Murillo García Sebastián 191838

Altura esperada del subárbol del antecesor común de dos elementos de un Árbol Binario de Búsqueda.

En la tarea número dos, a saber: *Altura esperada de un Árbol Binario de Búsqueda*, se resolvió la pregunta relacionada con la altura de la estructura en los tres diferentes casos: para el **mejor** caso se tuvo un valor muy cercano a $\mathbf{log_2(n)}$ e incluso en algunos puntos (cuando el número de elementos en el árbol es de 2^n - 1) toca a esa función; para el **peor** caso el valor de la altura de un árbol con **n** datos es exactamente **n** y este resultado no varía; por último, para el caso **promedio** no se obtuvo un valor concreto sobre la altura, pero se hizo un pequeño análisis de los resultados y se concluyó que la altura en el caso promedio no se alejaba abismalmente de los valores del mejor caso y que la estructura sigue siendo conveniente.

Ahora bien, ¿qué sucede con la altura de los subárboles de la estructura? Específicamente ¿cuál es la altura esperada del antecesor común más cercano de dos elementos que están en la estructura? Si pensamos un poco y sin algún tipo de gráficas o tablas podemos decir que la altura invariablemente va a ser menor o igual porque el antecesor común se puede encontrar dentro de los niveles de la estructura o en la raíz misma. Esta es una hipótesis que se piensa resolver en este documento.

Una vez que en la hipótesis hacemos una aseveración relacionada con los resultados obtenidos en la tarea pasada, en las gráficas se van a implementar aquellas curvas que hacen referencia a estos resultados con el nombre de "***referencia***". Esta curva de *referencia* va a corresponder al mismo caso del que se trata, pero con la altura del árbol completo; en otras palabras, si se está analizando la altura del antecesor común en el mejor de los casos, la curva *referencia* va a representar la altura total del árbol en el mejor de los casos.

Después de haber dicho esto, se debe mencionar que en este documento se expondrán los resultados obtenidos en tres casos considerados: el mejor, el peor y el caso promedio con diferentes cantidades de datos insertados en el árbol. De esta manera, se tendrá una idea más clara de la altura esperada del antecesor común de dos elementos en un BST en una notación asintótica. Para analizar los diferentes casos se exponen distintas gráficas y tablas.

Algunas cosas a considerar.

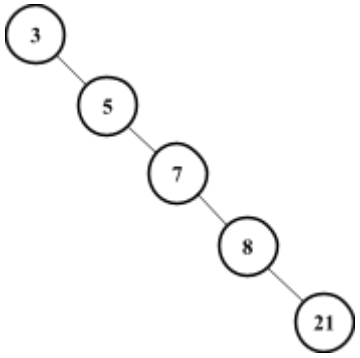
Los métodos que se programaron para resolver la cuestión antes mencionada fueron los siguientes:

I. Método **mitades**. Al igual que la tarea número 2, este método busca acomodar los elementos de un arreglo *arr* en un arreglo de salida *res* de tal manera que al insertar los datos en un árbol binario de búsqueda, este árbol esté relativamente balanceado. De esta manera se podrá tratar con árboles donde su altura esperada va a ser de $\mathbf{\log_2(n)}$. Recordemos el código:

```
private void mitades(T[] arr, int min, int max, ArrayStack<T> pila){
        int mit;

        if(min >= max)
            return;
        mit = (min + max)/2;
        pila.push(arr[mit]);
        mitades(arr, min, mit, pila);
        mitades(arr, mit+1, max, pila);
}

public void mitades(T[] arr, T[] res){
        ArrayStack<T> pila = new ArrayStack<T>();
        ArrayStack<T> aux = new ArrayStack<T>();
        int i;

        mitades(arr, 0, arr.length-1, pila);

        while(!pila.isEmpty())
            aux.push(pila.pop());

        i = 0;
        while(!aux.isEmpty()){
            res[i] = aux.pop();
            i++;
        }
}
```

II. Método **ancestor**. Este método encuentra al antecesor común de dos elementos en un árbol binario de búsqueda. Lo que se hace en las entrañas del código son comparaciones: si el nodo en el que nos encontramos (empezando por la raíz) es mayor que ambos elementos, eso quiere decir que ambos elementos se encuentran en su subárbol izquierdo; si fuera más chico que ambos, entonces los elementos se encuentran en el subárbol derecho dle nodo. Esto es por las propiedades de los BST. En cualquiera de ambos casos nos movemos para el nodo hijo que corresponda para continuar con las comparaciones ¿Cuándo encontramos al antecesor común? Cuando un elemento es más chico que el elemento del nodo en el que nos encontramos y el otro más grande.

