Correctez.

   Suponemos que la cola de prioridad funciona como está especificada.
   Observaciones:

   a) El for de las lineas 3-5 y 6-8 se ejecutan $|L|$ veces

   b) En la fila 4 se ingresa cada número L en la cola de prioridad con su valor como prioridad.

   (Dice el ayudante que esto aún no es correctez, esta respuesta tiene 0.95)

2. Invariante: En cada paso i del ciclo , los elementos $L_o[0]$ a $L_o[i]$ se encuentran ordenados.

   Demostración:

   Al inicio de la ejecución esto es trivialmente cierto ya que el arreglo está vacío.

   En cada iteración del ciclo de la línea 3 se asigna al arreglo de resultados el valor con menor prioridad dentro de la cola, en este caso, el elemento con valor menor (como aparece en la observación b). Al eliminarse el elemento de la cola Q (linea 7) se garantiza que ningún elemento puede repetirse.

   Al final de la ejecución, en el arreglo L_o quedan los elementos de Q sin repetir (solo una vez cada uno) en orden ascendente, ya que durante todo el ciclo siempre se asignó el valor mínimo de la cola de prioridad de los elementos restantes.

3. Complejidad

Suponemos una implementación eficiente de la cola de prioridad Q y n como el número de elementos a ordenar.

Ciclo 2-5: Se ejecuta n veces y la complejidad de insertWithPriority es O(log n)
Ciclo 6-8: Se ejecuta n veces y la complejidad de removeMin es O(log n)

Por lo tanto, la complejidad es O(nlog n).