Treaps y árboles binarios de búsqueda aleatorios

Antonio Jesús Otaño Barrera
Grupo: *C-411*

December 7, 2020

1 Introducción

El trabajo y las operaciones eficientes sobre conjuntos es un requisito indispensable para todo programador competitivo. Existen numerosas estructuras de datos destinadas a tal objetivo, como son los árboles rojo-negros y los AVL, capaces de realizar las operaciones de búsqueda, inserción y eliminación en complejidad temporal $O(\log n)$. Sin embargo, la complejidad de su implementación las hace poco recomendables para ser usada en concursos que duran pocas horas. La alternativa presentada en este artículo es una estructura de datos conocida como árbol binario de búsqueda aleatorio (randomized binary search tree), que es un tipo particular de una estructura de datos llamada Treap, ambas presentadas en [1]. En un árbol binario de búsqueda aleatorio las operaciones de búsqueda, inserción y eliminación se comportan como $O(\log n)$ en el caso promedio y en la práctica es tan efectivo su uso como el de las otras estructuras anteriormente mencionadas.

2 Treaps

Un treap es un árbol binario de búsqueda con respecto a las llaves que guardan los nodos, y donde además cada nodo una tiene una prioridad de tal manera que para todo nodo y que se encuentra en la descendencia de un nodo x se cumple que $prioridad(y) \leq prioridad(x)$ (Figura 1).

Si asumimos que todas las llaves son distintas al igual que todas las prioridades entonces es fácil ver que el Treap que determinan es único: el nodo con mayor prioridad será la raíz y tanto la descendencia izquierda como la derecha, que cumplen con la condición de árbol binario de búsqueda respecto a las llaves se construyen recursivamente. Visto de otra forma, el Treap resultante de un conjunto X es el árbol binario de búsqueda usual, resultante de insertar los elementos ordenados por prioridad en orden descendente.

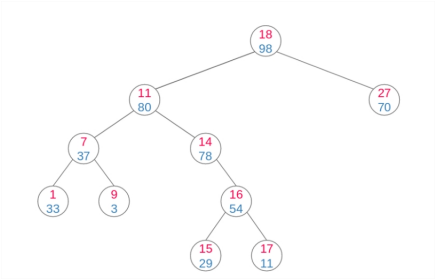

Figure 1: Ejemplo de Treap. Los valores en rojo corresponden a las llaves y los valores en azul a las prioridades

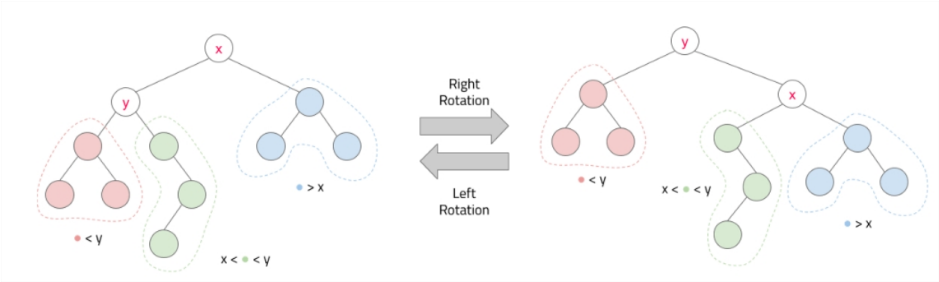

Figure 2: Rotaciones en el Treap

2.1 Operaciones sobre los Treaps

Para mantener consistente la estructura del Treap es necesario realizar rotaciones sobre los nodos necesarios durante las operaciones de inserción y eliminación. Dichas rotaciones son las mismas que se realizan sobre otras estructuras como los árboles AVL (Figura 2).

El acceso a un elemento es similar al que se realiza en un árbol binario de búsqueda corriente. A la hora de insertar primero se procede como en un árbol binario de búsqueda normal. Luego se comprueba si la prioridad del padre del nuevo nodo es mayor que la de este. En caso afirmativo termina el procedimiento, sino corresponde realizar una rotación para arreglar este conflicto. Si el nuevo nodo raíz del subárbol resultante de la rotación cumple que su prioridad es menor que la de su padre o es la raíz de la estructura más general, termina el procedimiento, en caso contrario continúa ascendiedo el nodo mediante rotaciones hasta cumplirse una de las dos condiciones de parada mencionadas anteriormente (Figura 3). La eliminación es exactamente el procedimiento inverso a la inserción: se busca el nodo a eliminar, y se comienza a bajar por todo el árbol mediante rotaciones hasta que llegue a convertirse en una hoja, de tal forma que no haya conflictos luego de su eliminación. Como se puede ver la complejidad temporal de estas operaciones está determinada por la altura del árbol.

