CC4102 - Diseño y Análisis de Algoritmos
Auxiliar 5

Prof. Gonzalo Navarro; Aux. Mauricio Quezada

30 de Noviembre, 2011

1 Linear time sorting

1. Describa un algoritmo que, dados n enteros en $[0..k]$, preprocesa su entrada y responde cuántos de los n enteros caen en el rango $[a..b]$ en tiempo constante. Su algoritmo debería tomar $\Theta(n+k)$ en el preprocesamiento.

2. Muestre cómo ordenar un arreglo de enteros, donde los enteros no necesariamente tienen la misma cantidad de dígitos, pero la cantidad total de dígitos es n; en tiempo $O(n)$.

3. Muestre cómo ordenar n enteros en el rango $[0..n^3-1]$ en tiempo $O(n)$.

1.1 Solución P1.1

Genere el arreglo C como en Counting sort. El numero de enteros en el rango $[a..b]$ es $C[b]-C[a-1]$ donde interpretamos a $C[-1]$ como 0.

CountingSort(A, B, k):

- Sea $C[0..k]$ un arreglo vacio
- Para $i = 0..k, C[i] = 0$
- Para $j = 0..|A|$, $C[A[j]] = C[A[j]] + 1$
- Para $i = 1..k$, $C[i] = C[i] + C[i-1]$
- Para $j = |A|..1$
  - $B[C[A[j]]] = A[j]$
  - $C[A[j]] = C[A[j]] - 1$

1.2 Solucion P1.2

- Usando radix sort usual, considere d como la cantidad de digitos del entero mas grande. Suponga que hay m enteros ($m \leq n$). En el peor caso, tendremos un entero con $n/2$ digitos y $n/2$ enteros con un digito cada uno. Por lo tanto, $d = n/2$ y $m = n/2 + 1$, por lo que el tiempo total sera $\Theta(dm) = \Theta(n^2)$.

- Asuma sin perdida de generalidad que todos los enteros son positivos y que no tienen ceros al comienzo (sino los consideramos por separado). Bajo esta suposicion, podemos observar que los enteros con más digitos siempre serán más grandes que los enteros con menos digitos. Luego, primero podemos ordenar los enteros de acuerdo a su número de dígitos (usando counting sort, por ejemplo), y luego usar radix sort para ordenar cada grupo de enteros del mismo largo.

Note que cada entero tiene entre 1 y n dígitos; sea m_i el número de enteros con i dígitos ($i = 1, 2, \cdots, n$), por lo que $\sum_{i=1}^{n} i \cdot m_i = n$.

Toma $O(n)$ calcular cuántos dígitos tienen todos los enteros y, una vez que las cantidades de dígitos han sido calculadas, toma $O(m + n) = O(n)$ en agrupar los enteros por su cantidad de dígitos. Para ordenar el grupo de m_i enteros con radix sort, necesitamos tiempo $\Theta(i \cdot m_i)$. Por lo tanto, el tiempo para ordenar todos los grupos es

$$\sum_{i_1}^{n} \Theta(i \cdot m_i) = \Theta(\sum_{i=1}^{n} i \cdot m_i) = \Theta(n)$$

1.3 Solucion P1.3

Considera a cada numero como un numero de 2 dígitos en radix n. Cada digito varia entre 0 y $n - 1$. Ordene cada numero de 2 digitos con radix sort.

Como hay dos llamadas a Counting Sort, cada una tomando $\Theta(n + n) = \Theta(n)$, por lo que el tiempo total se cumple.

2 Tries, Patricia, Suffix trees

1. Construya un Patricia Tree con la siguiente secuencia de inserciones: `ababb`, `ababa`, `ba`, `aaabba`, `ab`.

2. Construya el Suffix Tree de `mississippi` y muestre cómo buscar los strings `ssi` y `sissy` en él.

2.1 Solución P2.1

Ver http://allisons.org/ll/AlgDS/Tree/PATRICIA/

2.2 Solucion P2.2

Ver http://www.cs.ucf.edu/s̃hzhang/Combio/lec3.pdf