Ordenamiento Shell

De Wikipedia, la enciclopedia libre

El **ordenamiento Shell** (**Shell sort** en inglés) es un algoritmo de ordenamiento. El método se denomina **Shell** en honor de su inventor Donald Shell. Su implementación original, requiere $O(n^2)$ comparaciones e intercambios en el peor caso. Un cambio menor presentado en el libro de V. Pratt produce una implementación con un rendimiento de $O(n \log^2 n)$ en el peor caso. Esto es mejor que las $O(n^2)$ comparaciones requeridas por algoritmos simples pero peor que el óptimo $O(n \log n)$. Aunque es fácil desarrollar un sentido intuitivo de cómo funciona este algoritmo, es muy difícil analizar su tiempo de ejecución.

Proceso paso a paso de ordenamiento según el algoritmo de Shell.

El Shell sort es una generalización del ordenamiento por inserción, teniendo en cuenta dos observaciones:

1. El ordenamiento por inserción es eficiente si la entrada está "casi ordenada".
2. El ordenamiento por inserción es ineficiente, en general, porque mueve los valores sólo una posición cada vez.

El algoritmo Shell sort mejora el ordenamiento por inserción comparando elementos separados por un espacio de varias posiciones. Esto permite que un elemento haga "pasos más grandes" hacia su posición esperada. Los pasos múltiples sobre los datos se hacen con tamaños de espacio cada vez más pequeños. El último paso del Shell sort es un simple ordenamiento por inserción, pero para entonces, ya está garantizado que los datos del vector están casi ordenados.

Índice

Ejemplo

Considere un valor pequeño que está inicialmente almacenado en el final del vector que se quiere ordenar de forma ascendente. Usando un ordenamiento $O(n^2)$ como el ordenamiento de burbuja o el ordenamiento por inserción, tomará aproximadamente n comparaciones e intercambios para mover este valor hacia el otro extremo del vector. El Shell sort primero mueve los valores usando tamaños de espacio gigantes, de manera que un valor pequeño se moverá bastantes posiciones hacia su posición final, con sólo unas pocas comparaciones e intercambios.

Uno puede visualizar el algoritmo Shell sort de la siguiente manera: coloque la lista en una tabla y ordene las columnas (usando un ordenamiento por inserción). Repita este proceso, cada vez con un número menor de columnas más largas. Al final, la tabla tiene sólo una columna. Mientras que transformar la lista en una tabla hace más fácil visualizarlo, el algoritmo propiamente hace su ordenamiento en contexto (incrementando el índice por el tamaño de paso, esto es usando `i += tamaño_de_paso` en vez de `i++`).

Por ejemplo, considere una lista de números como `[ 13 14 94 33 82 25 59 94 65 23 45 27 73 25 39 10 ]`. Si comenzamos con un tamaño de paso de 5, podríamos visualizar esto dividiendo la lista de números en una tabla con 5 columnas. Esto quedaría así:

```
13 14 94 33 82
25 59 94 65 23
45 27 73 25 39
10
```

Entonces ordenamos cada columna, lo que nos da

```
10 14 73 25 23
13 27 94 33 39
25 59 94 65 82
45
```

Cuando lo leemos de nuevo como una única lista de números, obtenemos `[ 10 14 73 25 23 13 27 94 33 39 25 59 94 65 82 45 ]`. Aquí, el 10 que estaba en el extremo final, se ha movido hasta el extremo inicial. Esta lista es entonces de nuevo ordenada usando un ordenamiento con un espacio de 3 posiciones, y después un ordenamiento con un espacio de 1 posición (ordenamiento por inserción simple).

