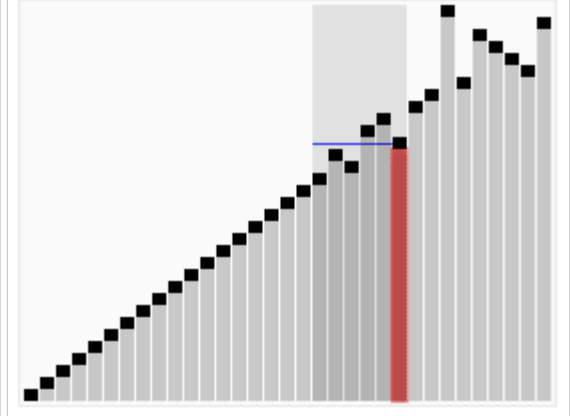
Quicksort

De Wikipedia, la enciclopedia libre

El **ordenamiento rápido** (**quicksort** en inglés) es un algoritmo creado por el científico británico en computación C. A. R. Hoare, basado en la técnica de divide y vencerás, que permite, en promedio, ordenar *n* elementos en un tiempo proporcional a *n* log *n*.



Quicksort en acción sobre una lista de números aleatorios. Las líneas horizontales son valores **pivote**.

Índice

Descripción del algoritmo

El algoritmo trabaja de la siguiente forma:

- Elegir un elemento de la lista de elementos a ordenar, al que llamaremos **pivote**.
- Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
- La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
- Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.

Como se puede suponer, la eficiencia del algoritmo depende de la posición en la que termine el pivote elegido.

- En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño. En este caso, el orden de complejidad del algoritmo es **O(n·log n)**.
- En el peor caso, el pivote termina en un extremo de la lista. El orden de complejidad del algoritmo es entonces de **O(n²)**. El peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas. Pero principalmente depende del pivote, si por ejemplo el algoritmo implementado toma como pivote siempre el primer elemento del array, y el array que le pasamos está ordenado, siempre va a generar a su izquierda un array vacío, lo que es ineficiente.

- En el caso promedio, el orden es **O(n·log n)**.

No es extraño, pues, que la mayoría de optimizaciones que se aplican al algoritmo se centren en la elección del **pivote**.

Demostración de un caso particular

Supongamos que el número de elementos a ordenar es una potencia de dos, es decir, $n = 2^k$ para algún natural k. Inmediatamente $k = log_2(n)$, donde k es el número de divisiones que realizará el algoritmo.

En la primera fase del algoritmo habrá n comparaciones. En la segunda fase el algoritmo instanciará dos sublistas de tamaño aproximadamente n/2. El número total de comparaciones de estas dos sublistas es: 2(n/2) = n. En la tercera fase el algoritmo procesará 4 sublistas más, por tanto el número total de comparaciones en esta fase es 4(n/4) = n.

En conclusión, el número total de comparaciones que hace el algoritmo es:

$n + n + n + \ldots . + n = kn$, donde $k = log_2(n)$, por tanto el Orden de Complejidad del algoritmo en el peor caso es $O(n. log_2 n)$.

Técnicas de elección del pivote

El algoritmo básico del método Quicksort consiste en tomar cualquier elemento de la lista al cual denominaremos como pivote, dependiendo de la partición en que se elija, el algoritmo será más o menos eficiente.

- Tomar un elemento cualquiera como pivote tiene la ventaja de no requerir ningún cálculo adicional, lo cual lo hace bastante rápido. Sin embargo, esta elección «a ciegas» siempre provoca que el algoritmo tenga un orden de $O(n^2)$ para ciertas permutaciones de los elementos en la lista.
- Otra opción puede ser recorrer la lista para saber de antemano qué elemento ocupará la posición central de la lista, para elegirlo como pivote. Esto puede hacerse en **O(n)** y asegura que hasta en el peor de los casos, el algoritmo sea **O(n·log n)**. No obstante, el cálculo adicional rebaja bastante la eficiencia del algoritmo en el caso promedio.
- La opción a medio camino es tomar tres elementos de la lista - por ejemplo, el primero, el segundo, y el último - y compararlos, eligiendo el valor del medio como pivote.

Técnicas de reposicionamiento

Una idea preliminar para ubicar el **pivote** en su posición final sería contar la cantidad de elementos menores que él, y colocarlo un lugar más arriba, moviendo luego todos esos elementos menores que él a su izquierda, para que pueda aplicarse la recursividad.

