Análisis de Árboles Binarios Aleatorizados

Miguel Oviedo

1. Resumen

Los árboles son estructuras de datos que pueden organizar elementos con ciertas características básicas como elementos padre, hijos, antescesores o sucesores. Además se pueden implementar variar operaciones como INSERCIÓN, ELIMINACIÓN, BÚSQUEDA, etc. Presentan una gran cantidad de variantes y su uso dependen en gran medida de la aplicación que se necesite. En este apartado presentamos en concreto la inserción aleatoria de elementos en los árboles de búsqueda binaria que es uno de los árboles más básicos. Analizamos la esperanza matemática de dichos elementos equiprobables asi como una implementación de su código en R.

2. Introducción

2.1. Árboles de búsqueda binaria

Un árbol de búsqueda binaria es una estructura de datos para contener datos ordenados en forma de aŕbol binario. Un árbol binario está vacío o consta de un nodo raíz que contiene un elemento y apunta a los subárboles izquierdo y derecho. La propiedad fundamental en árboles binarios es que tanto en un árbol como un subárbol, todos los elementos del subárbol izquierdo son menores que el elemento raíz, mientras que todos los elementos del subárbol derecho son mayores que que el elemento raíz. Esta distribución implica que podemos buscar un elemento particular haciendo una búsqueda binaria: si un elemento no está en la raíz, podemos recurrir en el subárbol izquierdo o derecho dependiendo de si es más pequeño o más grande que el elemento de la raíz.

Los árboles binarios soportan muchas operaciones dinámicas como búsqueda, inserción, eliminación, etc de un elemento. Las operaciones básicas en un árbol de búsqueda binaria toman tiempo proporcional a la altura del árbol. Para un árbol debidamente construido (completo) con n nodos toma un tiempo de $\Theta(lgn)$ en el mejor caso, pero puede tomar $O(n)$ para las mismas operaciones en el peor caso si el árbol forma una cadena lineal de n nodos como se muestra en la figura 2.

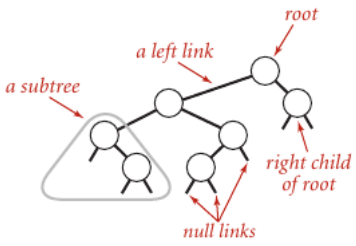

Figura 1: Anatomía de un árbol binario

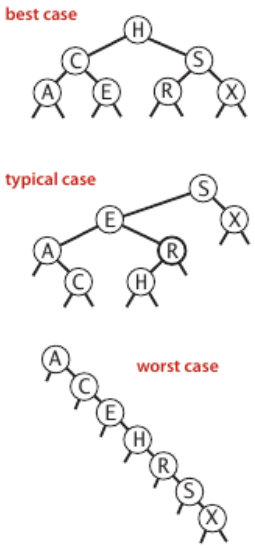

Figura 2: Posibilidades de un árbol binario

3. Construcción de un árbol aleatorizado

Si insertamos n elementos en un árbol de búsqueda binaria inicialmente vacío en orden aleatorio cada elemento seguirá las reglas que definen a un árbol binario y cuando llegamos a un árbol vacío, lo reemplazamos con un árbol que consiste únicamente en un elemento que es la raíz. Como elegimos un orden aleatorio, cada elemento tiene la misma probabilidad de ser la raíz, y todos los elementos menores que la raíz terminan en el subárbol izquierdo, mientras que todos los elementos superiores a la raíz terminan en el subárbol derecho.

Sea X_n la variable aleatoria que denota la altura del árbol aleatorizado de n elementos, y definimos la variable aleatoria de la *altura exponencial* Y_n como $Y_n = 2^{X_n}$. Cuando construimos un árbol de búsqueda binario en n teclas, elegimos una clave como la de la raíz, y dejamos que Rn denote la variable aleatoria que mantiene el rango de esta clave dentro del conjunto de n teclas; es decir, Definimos, por último, R_n como la variable aleatoria que mantiene la posición que ocuparía un elemento si se ordenara como un vector ordenado. EL valor de R_n es igualemente probable de ser cualquier elemento del conjunto $1, 2, \ldots, n$. Si $R_n = i$, entonces el subárbol izquierdo de la raíz es un subárbol binario aleatorizado de $i - 1$ elementos, y el subárbol derecho de la raíz es un subárbol binario aleatorizado de $n - 1$ elementos. Debido a que la altura del árbol es uno más que cualquiera de los dos subárboles que tienen a la raíz como padre, la altura exponencial del árbol binario es dos veces más largo que la altura exponencial de cualquiera de los subárboles,es decir, si $R_n = i$, entonces:

$$Y_n = 2 \cdot \text{máx}(Y_{i-1}, Y_{i+1})$$

Para el caso base tenemos que $Y_1 = 1$, debido a la altura exponencial del árbol con 1 nodo es $2^0 = 1$ y por conveniencia definimos $Y_0 = 0$. Definimos la variable indicadora $Z_{n,1}, Z_{n,2}, \ldots, Z_{n,n}$, donde $Z_{n,i} = I\{R_n = i\}$.
Debido a que R_n es igualmente probable para ser cualquier elemento de $\{1, 2, \ldots, n\}$, entonces $\mathbb{P}\{R_n = i\} = 1/n$ para $i = 1, 2, \ldots, n$, por lo tanto $E[Z_{n,i}] = 1/n$, para $i = 1, 2, \ldots, n$ Debido a que un valor de $Z_{n,i}$ is 1 y todos los demás son 0, también tenemos

