Clasificación en tiempo lineal

Patrichs Inocente Valle[1], Dashiel Sanchez Ramos[1]
[1]Escuela profesional de Ciencias de la Computación
Universidad Nacional de Ingenieria

20 de noviembre de 2018

SECCIÓN 1
Introducción

La motivación principal de este articulo, es encontrar la forma de ordenar n enteros de tamaño w-bits en tiempo Lineal para todo w, si esto se logra podremos implementar una cola de prioridad donde la inserción y eliminacion de un elemento sea en tiempo constante para cualquier tamaño w, basandonos en lo que Throup publicó(1).

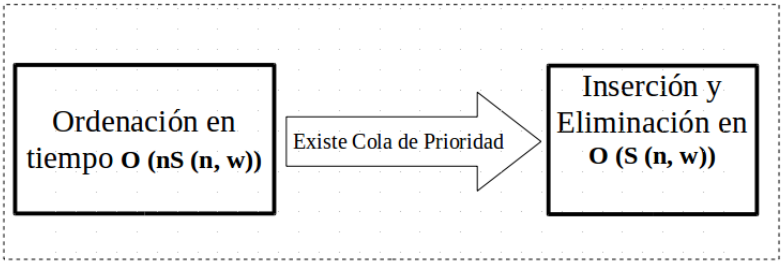

Figura 1. La reducción de la representación de colas de prioridades a la ordenación segun Throup.

Por lo que se busca encontrar ordenamientos en tiempo lineal que cubran todo el rango de w. Algunos avances:

Ordenamiento	Tiempo	Valor de w
Counting Sort	O(n)	lgn
Radix Sort	O(n)	O(lgn)
Van Emde Boas	O(nlg(lgn))	$lg^{O(1)}n$
Signature Sort	O(n)	$\Omega(lg^{2+\varepsilon}n)$

Figura 2. Avances de ordenación en tiempo Lineal.

SECCIÓN 2

¿Cómo logramos ordenar n elementos de tamaño $w = \Omega((lgn)^{2+\varepsilon})$ en Tiempo Lineal?

Lo lograremos mediante la clasificacion de firma(Signature Sort),cuyo proceso de construcción y de requerimientos principales, se plasma en el diagrama de la figura 3.

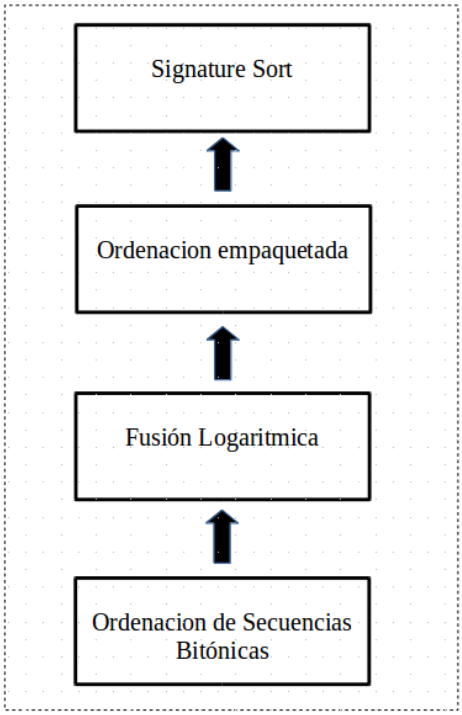

Figura3. Construccion de la clasificación Signature Sort.

SECCIÓN 3

Ordenamiento de Secuencias Bitónicas

Una secuencia bitónica es un cambio cíclico de una secuencia a_n, que aumenta de manera monótona seguida de una secuencia que disminuye de la misma manera. Tiene solo un mínimo local y/o un máximo local,como se muestra en la figura 4. El algoritmo de Ordenamiento de una Secuencia Bitonica consistirá en comparar-intercambiar los elementos a_i y $a_{i+n/2}$ de forma que el elemento izquierdo sea menor que el derecho,obteniendo dos nuevas secuencias bitónicas. Aplicando recursivamente este procedimiento,obtendremos listas bitonicas de tamaño 1, así la lista inicial de n elementos quedará ordenada,como se muestra en la figura 5.