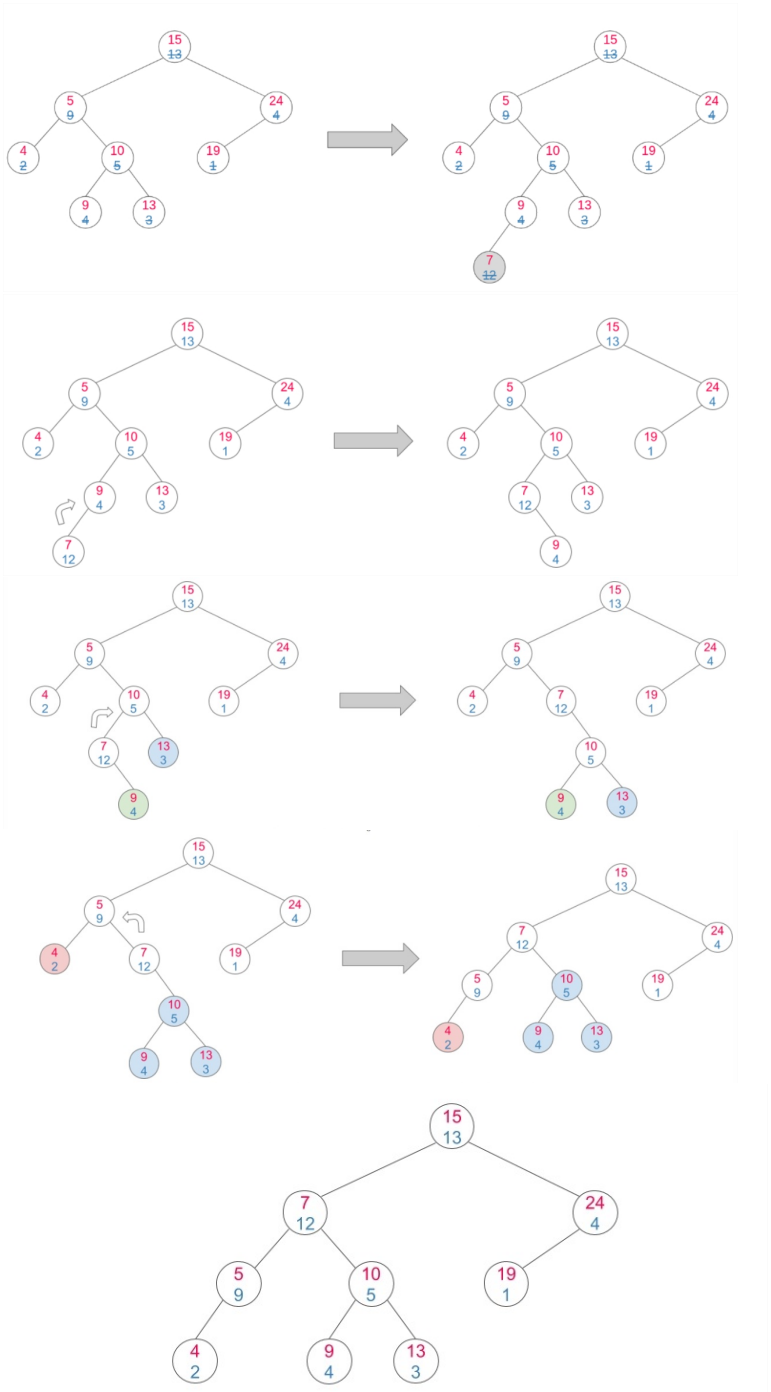

Figure 3: Inserción / eliminación en un Treap. De arriba hacia abajo se muestra el procedimiento para insertar el nodo con llave 7 y prioridad 12. En el sentido contrario se puede analizar como ocurre la eliminación de ese mismo nodo.

3 Árboles binarios de búsqueda aleatorios

Definición: Sea X un conjunto de elementos, cada uno de los cuales posee una llave extraída de un conjunto totalmente ordenado. Un árbol binario de búsqueda aleatorio de X es un Treap de X donde las prioridades de los nodos son variables aleatorias continuas independientes e idénticamente distribuidas.

