Universidad Nacional Autónoma de México
Facultad de Ciencias
Estructuras de Datos 2022-2
Práctica 3

Pedro Ulises Cervantes González
confundeme@ciencias.unam.mx

Yessica Janeth Pablo Martínez,
yessica_j_pablo@ciencias.unam.mx

Jorge Macías Gómez
jorgemacias@ciencias.unam.mx

19 de Abril del 2022

El código debe incluir comentarios y debe ser legible. Sigue las buenas prácticas al programar.
Únicamente envía los archivos con terminación .java.
No se reciben entregas fuera de Classroom.
El día limite es el martes 26 de Abril a la hora indicada en Classroom.

Actividades

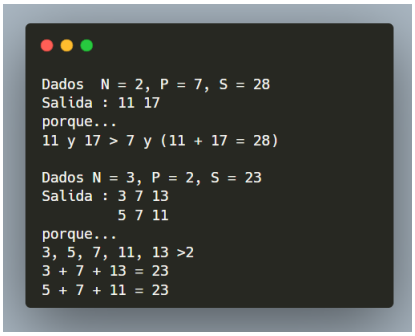
1. **En la clase Practica3**.

   - Completa el método. sumaCercana.
     Este método recibirá una lista de enteros y un entero n, deberá encontrar la pareja de números en la lista tal que la suma de estos, sea la mas cercana a n.
     Deberás hacerlo en una complejidad de tiempo de $O(nlogn)$
   - Completa el método permutacionesCadena.
     Dada una cadena. Debes imprimir todas las permutaciones de esta.
     Por ejemplo, para la cadena ABC las permutaciones son:

     - ABC
     - ACB
     - BAC
     - BCA
     - CBA
     - CAB

     Debes utilizar Backtracking.
   - Completa el método primosQueSuman.

     Dados 3 números, la suma S, el primo P, y un entero N, encuentra N primos mayores que P, tal que su suma sea igual a S.
     Ejemplos.



Debes utilizar backtracking y debes calcular los primos necesarios con la criba de Eratostenes.

- Completa el método N_Reinas.
  El problema de las N reinas consiste en colocar N reinas del juego de ajedrez en un tablero de ajedrez de NxN tal que no exista una pareja de reinas que se ataquen mutuamente. Para este ejercicio, puedes usar nuestra clase lista para simular una matriz que represente al tablero, o los arrays de Java.
  Debes utilizar BackTracking para la resolucion del problema.
- ¿Se puede hacer búsqueda binaria en casos no discretos? Un ejemplo de esto es que podemos encontrar el valor de $\sqrt{x}$ usando estas ideas, ya que se cumplen las propiedades para poder hacer este tipo de búsqueda:
  - Si $a^2 < x$, entonces podemos restringir nuestra búsqueda a números mayores que a.
  - Si $a^2 > x$, podemos restringir la búsqueda a números mayores que a.
  - Si $a^2 = x$, hemos terminado.

  Debemos considerar dos cosas. La primera es que no podemos ser tan optimistas de encontrar el valor exacto de $\sqrt{x}$, por lo que debemos definir un margen de error $1e-5 = 0{,}00001$.
  La segunda es que primero debemos definir nuestro intervalo de búsqueda. ¿Es $[0, x]$ un buen intervalo para empezar nuestra búsqueda? Tu misión. Sera escribir un método sqrtBusqBin que aproveche las cualidades de búsqueda binaria para aproximarse al resultado.

2. clase ArbolBinarioBusqueda.java
   Un árbol binario de búsqueda o BST por sus siglas en ingles es un tipo particular de arbol binario. Veamos la definición de un BST

   Sea A un árbol binario de raíz R e hijos izquierdo y derecho (posiblemente nulos) HI y HD , respectivamente.

   Decimos que A es un árbol binario de búsqueda (BST) si y solo si cumple que:

   El subárbol izquierdo de un nodo contiene solo nodos con elementos menores que el elemento del nodo. El subárbol derecho de un nodo contiene solo nodos con elementos mayores que el elemento del nodo. El subárbol izquierdo y derecho también deben ser un árbol de búsqueda binaria. No debe haber nodos duplicados.

   Tu misión sera crear este árbol. Con los siguientes métodos.

   - buildUnsorted(lista): Construye el BST a partir de una lista no ordenada. $O(nlogn)$
   - buildSorted(lista): Construye el BST a partir de una LISTA ordenada.$O(n)$
     OJO... con la complejidad de encontrar el i-esimo elemento de una lista
   - convertBST(arbolBinario): Construye el BST balanceado a partir de un arbolBinario.
   - search(root, elem): Función que nos permite preguntar por la existencia del elemento en el BST.
   - insert(root, elem): Función que nos permite insertar un elemento en el BST.
   - delete(root,elem): Función que nos permite eliminar un elemento en el BST. Si el nodo tiene un hijo, se sube el hijo al lugar del padre. si el nodo tiene más de un hijo, deberás encontrar al sucesor inOrder y reemplazarlo en el árbol. El cual sera el mínimo de el subárbol derecho del nodo a eliminar.

- toString(root): Método que imprime el árbol con in-Order DFS
- balance(root): Método que balancea el árbol BST. $O(n)$. Hint: Quita la limitante del buildSorted para hacerlo en esta complejidad. ¿Cual es la limitante?

¡Importante!

La intención de la tarea es que desarrolles tus habilidad de programación. Si tienes dudas de como escribir tus ideas en código no dudes en mandar un mensaje de Telegram al ayudante de laboratorio.

@Toaultor