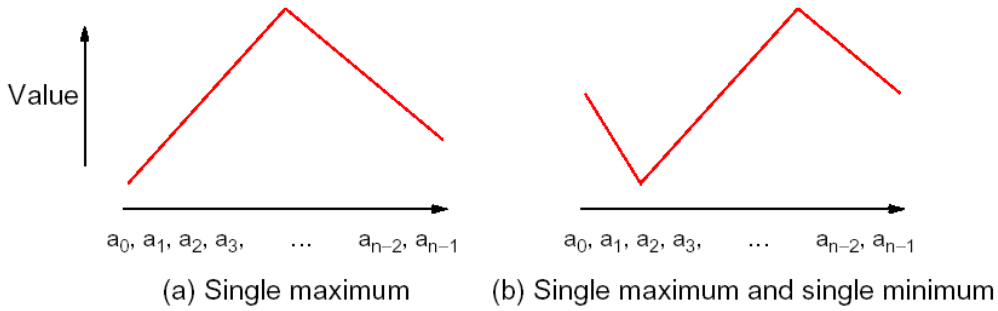

Figura 4. Los cambios cíclicos preservan la bitonicidad de una secuencia
Para ordenar una secuencia bitónica btcseq, ejecutamos el siguiente algoritmo.

Original sequence	3	5	8	9	10	12	14	20	95	90	60	40	35	23	18	0
1st Split	3	5	8	9	10	12	14	0	95	90	60	40	35	23	18	20
2nd Split	3	5	8	0	10	12	14	9	35	23	18	20	95	90	60	40
3rd Split	3	0	8	5	10	9	14	12	18	20	35	23	60	40	95	90
4th Split	0	3	5	8	9	10	12	14	18	20	23	35	40	60	90	95

Figura 5. Algoritmo de clasificación de secuencia bitónica

SECCIÓN 4

Operación de fusión logarítmica

Consiste en combinar dos palabras ordenadas.Para ello, concatenamos la primera palabra con el reverso de la segunda palabra (Para observar el proceso de revertir una palabra vea la Figura 6), obteniendo una secuencia bitónica.

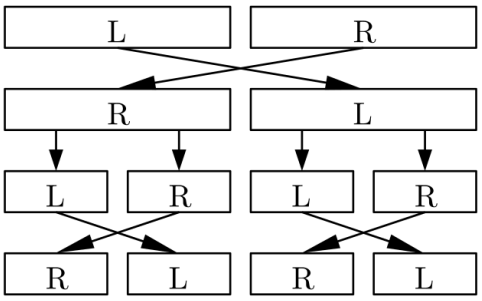

Figura 6. Los dos primeros pasos en la recursión para revertir una lista

En segundo Lugar, ordenar las secuencias bitonicas mediante el algoritmo de ordenacion Bitonico, solo que la unica diferencia es que el intercambio se dará en tiempo constante, aprovechando operaciones en bits, observar figura 7.

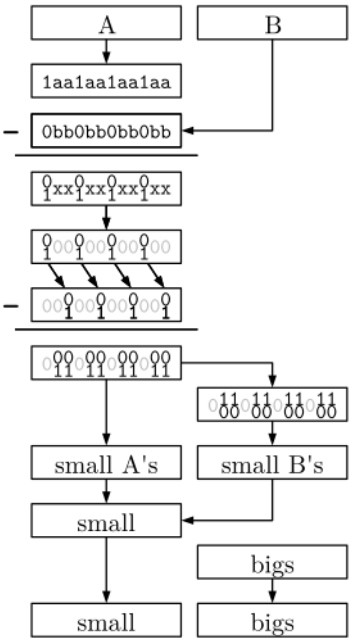

Figura 7. Operación de intercambio de las partes izquierda y derecha de una palabra en Tiempo constante.

Teorema 1. Suponga que dos palabras contienen k elementos de b-bits. Entonces podemos fusionar estas palabras en O (log k) tiempo.

SECCIÓN 5

Clasificación Empaquetada (Packed Sorting)

La clasificación empaquetada ordena n enteros de b-bits en tiempo O(n) para un tamaño de palabra de w $\geqslant$ 2(b + 1) log n log log n. Este límite para w nos permite empaquetar k = log n log log n elementos en una palabra, dejando un bit cero delante de cada entero, y $\frac{w}{2}$ bits cero al principio de la palabra.