También puede darse el caso de que el antecesor común sea uno de los elementos dados como parámetro y aquí entramos con un conflicto: ¿realmente ese es el antecesor? Estrictamente tendríamos que pensar que un nodo no puede ser su propio antecesor, entonces tenemos que tomar medidas en este caso: si el nodo no es la raíz, se regresa al papá del nodo obtenido como resultado (de esta manera para un elemento va a ser el antecesor directo y para el otro va a ser como su abuelo); si el nodo es la raíz, simplemente se regresa la raíz. Veamos un ejemplo rápido:



Si se buscara al antecesor común más cercano de los elementos **21 y 7**, el método recursivo **antecesorAux** regresaría al 7, pero esto es un poco contraintuitivo. Por lo tanto, se determinó que el antecesor real en un caso como este va a ser el papá del nodo que regresa el método auxiliar. Aquí está el código:

```
private BinaryNode<T> ancestorAux(BinaryNode<T> actual, T e1, T e2){
        T act;

        if(actual == null)
            return null;

        act = actual.getElem();
        if(act.compareTo(e1) > 0 && act.compareTo(e2) > 0)
            return ancestorAux(actual.getIzq(), e1, e2);

        if(act.compareTo(e1) < 0 && act.compareTo(e2) < 0)
            return ancestorAux(actual.getDer(), e1, e2);

        return actual;
}
public BinaryNode<T> ancestor(T e1, T e2){
        BinaryNode<T> ancestor;
        T anc;

        ancestor = ancestorAux(raiz, e1, e2);
        anc = ancestor.getElem();
        if(anc.equals(e1) || anc.equals(e2))
            if(ancestor.equals(raiz))
                return ancestor;
            else
                return ancestor.getPa();
        return ancestor;
}
```

III. Método de **niveles** con un nodo como parámetro. El método recursivo programado en clase se modificó solo un poco para que se pudiera llamar con un nodo específico como parámetro y que no empezara completamente por la raíz. Esto se hizo debido a que en el diseño del código para los resultados primero se obtiene al antecesor común más cercano y luego al método que saca la altura para determinar los niveles de ese árbol. Aquí está el código:

```
private int lvlsAux(BinaryNode<T> actual){
        int val, izq, der;

        if(actual == null)
            return 0;
        izq = lvlsAux(actual.getIzq());
        der = lvlsAux(actual.getDer());
        val = Math.max(izq, der);
        return 1 + val;
}
// Empieza desde la raíz.
public int niveles(){
        return lvlsAux(raiz);
}
// Empieza desde un nodo dado.
public int niveles(BinaryNode<T> nodo){
        return lvlsAux(nodo);
}
```

IV. Método **selector**. Este código elige aleatoriamente (bueno, pseudo-aleatoriamente) dos diferentes elementos de un arreglo *arr* dado como parámetro y se insertan en las primeras dos posiciones de un arreglo resultante *res*. El método se programó debido a que se utilizó el mismo generador de elementos aleatorios dada una cantidad n de datos deseados utilizado en la tarea número 2. Una vez que no sabemos con exactitud qué elementos están y sería muy tardado buscar cuáles datos sí se van a insertar en la estructura y cuáles no, entonces este método elige dos elementos del arreglo que se va a insertar en la estructura para después llamar al método **ancestor** y que nunca regrese un nodo nulo. Aquí está el código:

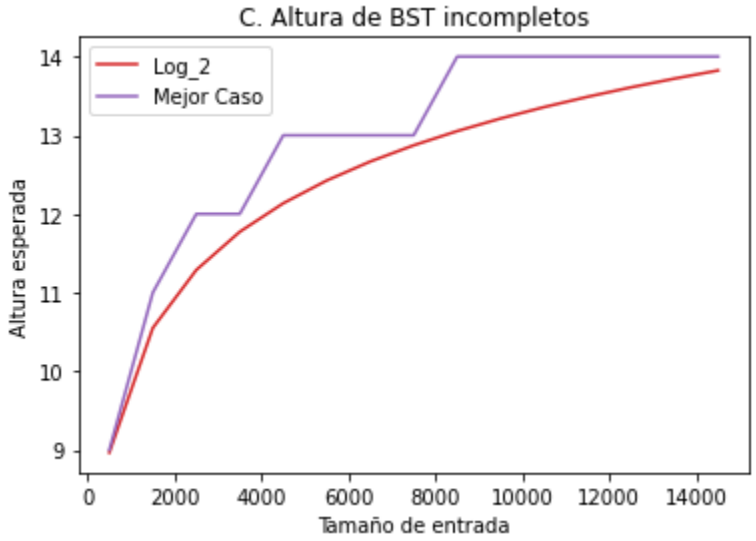
```
public void selector(T[] arr, T[] res){
        Random rand = new Random();
        int v1, v2;

        v1 = rand.nextInt(arr.length);
        v2 = rand.nextInt(arr.length);
        while(v1 == v2)
            v2 = rand.nextInt(arr.length);
        res[0] = arr[v1];
        res[1] = arr[v2];
    }
```

Ahora sí, analicemos los resultados obtenidos.

El mejor de los casos.

Para considerar este caso se utilizó el método **mitades** para tener árboles relativamente balanceados en la estructura. Como se dijo anteriormente, los resultados de la tarea número dos arrojaron que dado un árbol binario de búsqueda en su mejor caso con **n** elementos en su interior, la altura esperada va a tocar en algunos puntos (cuando el árbol está lleno) a la función de $\mathbf{log_2(n)}$ y va a estar cerca de estos valores cuando el árbol esté incompleto. Recordemos la gráfica obtenida:

Figura C de Tarea 2: Altura de BST incompletos.



Recordemos la hipótesis pensada antes de mostrar los resultados, a saber: la altura esperada del antecesor común de dos elementos en un árbol binario de búsqueda invariablemente va a ser menor o igual que la altura esperada total de la estructura. Una vez que los valores en el mejor de los casos son muy cercanos a $\mathbf{log_2(n)}$, entonces la altura esperada que responde a la pregunta esencial de este documento va a estar muy cercana a $\mathbf{log_2(n)}$. Veamos si esto es cierto.

Figura A : Altura del antecesor común en el mejor caso.

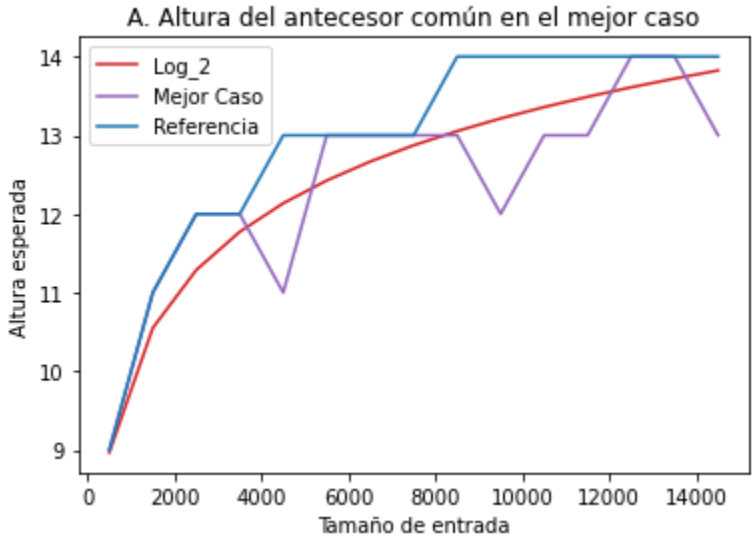
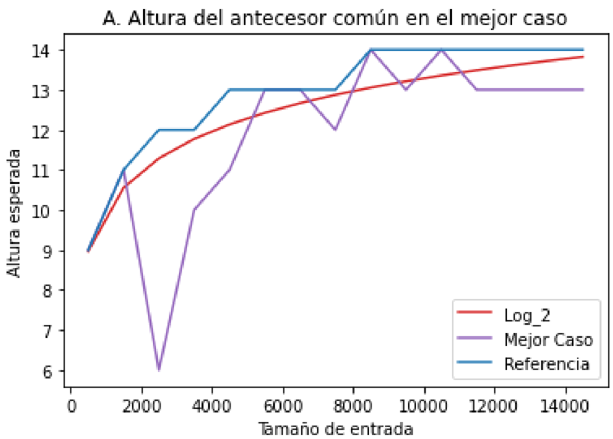


