Universidad Teconologica de Pereira

Pereira

Estructura de Datos

Analisis proyecto de aleatoriedad

Aplicado en BST

Santiago Castañeda Pérez
October 7, 2024

Abstract

Este documento presenta un análisis del proyecto de aleatoriedad destinado a árboles de búsqueda binaria (BST). El objetivo es explorar diversas propiedades de los árboles generados,

1 Introducción

El árbol de búsqueda binaria (BST) es una estructura de datos fundamental que permite realizar operaciones de búsqueda, inserción y eliminación de manera eficiente. Este análisis se centra en las características de los BST generados a partir de un conjunto de 100 datos basados en nombres de distintas cintas audiovisuales.

```
1   //Lista de 100 peliculas
2   string movies[100] = {
3       "Inception", "Interstellar", "Gladiator", "
          Titanic", "Avatar", "Pulp", "Fight", "Forrest
          ", "Star", "Jurassic",
4       "Alien", "Blade", "Schindlers", "Braveheart", "
          Memento", "Se7en", "Goodfellas", "Psycho", "
          Vertigo", "Jaws",
5       "Rocky", "Taxi", "Birdman", "Amelie", "Akira", "
          Paprika", "Summer", "Wolf", "Your", "Matrix",
6       "Gravity", "Frozen", "Coco", "Up", "Cars", "
          Ratatouille", "Shrek", "Moana", "Zootopia", "
          Tangled",
7       "Brave", "Bolt", "Hercules", "Tarzan", "Mulan",
          "Aladdin", "Pocahontas", "Bambi", "Dumbo", "
          Pinocchio",
8       "Cinderella", "Fantasia", "Frozen", "Tangled", "
          Brave", "Bolt", "Hercules", "Tarzan", "Mulan"
          , "Aladdin",
9       "Pocahontas", "Bambi", "Dumbo", "Pinocchio", "
          Cinderella", "Fantasia", "Frozen", "Tangled",
           "Brave", "Bolt",
10      "Hercules", "Tarzan", "Mulan", "Aladdin", "
          Pocahontas", "Bambi", "Dumbo", "Pinocchio", "
          Cinderella", "Fantasia",
11      "Frozen", "Tangled", "Brave", "Bolt", "Hercules"
          , "Tarzan", "Mulan", "Aladdin", "Pocahontas",
           "Bambi"
12  };
```

2 Metodología

2.1 Generación de BST

La Lista de datos anteriormente mencionada es insertada en un Vector y posteriormente se ordena alfabeticamente, por último estos datos son tomados de manera aleatoria para ser incluidos en el BST y de esta forma proporcionar distintas formas del mismo, sin tomar un orden en especifico.

```
Vector<string> v;
    for (int i = 0; i < 100; i++) {
        v.push_back(movies[i]);
    }
    // v.print(); //imprimir el vector
    v.sort(); //ordenar el vector
    BST<string, int> bst; //creacion de un arbol binario
        de busqueda
    //seleccion de los elementos aleatorios del vector
    srand(time(0));
    for (int i = 0; i < 100; i++) {
        int index = rand() % v.getSize();
        bst.insert(v.at(index), i);
        v.remove(index);
    }
```

2.2 Recolección de Datos

Los datos recopilados del BST son su altura y el número de hojas, Es importante destacar que un árbol profundo con pocas hojas puede ser menos eficiente. En un árbol perfectamente balanceado, el número de hojas debería ser proporcional a la profundidad del árbol.

3 Análisis

3.1 Análisis de Altura

Para mi analisis, decidi importar un conjunto de herramientas proporcionadas por Graphviz, la cual me permitio visualizar el BST posterior a su creación, como por ejemplo el siguiente arbol:

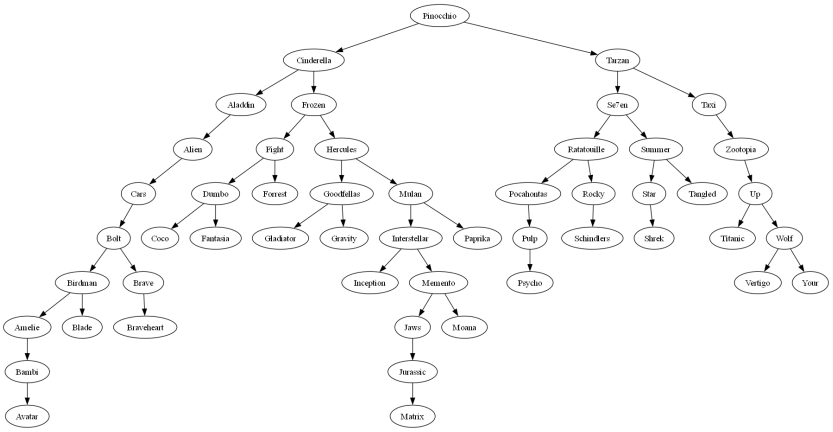

Figure 1: Heigth: 10, Leaves: 20

Este arbol contiene precisamente una cantidad significativa de menos elementos que el vector, ya que algunos datos se repiten, lo hice por la unica razon de demostrar que la inserción del BST no permite duplicar el mismo dato. Por lo general la distribución de hojas suele ser mucho mayor a la profundidad o altura del arbol,