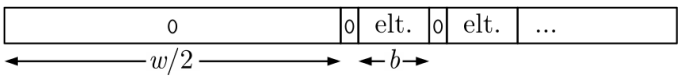

Figura 8. Estructura para empaquetar enteros de b-bit en una palabra de w-bit.
El proceso se ilustra aqui en la figura 9:

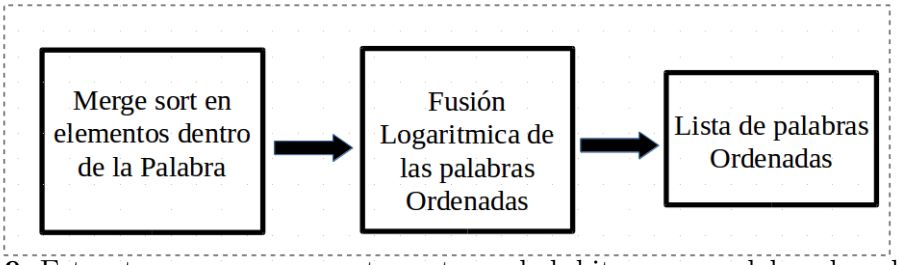

Figura 9. Estructura para empaquetar enteros de b-bit en una palabra de w-bit.

Teorema 2. Deje que el tamaño de la palabra $w \geqslant 2(b + 1) \log n \log \log n$. Luego, la ordenación empaquetada clasifica n enteros de b bits en tiempo O(n).

SECCIÓN 6

Algoritmo de orden de firma (Signature Sort)

1. Divida cada entero en $(\lg n)^{\varepsilon}$ trozos de igual tamaño.
2. Reemplace cada fragmento por un hash O(lg n)-bit.
3. Ordene las firmas en tiempo lineal con clasificación empaquetada.
4. Ahora queremos rescatar las identidades de las firmas. Construir un trie comprimido sobre las firmas.

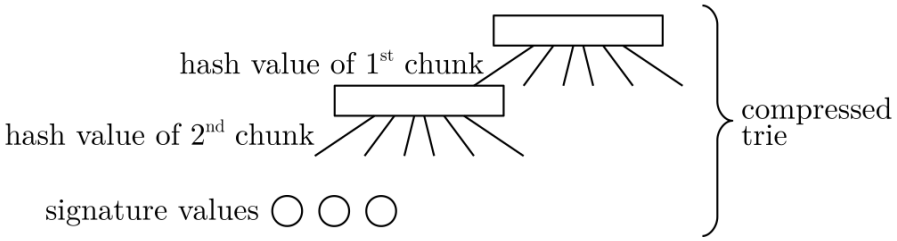

Figura 10. Un trie para el almacenamiento de firmas,las hojas representan posibles valores de firma.

5. Ordene recursivamente los bordes de cada nodo en el trie en función de sus valores reales.
6. Permuta los bordes del hijo.
7. Haga un recorrido inorder para obtener la lista ordenada deseada de las hojas.

Juntando estos pasos, obtenemos:

Teorema 3. Deje $w \geqslant (\log n)^{2+\varepsilon} \log \log n$. Luego, la ordenación de firmas ordenará n enteros de w-bit en tiempo O(n).

Prueba. Romper los números enteros en trozos y hacerlos hash (Pasos 1 y 2) toma tiempo lineal. Clasificación estos trozos de hash que usan ordenación empaquetada (Paso 3) también toman tiempo lineal. La creación del conjunto comprimido sobre las firmas toma tiempo lineal (Paso 4), y la clasificación de los bordes de cada nodo (Paso 5) toma tiempo constante por nodo para un número lineal de nodos. Finalmente, escaneando y permutando los nodos (Paso 6) y el recorrido inorder de las hojas del tree (Paso 7) tomarán un tiempo lineal, completando la prueba.

Referencias

[1] M. Thorup: Equivalence between Priority Queues and Sorting. FOCS 2002: 125-134 (2002).

[2] Demaine, Erik. [MIT OpenCourseWare]. (2013, Diciembre 18). 14. Sorting in Linear Time [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=pOKy3RZbSws

[3] Demaine, Erik. [MIT OpenCourseWare]. (2012, Abril 03). Lecture 14, Sorting in Linear Time. Advanced Data Structures [Archivo de texto]. 1- 8 pp. Recuperado de http://ocw.mit.edu