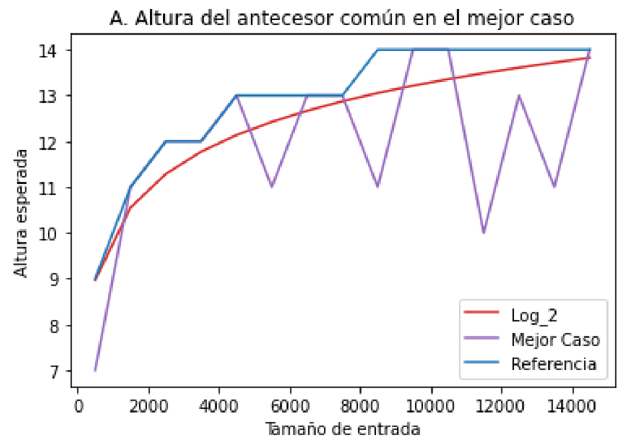
Tabla A : Valores numéricos de la Figura A.

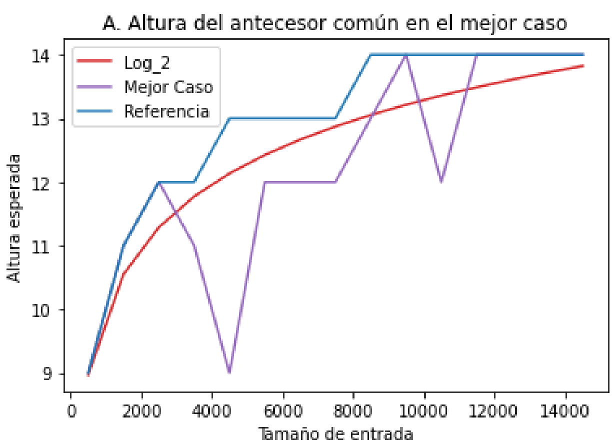
Cantidad de elementos en el árbol (n)														
500	1500	2500	3500	4500	5500	6500	7500	8500	9500	10.5 k	11.5 k	12.5 k	13.5 k	14.5 k
Valores de $\log_2(n)$														
8.9	10.5	11.2	11.7	12.1	12.4	12.6	12.8	13.0	13.2	13.3	13.4	13.6	13.7	13.8
Altura esperada TOTAL del árbol														
9	11	12	12	**13**	13	13	13	**14**	**14**	**14**	**14**	14	14	**14**
Altura esperada del antecesor común														
9	11	12	12	**11**	13	13	13	**13**	**12**	**13**	**13**	14	14	**13**

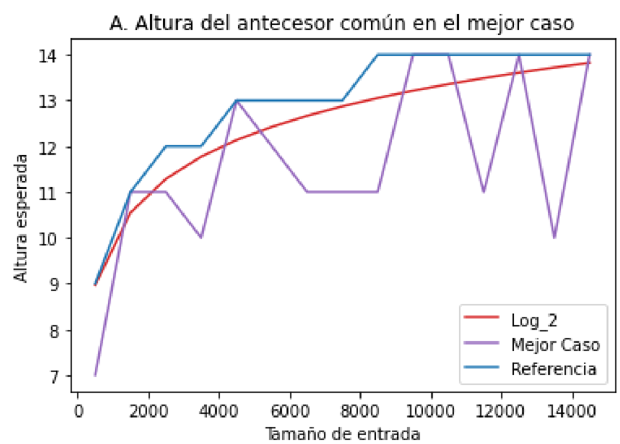
Como podemos observar tanto en la gráfica como en la tabla es que la altura esperada del antecesor común de dos elementos en un árbol binario de búsqueda en el **mejor caso** (y seguramente en los siguientes casos también) va a ser **menor o igual** a la altura esperada total del BST tratado. Por lo tanto, en este caso nuestra hipótesis se confirmó. Pero no nos vamos a quedar solamente con esta respuesta, vamos a dar una primera conclusión. ¿Cuándo es menor y cuándo es igual a la altura total? Esta respuesta es sencilla: la altura del antecesor común va a ser **igual** a la altura total cuando el antecesor sea la **raíz**. Por otro lado, la altura del antecesor va a ser **menor** a la altura total cuando el antecesor sea uno de los nodos intermedios de la estructura (puntos color **morado**). Parece ser entonces que los valores de la altura esperada del antecesor tiene como **cota superior** los valores del caso a tratar en la altura total y esto se cumple para los tres casos. ¿Existe una cota inferior para los tres casos? Conforme se avance en el documento se van a dar a conocer otras conclusiones importantes.