Existe, no obstante, un procedimiento mucho más efectivo. Se utilizan dos índices: **i**, al que llamaremos índice izquierdo, y **j**, al que llamaremos índice derecho. El algoritmo es el siguiente:

- Recorrer la lista simultáneamente con i y j: por la izquierda con i (desde el primer elemento), y por la derecha con j (desde el último elemento).
- Cuando lista[i] sea mayor que el pivote y lista[j] sea menor, se intercambian los elementos en esas posiciones.
- Repetir esto hasta que se crucen los índices.
- El punto en que se cruzan los índices es la posición adecuada para colocar el pivote, porque sabemos que a un lado los elementos son todos menores y al otro son todos mayores (o habrían sido intercambiados).

Transición a otro algoritmo

Como se mencionó anteriormente, el algoritmo quicksort ofrece un orden de ejecución **$O(n^2)$** para ciertas permutaciones "críticas" de los elementos de la lista, que siempre surgen cuando se elige el pivote «a ciegas». La permutación concreta depende del pivote elegido, pero suele corresponder a secuencias ordenadas. Se tiene que la probabilidad de encontrarse con una de estas secuencias es inversamente proporcional a su tamaño.

- Los últimos pases de quicksort son numerosos y ordenan cantidades pequeña de elementos. Un porcentaje medianamente alto de ellos estarán dispuestos de una manera similar al peor caso del algoritmo, volviendo a éste ineficiente. Una solución a este problema consiste en ordenar las secuencias

pequeñas usando otro algoritmo. Habitualmente se aplica el algoritmo de inserción para secuencias de tamaño menores de 8-15 elementos.
- Pese a que en secuencias largas de elementos la probabilidad de hallarse con una configuración de elementos "crítica" es muy baja, esto no evita que sigan apareciendo (a veces, de manera intencionada). El algoritmo introsort es una extensión del algoritmo quicksort que resuelve este problema utilizando heapsort en vez de quicksort cuando el número de recursiones excede al esperado.

Nota: Los tres parámetros de la llamada inicial a Quicksort serán array[0], 0, numero_elementos -1, es decir, si es un array de 6 elementos array, 0, 5

Ejemplo

En el siguiente ejemplo se marcan el pivote y los índices **i** y **j** con las letras **p**, **i** y **j** respectivamente.

Comenzamos con la lista completa. El elemento pivote será el 4:

```
5 - 3 - 7 - 6 - 2 - 1 - 4
                    p
```

Comparamos con el 5 por la izquierda y el 1 por la derecha.

```
5 - 3 - 7 - 6 - 2 - 1 - 4
i                 j   p
```

5 es mayor que 4 y 1 es menor. Intercambiamos:

```
1 - 3 - 7 - 6 - 2 - 5 - 4
i                 j   p
```

Avanzamos por la izquierda y la derecha:

```
1 - 3 - 7 - 6 - 2 - 5 - 4
    i         j       p
```

3 es menor que 4: avanzamos por la izquierda. 2 es menor que 4: nos mantenemos ahí.

```
1 - 3 - 7 - 6 - 2 - 5 - 4
        i       j       p
```

7 es mayor que 4 y 2 es menor: intercambiamos.

```
1 - 3 - 2 - 6 - 7 - 5 - 4
        i       j       p
```

Avanzamos por ambos lados:

```
1 - 3 - 2 - 6 - 7 - 5 - 4
          iyj           p
```

En este momento termina el ciclo principal, porque los índices se cruzaron. Ahora intercambiamos lista[i] con lista[sup] (pasos 16-18):

```
1 - 3 - 2 - 4 - 7 - 5 - 6
          p
```

Aplicamos recursivamente a la sublista de la izquierda (índices 0 - 2). Tenemos lo siguiente:

```
1 - 3 - 2
```

1 es menor que 2: avanzamos por la izquierda. 3 es mayor: avanzamos por la derecha. Como se intercambiaron los índices termina el ciclo. Se intercambia lista[i] con lista[sup]:

```
1 - 2 - 3
```

El mismo procedimiento se aplicará a la otra sublista. Al finalizar y unir todas las sublistas queda la lista inicial ordenada en forma ascendente.

```
1 - 2 - 3 - 4 - 5 - 6 - 7
```

Véase también

- Algoritmo de ordenamiento

Enlaces externos

- Distintas implementaciones del algoritmo en Wikibooks (inglés) (https://en.wikibooks.org/wiki/Algorithm_Implementation/Sorting/Quicksort)
- Distintas implementaciones del algoritmo en RosettaCode.org (inglés) (http://rosettacode.org/wiki/Sorting_algorithms/Quicksort)