El requisito de que las variables aleatorias sean continuas no es realmente necesario. Solo se pone en la definición para asegurarse de que todas las prioridades sean distintas con probabilidad 1. En la práctica basta con generar enteros aleatorios en un rango lo suficientemente grande como para que la probabilidad de que muchas prioridades sean iguales sea muy pequeña.

3.1 Análisis de la altura de los árboles binarios de búsqueda aleatorios

En lo que sigue estaremos trabajando con el conjunto $X = \{x_i = (k_i, p_i) \mid 1 \leq i \leq n\}$ donde (k_i, p_i) son las respectivas llave y la prioridad del elemento x_i. Se asume que $k_i < k_{i+1}$ para $1 \leq i \leq n-1$.

Estudiar el comportamiento de la altura de un árbol binario de búsqueda aleatorio directamente es bastante complicado. En su lugar trabajaremos con las siguiente definiciones:

Definición: Sea T un Treap sobre X.

1. Llamaremos *D(x)* a la profundidad de un nodo x en T, i.e la cantidad de nodos que hay en el camino de la raíz de T hasta x.

2. Llamaremos $A_{i,j}$ a la función que nos indica si x_i es ancestro de x_j. Se considera que un nodo es ancestro de sí mismo. Formalmente:

$$A_{i,j} = \begin{cases} 1 & \text{si } x_i \text{ es ancestro de } x_j \\ 0 & \text{e.o.c} \end{cases}$$

Un resultado inmediato de las anterirores definiciones es la siguiente igualdad:

$$D(x_l) = \sum_{1 \leq i \leq n} A_{i,l}$$

pues la profundidad de un nodo no es otra cosa que el número de sus ancestros.

Dado que para un conjunto fijo de llaves la forma del Treap depende de las prioridades, las cuales son variables aleatorias, se tiene que tanto $D(x)$ como $A_{i,j}$ serán también variables aleatorias, por lo cual tiene sentido analizar su valor esperado. Sea $a_{i,j} = Ex[A_{i,j}]$. A partir de la linealidad del valor esperado se obtiene:

$$Ex[D(x_l)] = \sum_{1 \leq i \leq n} a_{i,l}$$

Además como $A_{i,j}$ es una Bernoulli, su valor esperado se puede calcular como:

$$a_{i,j} = Ex[A_{i,j}] = P(A_{i,j} = 1) = P(x_i \text{ sea ancestro de } x_j)$$

Para determinar esta probabilidad nos apoyaremos en el siguiente lema que nos permitirá caracterizar completamente la relación de ancestro.

Lema: Sea T un Treap sobre X donde todas las prioridades de sus nodos son distintas y sean $1 \leq i, j \leq n$. Entonces x_i es ancestro de x_j si y solo si $x_i = max\{x_h \mid i \leq h \leq j\}$

Demostración: Sea x_m el elemento con la mayor prioridad en T y sean $X' = x_v \mid 1 \leq v < m$ y $X'' = x_u \mid m < u \leq n$. Por definición de Treap, x_m será la raíz de T y sus subárboles izquierdo y derecho, T' y T" serán Treaps para X' y X" respectivamente. Obviamente todo par de nodos que involucre al nodo raíz cumplirá con la caracterización de ancestro propuesta. También se puede verificar que para todo par de nodos v ∈ T', u ∈ T", el nodo x_m se encuentra en el camino de v a u, por lo que ni u será ancestro de v, ni v de u. Finalmente, por inducción la caracterización es correcta para todo par de nodos de T' y para todo par de nodos de T".

A partir del anterior Lema es fácil determinar la probabilidad de que un nodo sea ancestro de otro:

Corolario: En un árbol binario de búsqueda aleatorio T sobre el conjunto X, se cumple que x_i es ancestro de x_j con probabilidad $\frac{1}{|i-j|+1}$.

Demostración: Para que x_i sea ancestro de x_j necesitamos que x_i tenga la mayor prioridad entre los $|i-j|+1$ elementos que hay entre x_i y x_j. Dado que las prioridades son variables aleatorias continuas independientes e idénticamente distribuidas esto ocurre con probabilidad $\frac{1}{|i-j|+1}$

De ester resultado se desprende que:

$$a_{i,j} = \frac{1}{|i-j|+1}$$

Ahora estamos en condiciones de determinar una cota superior para el valor esperado de $D(x)$. Para ello recordemos que el k-ésimo numéro armónico se define como $H_k = \sum_{1 \leq i \leq k} \frac{1}{k}$ y que se cumple la desigualdad: $\ln k < H_k < \ln k + 1$ para $n > 1$.