Aquí tenemos otras gráficas que representan la altura esperada del antecesor común de dos elementos en un árbol binario de búsqueda en el mejor de los casos simplemente para visualizar en otros resultados lo antes mencionado.









El peor de los casos.

Recordemos que en el peor de los casos, un árbol binario de búsqueda es **tan ineficiente** como una lista. Al igual que la tarea número dos, en este caso se consideraron dos situaciones para analizar los resultados: cuando los datos se insertan en un orden estrictamente de menor a mayor, o de mayor a menor. Por los resultados obtenidos en dicha tarea, ahora sabemos que dada una cantidad **n** de datos insertados en un árbol binario de búsqueda en su peor caso la altura total esperada del árbol es de **n**. En este caso tal vez no es tan necesario ver la gráfica puesto a que es la función **f(x) = y**. También, para fines de este escrito, se ocupará el caso en el que los datos estén ordenados de mayor a menor, pues ambos casos generan los mismos resultados.

Figura B : Altura del antecesor común en el peor caso.

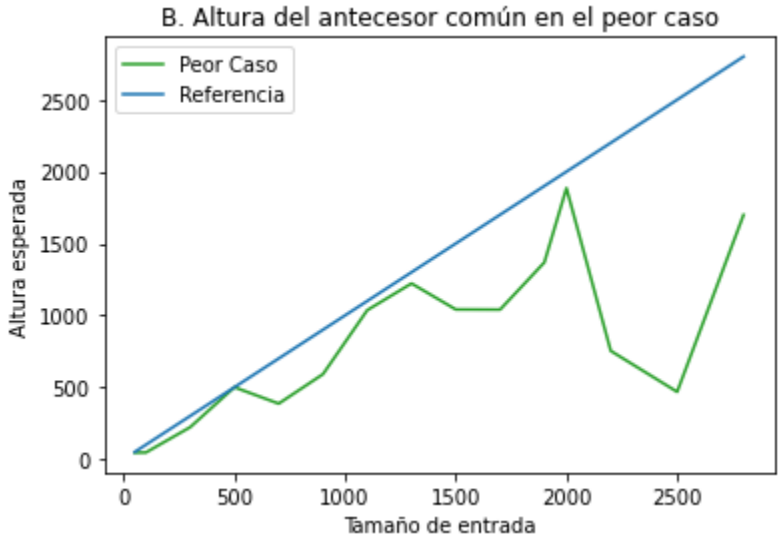


Tabla B : Valores numéricos de la Figura B.

Cantidad de elementos en el árbol (n)														
50	100	300	500	700	900	1100	1300	1500	1700	1900	2000	2200	2500	2800
Altura esperada TOTAL del árbol														
50	100	300	**500**	700	900	1100	1300	1500	1700	1900	2000	2200	2500	2800
Altura esperada del antecesor común														
45	46	222	**500**	386	591	1036	1222	1041	1040	1368	1885	753	468	1699

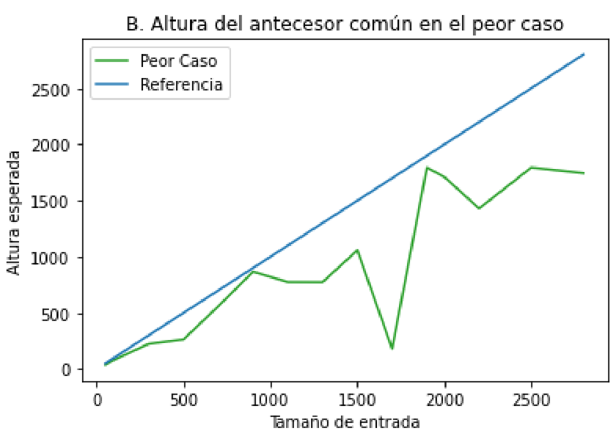
Podemos ver nuevamente que la hipótesis también se cumplió para el peor de los casos: la altura esperada del antecesor común de dos elementos en un BST es menor o igual que la altura total del mismo árbol. A pesar de tener la misma respuesta, podemos ver que en este caso **parece** ser que es más **difícil** que la altura esperada del antecesor sea **igual** a la altura total. ¿A qué se debe esto? Podemos responder que se debe precisamente a la estructura del árbol como tal.

