# Wavelet Tree

Club de Algoritmia ESCOM

## 1. Introducción

Sea $A = \{a_1, a_2, \ldots, a_n\}$ un arreglo de números enteros. Esta estructura de datos nos ayuda a responder a las preguntas del tipo:

1. Hallar el numero de elementos en el subarreglo A[L ... R], donde $1 \leq L \leq R \leq n$, que son menores o iguales a un numero $y$.
2. Hallar el numero de veces que aparece un numero $x$ en el subarreglo A[L ... R].
3. Hallar el $k - esimo$ elemento del subarreglo, ordenado, A[L ... R].

Para resolver estas *queries* hay diferentes métodos, aquí presentaremos una estructura de datos que resuelve estas mismas con una complejidad de $O(lg(\sigma))$, donde $\sigma$ es el máximo elemento en el arreglo $A$.

## 2. Estructura

El Wavelet Tree es un árbol binario en el cual, cada nodo de este mismo es una subsecuencia del arreglo inicial. En el nodo raíz del árbol tendremos la secuencia asociada con todo el arreglo, es decir $A[1 \ldots n]$. Si en la secuencia asociada al nodo hay dos o mas valores distintos, estos valores se dividirán en dos subsecuencias, llamémosles $subL$ y $subR$, que formaran al hijo izquierdo y al derecho respectivamente. Este proceso se repetirá hasta que no haya dos o mas valores distintos en la subsecuencia asociada a cualquier nodo, en ese momento habremos llegado a un nodo $hoja$.

**¿Como sabremos que elementos van de cada lado?**

Teniendo en cuenta que cada subsecuencia tiene un valor mínimo y un máximo, los cuales tienen que ser diferentes a menos que se trate de un nodo hoja, definiremos a $m_S$ como un **entero** calculado de la forma $\frac{low+high}{2}$, donde $low$ es el valor mínimo en la subsecuencia y $high$ el valor máximo. Ya teniendo definido a $m_S$, todos los valores de la subsecuencia asociada al nodo, que sean menores o iguales a $m_S (a_i \leq m_S)$, formaran parte del hijo izquierdo, mientras los demás serán parte del derecho. Una condición importante es que al asociar estos elementos a un hijo, no se podrá alterar el orden en el que aparecen en la subsecuencia original. Presentaremos un ejemplo para visualizar esto:

Ejemplo: Sea $A = [3, 5, 3, 2, 6, 8, 8, 9, 2, 1, 4, 10, 7, 2, 9]$

Definiremos a $U$ como nuestro nodo inicial y a $S$ como la subsecuencia asociada a este mismo. Como establecimos anteriormente, nuestro nodo raíz tendrá a toda la subsecuencia original, por lo tanto $S = \{3, 5, 3, 2, 6, 8, 8, 9, 2, 1, 4, 10, 7, 2, 9\}$. Después, tenemos que $min(S) = 1 = low$ y $max(S) = 10 = high$ por lo tanto, $m_S = \frac{low+high}{2} = \frac{10+1}{2} = 5.5 = 5$ (tomando en cuenta el redondeo que hace el compilador).

Con el valor de $m_S$ calculado ya sabemos que números, contenidos en nuestro arreglo inicial, formaran parte de las subsecuencias de nuestros hijos. En la subsecuencia del hijo izquierdo irán los números menores a 5, los cuales son $\{1, 2, 3, 4, 5\}$ y en el hijo derecho serán $\{7, 8, 9, 10\}$.

Definiremos a $Left(U)$ y a $Right(U)$ como los hijos izquierdo y derecho respectivamente. También, sea $S'$ la subsecuencia asociada a $Left(U)$ y $S''$ la subsecuencia asociada a $Right(U)$. Cumpliendo con la condición establecida al final del párrafo anterior, $S' = \{3, 5, 3, 2, 2, 1, 4, 2\}$ y $S'' = \{6, 8, 8, 9, 10, 7, 9\}$. Los elementos que forman a $S'$ son los menores o iguales a $m_S$, y cumplen con la condición que aparecen en $S'$ en el mismo orden en el que aparecían en $S$; esto ultimo también ocurre con $S''$.

## 2.1. Mapeo

Necesitamos definir la operación de **mapeo** para poder entender como movernos(traverse) en el árbol. Esta operación mapea un índice de la subsecuencia asociada a un nodo, a los índices correspondientes en sus hijos izquierdo y derecho, cumpliendo una condición que explicaremos a continuación.

Para el siguiente ejemplo usemos los mismos datos del ejemplo anterior. Definamos a $mapLeft(S, i)$ y $mapRight(S, i)$ como las operaciones que mapean un índice de la subsecuencia $S$, asociada al nodo $U$, a las subsecuencias de sus hijos izquierdo y derecho respectivamente. Lo que harán estas operaciones sera contar el numero de elementos de la subsecuencia $S$ hasta el índice $i$ que irán a la subsecuencia del nodo hijo correspondiente según la operación elegida.