Lema: Sea T un árbol binario de búsqueda aleatorio de n nodos sobre X y sea $1 \leq l \leq n$. Entonces se cumple que:

$$Ex[D(x_l)] = H_l + H_{n+1-l} - 1 < 1 + 2 \ln n$$

Demostración:

$$\begin{aligned} Ex[D(x_l)] = \sum_{1\leq i\leq n} a_{i,l} &= \sum_{1\leq i\leq n} \frac{1}{|i-l|+1} \\ &= \sum_{1\leq i\leq l} \frac{1}{|i-l|+1} + \sum_{l+1\leq i\leq n} \frac{1}{|i-l|+1} \\ &= H_n + H_{n+1-l} - 1 \\ &< 1 + \ln n + \ln(n+1-l) \\ &\leq 1 + \ln n + \ln n = 1 + 2\ln n \end{aligned}$$

Para concluir necesitamos determinar cuan probable es que la profundidad de un nodo sea mayor que $1 + 2\ln n$. El resultado se muestra a continuación.

Teorema: Sea T un árbol binario de búsqueda aleatorio de $n > 1$ nodos sobre X. Se tiene que para cualquier $1 \leq l \leq n$ y para cualquier $c > 1$ se cumple que:

$$P(D(X_l) \geq 1 + 2c\ln n) < 2(n/e)^{-c\ln(c/e)}$$

Demostración: Sabemos que $D(X_l) = \sum_{i=1}^{n} A_{i,l}$. Sean $L = \sum_{i=1}^{l-1} A_{i,l}$ y $R = \sum_{i=l+1}^{n} A_{i,l}$. Luego $D(x) = 1 + L + R$. Para que $D(x_l)$ sea mayor que $1 + 2c\ln n$ al menos uno entre L y R tiene que ser mayor que $c\ln n$. Asumamos que es R. El caso para L es simétrico.

Para $i > l$ las variables A_i, l son independientes y entonces se pueden aplicar las cotas de Chernoff [2], [3], específicamnete una de sus variantes que plantea que si Z es una suma de Bernoulli independientes, entonces:

$$P(Z \geq c \cdot Ex[Z]) < e^{-c\ln(c/e)Ex[Z]}$$

.

Para hacer esta cota independiente de l, consideremos la variable aleatoria $R' = \sum_{l<i<l+n} A_{i,l}$ donde usamos nuevas variables Bernoulli independientes $A_{i,l}$ para $i > n$, con $Ex[A_{i,l}] = \frac{1}{i-l+1}$. De esta forma $Ex[R'] = H_n - 1$. Además se sabe que para $k > 0$ se cumple que $P(R \geq k) \leq P(R' \geq k)$. Por último conocemos que $\ln n - 1 < H_n - 1 < \ln n$. Juntando todo obtenemos el resultado deseado:

$$\begin{aligned} P[R \geq c\ln n] &\leq P(R' \geq c\ln n) \\ &< P(R' \geq c(H_n - 1) \\ &< e^{-c\ln(c/e)(H_n-1)} \\ &< e^{-c\ln(c/e)(\ln n-1)} = (n/e)^{-c\ln(c/e)} \end{aligned}$$

Luego, la probabilidad de que al menos uno entre L y R sea mayor que $c\ln n$ será $2(n/e)^{-c\ln(c/e)}$. □

Este resultado implica que la probabilidad de que al menos uno de los n nodos de T tenga una profundidad mayor o igual que $1 + 2c \ln n$ esté acotada superiormente por $2n(n/e)^{-c \ln(c/e)}$. Escogiendo un valor fijo y adecuado para c esta probabilidad será sumamente pequeña por lo que en el caso promedio podemos afirmar que la profundidad de un nodo se comporta logarítmica, y por tanto la altura del Treap.

References

[1] R. Seidel, C. R. Aragon, *Randomized Search Trees*, Algorithmica, (1996).

[2] T. Hagerup, C. Rub, A guided tour of Chernoff bounds, Inform.Process.Lett, (1989-1990)

[3] K.L. Clarkson, K. Mehlhorn and R. Seidel, Four results on randomized incremental construction, Comput. Geom. Theory Appl., (1993)