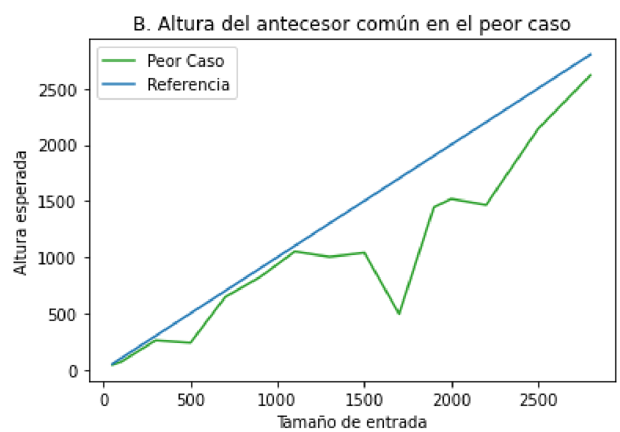
Ya que el árbol en este caso parece una lista, el valor va a ser **igual** cuando uno de los elementos de los que se busca sea la raíz o sea el primer hijo de la raíz (el único punto en la tabla de color **verde**) y esto difiere con el mejor caso. Para entender esto hay que recordar el ejemplo que se dio en la explicación del método *ancestor*. En el mejor caso para cada nodo se tienen tanto elementos en su subárbol izquierdo como otros en su subárbol derecho (incluso más o menos se tiene la misma cantidad de ambos lados); entonces al buscar al antecesor común entre dos elementos puede ser más probable encontrarnos con la raíz por el simple hecho de que si se tiene que un elemento dado como parámetro se encuentra en cualquier nivel del subárbol izquierdo de la raíz y el otro en el subárbol derecho,

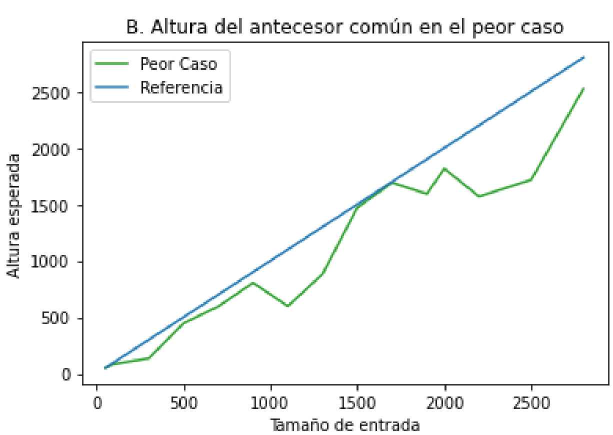
inmediatamente la raíz va a ser el antecesor común. En este peor caso solamente se tienen subárboles derechos (si los datos están, ordenados de menor a mayor) o subárboles izquierdos (si están ordenados de mayor a menor); por lo tanto, la única manera de que ambas alturas sean iguales es cuando se tiene que alguno de los elementos dados como parámetro sean la raíz o el primer hijo.