Realicemos la operación $mapLeft(S, 10)$. Podemos notar que el índice 10 (con indexación iniciada en 1) de la subsecuencia $S$ es el numero 1, por lo cual, contaremos todos los elementos desde 1 hasta $i = 10$ que formen parte de la subsecuencia $S'$. Así conseguimos que $mapLeft(S, 10) = 6$, por lo cual asociamos el índice 10 de la subsecuencia $S$ con el índice 6 de $S'$. Para el otro mapeo tendríamos que $mapRight(S, 10) = 4$ (**Proof left as an excercise to the reader**).

# 3. Operaciones

A continuación definiremos tres operaciones necesarias para poder resolver los tres tipos de pregunta que planteamos al inicio.

## 3.1. Rank

Lo que hace esta operación es contar el número de apariciones de un número $q$ hasta un índice $i$ de una subsecuencia $S$. De manera más formal se define como:

$$rank_q(S, i) = |\{k \in \{1, \ldots, i\} | s_k = q\}|$$

La manera de calcular $rank_q(S, i)$ es la siguiente:

Sea $U$ el nodo inicial de nuestro Wavelet Tree $T$, el cual tiene asociada la subsecuencia $S$, y queremos calcular $rank_q(S, i)$ con entero arbitrario $q$. Si $q \leq m_S$ entonces podemos afirmar que todas las ocurrencias de $q$ en $S$ estarán en el hijo izquierdo; por lo tanto:

$$rank_q(S, i) = rank_q(Left(U), mapLeft(S, i))$$

De otra manera, si $q > m_S$ entonces:

$$rank_q(S, i) = rank_q(Right(U), mapRight(S, i)$$

Este proceso se aplica hasta llegar a un nodo hoja, en ese caso podremos saber que $rank_q(S, i) = i$.

## 3.2. Range quantile

Dada una secuencia $S$,$quantile_k(S, i, j)$ encontrará el valor del k-ésimo elemento en la secuencia $(s_i, s_{i+1}, \ldots, s_j)$, si esta estuviera ordenada. Primero definiremos una versión más fácil de esta operación al fijar a $i = 1$. Ahora, haremos uso de nuestra operación de mapeo:

Recordemos que $mapLeft(S, i)$ cuenta cuantos elementos de $S$ hasta el índice $i$ se encuentran en el hijo izquierdo. Entonces, si tenemos que $k \leq mapLeft(S, j)$ podemos asegurar que el elemento que estamos buscando se encuentra en la subsecuencia asociada al hijo izquierdo. Por lo tanto podremos avanzar en el árbol de la forma:

$$quantile_k(Left(U), 1, mapLeft(S, j))$$

.

De otra manera, si $k > mapLeft(S, j)$, el elemento estará en el lado derecho. Al momento de querer avanzar al lado derecho debemos tomar en cuenta que ya hemos descartado todos los elementos del hijo izquierdo, por lo que nuestro $k$ cambiara. Sea $m = mapLeft(S, j)$, entonces la función que usaremos para avanzar en el árbol sera:

$$quantile_{k-c}(Right(U), 1, mapRight(S, j))$$

.

Retomando la query inicial que es $quantile_k(S, i, j)$ haremos uso de la estrategia planteada anteriormente. Primero hay que darnos cuenta que los elementos dentro el rango $i$ a $j$ que irán al lado izquierdo es igual a: $mapLeft(S, j) - mapLeft(S, i - 1)$. Si $k \leq mapLeft(S, j) - mapLeft(S, i - 1)$ podemos asegurar que el k-esimo elemento estará del lado izquierdo, por lo que de igual manera definiremos nuestra función para avanzar hacia aquel lado. Esta seria:

$$quantile_k(Left(U), mapLeft(S, i - 1) + 1, mapLeft(S, j))$$

En el caso de que $k > mapLeft(S, j) - mapLeft(S, i - 1)$, definiremos a $c$ igual a la resta del mapeo de $j$ menos el de $i - 1$. Nuestra función para ese lado seria:

$$quantile_{k-c}(Right(U), mapRight(S, i - 1) + 1, mapRight(S, j))$$

El proceso se repetirá hasta llegar a un nodo hoja, el cual contendrá el valor del k-esimo elemento que buscamos.

### 3.3. Range counting

Esta operación, definida de la manera $range_{[x,y]}(S, i, j)$, cuenta el numero de elementos con valores entre $x$ y $y$ en todas las posiciones entre los índices $i$ y $j$.

## 4. Referencias

Paper: "Wavelet Trees for Competitive Programming"