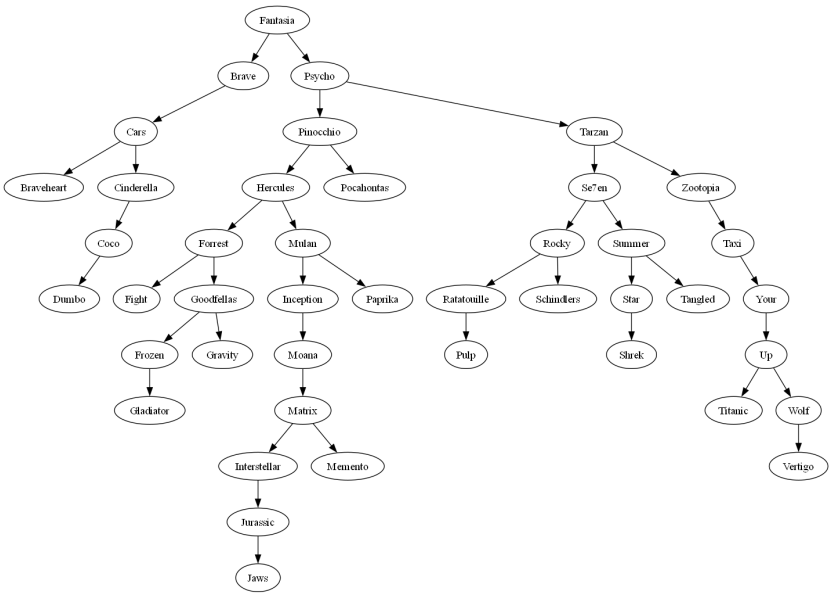

Figure 2: Heigth: 11, Leaves: 15

La cantidad significativamente mas grande que existe entre hojas y altura

indica un árbol potencialmente desbalanceado y no óptimo. Esto podría afectar negativamente la eficiencia de las operaciones de búsqueda, inserción y eliminación. Lo que determino recomendable son las tecnicas de balanceo proximas a investigar.

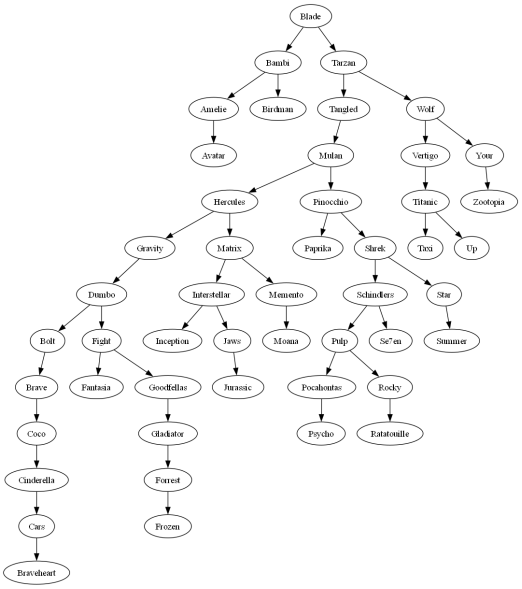

Figure 3: Heigth: 13, Leaves: 18

En términos visuales, el árbol podría parecerse más a una lista enlazada, donde muchos de los nodos están en una cadena, y solo algunos de ellos se ramifican en hojas. Esto puede resultar en un mayor número de niveles en comparación con la cantidad de hojas.

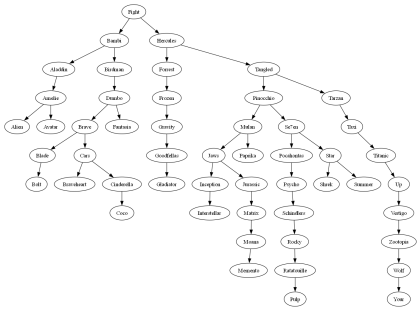

Figure 4: Heigth: 11, Leaves: 15

Cabe destacar que al tener una mayor altura, la inserción y eliminación de nodos también pueden ser más costosas en términos de tiempo, ya que se puede requerir un recorrido más largo para encontrar el lugar correcto para realizar estas operaciones. Además la altura del árbol implica que en el peor de los casos, para buscar un elemento se tendria que recorrer hasta 10 nodos, lo cual no es eficiente.

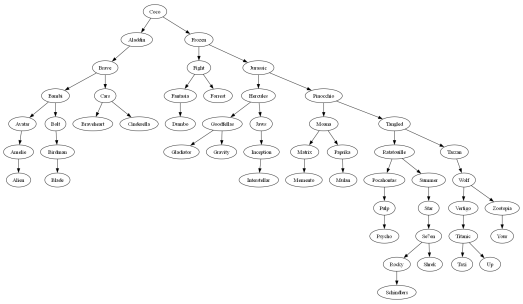

Figure 5: Heigth: 11, Leaves: 18

La diferencia en tamaño del hijo derecho del root puede sugerir que el árbol está desbalanceado. En un árbol bien balanceado los subárboles izquierdo y derecho deberían tener una cantidad razonablemente similar de nodos. Un árbol que está desbalanceado puede tener un rendimiento similar al de una lista enlazada en el peor de los casos, donde las operaciones se vuelven lineales en lugar de logarítmicas.

4 Conclusión

El análisis de este proyecto ha revelado aspectos críticos sobre su estructura y rendimiento. En particular, se observó que un árbol profundo con un número relativamente mayor de hojas con respecto a la altura puede ser un indicador de ineficiencia. La relación entre la altura del árbol y el número de hojas debe ser equilibrada para maximizar la eficiencia en las operaciones de búsqueda, inserción y eliminación.

Esta situación podría resultar en un rendimiento subóptimo, acercándose así a un comportamiento similar al de una lista enlazada, donde las operaciones de búsqueda podrían volverse lineales, incrementando el tiempo requerido para acceder a los elementos.

Dado que un árbol bien balanceado debería tener subárboles izquierdo y derecho de tamaños relativamente similares, es recomendable investigar

técnicas de balanceo. Estas técnicas pueden mejorar la eficiencia del BST y asegurar que las operaciones se realicen en tiempos logarítmicos en lugar de lineales, optimizando así el rendimiento general del árbol.