El Shell sort lleva este nombre en honor a su inventor, Donald Shell, que lo publicó en 1959. Algunos libros de texto y referencias antiguas le llaman ordenación "Shell-Metzner" por Marlene Metzner Norton, pero según Metzner, "No tengo nada que ver con el algoritmo de ordenamiento, y mi nombre nunca debe adjuntarse a éste." [1] (http://www.nist.gov/dads/HTML/shellsort.html)

Secuencia de espacios

La *secuencia de espacios* es una parte integral del algoritmo Shell sort. Cualquier secuencia incremental funcionaría siempre que el último elemento sea 1. El algoritmo comienza realizando un *ordenamiento por inserción con espacio*, siendo el espacio el primer número en la secuencia de espacios. Continua para realizar un ordenamiento por inserción con espacio para cada número en la secuencia, hasta que termina con un espacio de 1. Cuando el espacio es 1, el ordenamiento por inserción con espacio es simplemente un ordenamiento por inserción ordinario, garantizando que la lista final estará ordenada.

La secuencia de espacios que fue originalmente sugerida por Donald Shell debía comenzar con $N/2$ y dividir por la mitad el número hasta alcanzar 1. Aunque esta secuencia proporciona mejoras de rendimiento significativas sobre los algoritmos cuadráticos como el ordenamiento por inserción, se puede cambiar ligeramente para disminuir más el tiempo necesario medio y el del peor caso. El libro de texto de Weiss demuestra que esta secuencia permite un ordenamiento $O(n^2)$ del peor caso, si los datos están inicialmente en el vector como (pequeño_1, grande_1, pequeño_2, grande_2, ...) - es decir, la mitad alta de los números están situados, de forma ordenada, en las posiciones con índice par y la mitad baja de los números están situados de la misma manera en las posiciones con índice impar.

Quizás la propiedad más crucial del Shell sort es que los elementos permanecen k-ordenados incluso mientras el espacio disminuye. Se dice que un vector dividido en k subvectores esta k-ordenado si cada uno de esos subvectores está ordenado en caso de considerarlo aislado. Por ejemplo, si una lista fue 5-ordenada y después 3-ordenada, la lista está ahora no sólo 3-ordenada, sino tanto 5-ordenada como 3-ordenada. Si esto no fuera cierto, el algoritmo desharía el trabajo que había hecho en iteraciones previas, y no conseguiría un tiempo de ejecución tan bajo.

Dependiendo de la elección de la secuencia de espacios, Shell sort tiene un tiempo de ejecución en el peor caso de $O(n^2)$ (usando los incrementos de Shell que comienzan con n/2, n el tamaño del vector y se dividen por 2 cada vez), $O(n^{3/2})$ (usando los incrementos de Hibbard de $2^k - 1$), $O(n^{4/3})$ (usando los incrementos de

Sedgewick $O(n \log^2 n)$, y posiblemente mejores tiempos de ejecución no comprobados. La existencia de una implementación $O(n \log n)$ en el peor caso del Shell sort permanece como una pregunta por resolver.

NOTA: Los incrementos de Sedgewick se calculan intercalando los valores de las siguientes funciones:

$$f(i) = 9(4^i) - 9(2^i) + 1; f(0) = 1, f(1) = 19, f(2) = 109, f(3) = 505, \ldots)$$

$$g(i) = (2^{i+2}) * (2^{i+2} - 3) + 1; g(0) = 5, g(1) = 41, g(2) = 209, g(3) = 929, \ldots)$$

Referencias

- Weiss, Mark Allen (2002). *Data Structures & Problem Solving using Java*. Addison Wesley. ISBN 0-201-74835-5.
- Pratt, V (1979). *Shellsort and sorting networks (Outstanding dissertations in the computer sciences)*. Garland. ISBN 0-8240-4406-1. (Esto fue originalmente presentado como la tesis doctoral del autor en la Universidad de Stanford en 1971)

Enlaces externos

- Distintas implementaciones del algoritmo en Wikibooks (inglés) (https://en.wikibooks.org/wiki/Algorithm_Implementation/Sorting/Shell_sort)
- Distintas implementaciones del algoritmo en RosettaCode (inglés) (http://rosettacode.org/wiki/Sorting_algorithms/Shell_sort)
- Ordenamiento Shell en Python (http://web.archive.org/web/http://tutorial-python.com.ar/?p=136)
- Ordenación por Shell - From Scratch (http://snippets-tricks.org/ordenacion-shell/)