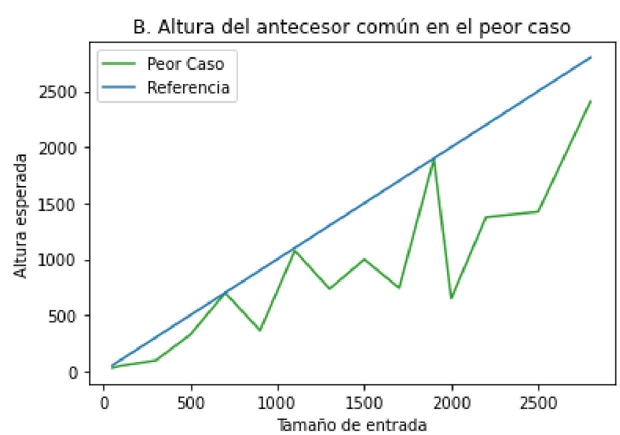
Además tenemos una <u>segunda conclusión</u>: la **diferencia** de las alturas entre la altura esperada del antecesor de dos elementos con respecto a la altura total del árbol va a ser cada vez mayor mientras que los elementos que se buscan en el arreglo están más alejados de la raíz dentro del mismo subárbol (ya sea izquierdo o derecho); además de que la altura esperada del antecesor va a ser más chica. En otras palabras, la diferencia va a ser mayor si por ejemplo se tiene que los dos elementos se encuentran en el subárbol derecho más lejano (y “recursivo”) de la raíz. Esto tiene sentido porque entre más abajo (en niveles) nos encontremos en el árbol, la distancia entre el nivel y las hojas que se encuentran en el último va a ser más chica a que si nos encontramos en la punta del árbol.

Al igual que en el caso pasado también presentamos otras gráficas del peor de los casos para visualizar los contrastes.









El caso promedio.

Para la altura esperada total del caso promedio se tuvo que realmente no se obtuvo un valor concreto, pero que la estructura sigue siendo conveniente pues los valores de la altura en el caso promedio no se alejan abismalmente de los valores en el mejor caso. Al igual que la tarea número dos, se implementó el generador de un arreglo con datos aleatorios para la inserción en la estructura. No podemos decir mucho de este caso, pues todo se aclaró en la tarea anterior, así que pasemos al análisis del caso promedio de la altura del antecesor común de dos elementos de un BST.

Figura C : Altura de BST en su caso promedio.

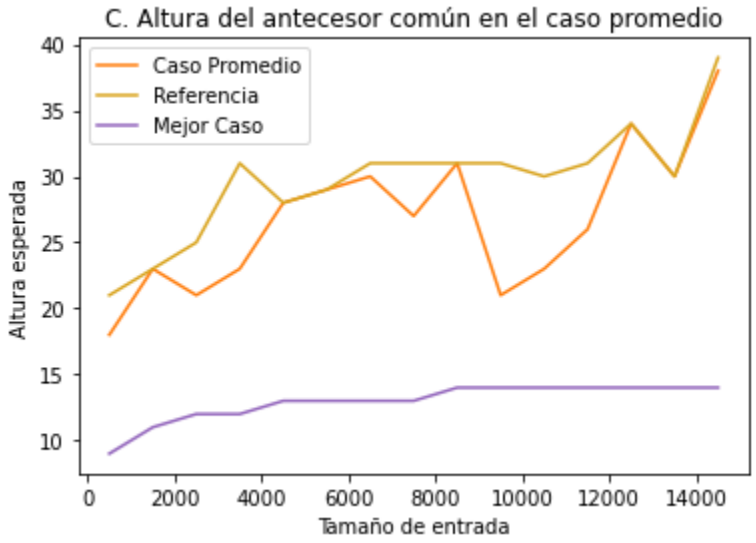
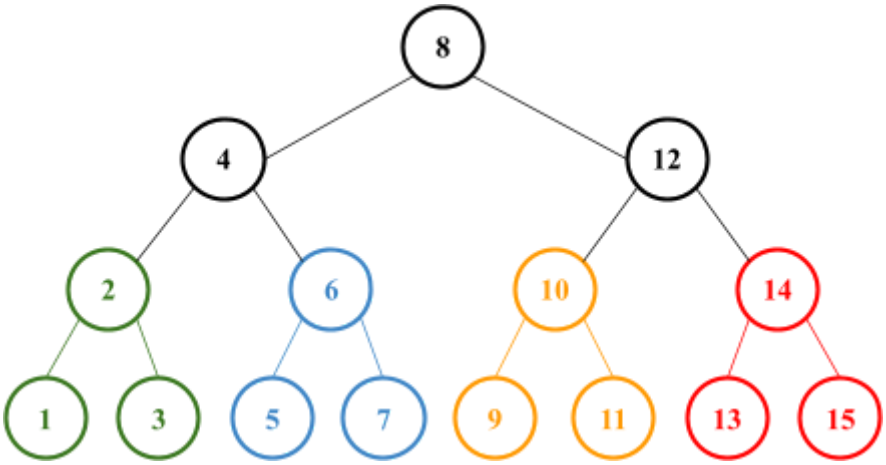


Tabla C : Valores numéricos de la Figura C.