$$Y_n = \sum_{i=1}^{n} Z_{n,i}(2 \cdot \text{máx}(Y_{i-1}, Y_{i+1}))$$

Podemos afirmar que el indicador $Z_{n,i} = IR_n = i$ es independiente de los valores Y_{i-1} y Y_{n-i}. Eligiendo $R_n = i$, el subárbol izquierdo es aleatorizado con $i - 1$ elementos cuyos valores son menores que i. Además de la cantidad de elementos que contiene la estructura de este subárbol no se ve afectada por la elección de $R_n = i$, por lo tanto las variables aleatorias Y_{i-1} y $Z_{n,i}$ son independiendes. De la misma manera para el subárbol derecho, cuya altura exponencial es Y_{n-i} es aleatorizado con $n-i$ elementos cuyos valores son mayores que i. Esta estructura también es independiente de los valores de R_n, y por lo tanto las variables

aleatorias Y_{n-i} y $Z_{n,i}$ son independiendes. Por lo tanto tenemos

$$\begin{aligned}
E[Y_n] &= E\left[\sum_{i=1}^{n} Z_{n,i}(2 \cdot \text{máx}(Y_{i-1}, Y_{n-i}))\right] \\
&= \sum_{i=1}^{n} E\big[Z_{n,i}(2 \cdot \text{máx}(Y_{i-1}, Y_{n-i}))\big] \\
&= \sum_{i=1}^{n} E\big[Z_{n,i}\big] E\big[(2 \cdot \text{máx}(Y_{i-1}, Y_{n-i}))\big] \\
&= \sum_{i=1}^{n} \frac{1}{n} \cdot E\big[(2 \cdot \text{máx}(Y_{i-1}, Y_{n-i}))\big] \\
&= \frac{2}{n} \sum_{i=1}^{n} E\big[(\text{máx}(Y_{i-1}, Y_{n-i}))\big] \\
&\leq \frac{2}{n} \sum_{i=1}^{n} (E[Y_{i-1}] + E[Y_{n-i}])
\end{aligned}$$

Además cada término $E[Y_i]$ se repite dos veces, una vez como $E[Y_{i-1}]$ y otra como $E[Y_{n-i}]$, entonces tenemos la recurrencia

$$E[Y_n] \leq \frac{4}{n} \sum_{i=0}^{n-1} E[Y_i]$$

Para encontrar el tiempo de implementación usaremos dos resultados. Usando el método de sustitución obtenemos

$$E[Y_n] \leq \frac{1}{4} \binom{i+3}{3}$$

y apoyándonos de la siguiente identidad tenemos

$$\sum_{i=0}^{n-1} \binom{i+3}{3} = \binom{n+3}{4}$$

Ahora para el caso base, tenemos $0 = Y_0 = E[Y_0] \leq (1/4)\binom{3}{3}1/4$

$$\begin{aligned}
&\leq \frac{4}{n}\sum_{i=0}^{n-1}\frac{1}{4}\binom{i+3}{3} \\
&= \frac{1}{n}\sum_{i=0}^{n-1}\binom{i+3}{3} \\
&= \frac{1}{n}\binom{n+3}{4} \\
&= \frac{1}{n}\cdot\frac{(n+3)!}{4!(n-1)!} \\
&= \frac{1}{4}\cdot\frac{(n+3)!}{3!n!} \\
&= \frac{1}{4}\binom{n+3}{3}
\end{aligned}$$

Por último, usamos una desigualdad notable conocida como la ***desigualdad de Jensen*** el cual nos permite calcular desigualdades de la forma $E[f(X)] \geq f(E[X])$, empleando esta desigualdad a $E[Y_n]$ tenemos

$$\begin{aligned}
2^{E[X_n]} &\leq \frac{1}{4}\binom{n+3}{3} \\
&= \frac{1}{4}\cdot\frac{(n+3)(n+2)(n+1)}{6}
\end{aligned}$$

Tomando logaritmos a ambos lados

$$\begin{aligned}
lg(2^{E[X_n]}) &= lg[\frac{1}{24}\cdot(n+3)\cdot(n+2)\cdot(n+1)] \\
E[X_n] &= O(lgn)
\end{aligned}$$

El resultado final indica que el tiempo de implementación esperado de un árbol de búsqueda binario con elementos insertados aleatoriamente es $O(lgn)$.

4. Conclusión

- Según los cálculos obtenidos de diferentes pruebas no hay un punto de comparación computaciones para un rango de hasta cien mil elementos debido que la potencia de cómputo de una computadora éstandar es mayor al igual como en el algoritmo de quicksort, pero matemáticamente se demuestra que la inserción de elementos aleatorizados tiene tiempo computacional aceptable.

Referencias

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 2009.

[2] PKS Prakash, Achyutuni Sri Krishna Rao *R Data Structures and Algorithms.* Packt Publishing, 2016

[3] James Aspnes *Notes on Randomized Algorithms CPSC 469/569: Fall 2016*