Cantidad de elementos en el árbol (n)														
500	1500	2500	3500	4500	5500	6500	7500	8500	9500	10.5 k	11.5 k	12.5 k	13.5 k	14.5 k
Altura esperada del árbol en el mejor de los casos														
9	11	12	12	13	13	13	13	14	14	14	14	14	14	14
Altura esperada del árbol TOTAL en el caso promedio														
21	23	**25**	**31**	28	29	**31**	**31**	31	**31**	**30**	**31**	34	30	**39**
Altura esperada del antecesor común en el caso promedio														
18	23	**21**	**23**	28	29	**29**	**27**	31	**21**	**23**	**26**	34	30	**38**

Como podemos ver, también se cumple la hipótesis antes mencionada. Esto cierra por completo el análisis de los tres casos y podemos afirmar que la hipótesis es completamente verdadera. Pero ¿existe algo particular de este caso? Sinceramente no. Desde un punto de vista sí cumple con la primera conclusión dada en el **mejor** caso y también cumple con la segunda conclusión dada en el **peor** caso. Además es cierto que la cota superior sigue siendo la curva de la altura total del árbol tratado y que mientras más abajo (en niveles más inferiores) se encuentre el antecesor en el árbol, más chica va a ser la altura esperada de este nodo y más grande será la diferencia de las alturas (la esperada del antecesor con la total). Por lo tanto, no hay más que una reiteración de lo antes mencionado.

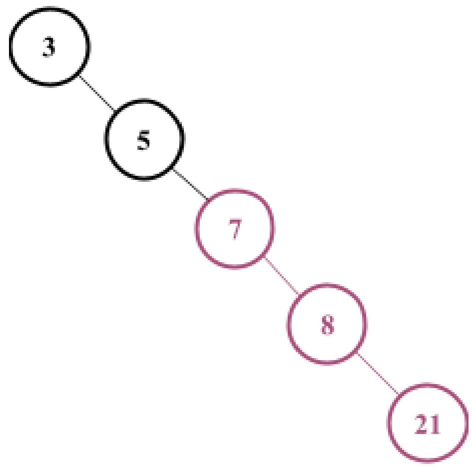
Pero no vamos a terminar el documento de esta manera. Es aquí cuando llegamos a mencionar la tercera conclusión: la **cota inferior**. ¿Existe? Claro que existe. Si mencionamos en la segunda conclusión que la altura esperada del antecesor va a ser cada vez más chica mientras el antecesor se encuentre en los niveles más inferiores del árbol (y dentro de la "misma rama recursiva" [el subárbol más más más izquierdo o derecho]), entonces para encontrar la cota inferior hay que ver los niveles más inferiores. Aquí un ejemplo:



Podemos observar claramente que cualquiera de los nodos con colores diferentes son una de las condiciones "de esquina" y cumple con lo dicho anteriormente en el sentido de que sí se dan dos elementos para buscar un antecesor común (pensemos en los elementos 13 y 15 como parámetros y 14 como su antecesor). Además, nos encontramos en los niveles más inferiores de la estructura (los nodos del 13 y 15 no pueden llamar a un método de antecesor común, por lo que este es uno de los límites). Por último, estamos en los subárboles "más recursivos". Entonces, en este caso, ¿cuál es la altura esperada del antecesor común? La respuesta es sencilla: 2. Por lo tanto, la **cota inferior** de los casos **mejor y promedio** va a ser **2** si se toman como parámetros las hojas más inferiores de la estructura.

¿Y para el **peor caso**? La respuesta es **3** debido a que como tratamos con listas, las condiciones de esquina van a ser cuando se busque al antecesor común de **la hoja** y del **papá** de la hoja.

Podemos ver aquí a qué nos referimos:



Por lo tanto, ya sea que nos encontremos con un árbol binario de búsqueda con datos que fueron insertados de mayor a menor o con un BST con datos insertados de menor a mayor, la **cota inferior** siempre va a ser de **3**.

Por último (y como en los casos anteriores) se presentan otras gráficas que representan la altura esperada del antecesor común de dos elementos en un árbol binario de búsqueda en su caso promedio.