Estructuras de Datos

Memoria PL2 2021-2022

Marco González Martínez 09064995M

Detalles y justificación de la implementación

Especificación de los TADs implementados, definición de las operaciones

espec ARBOL[PACIENTES]

usa entero

parámetro

genero pacientes

genero cola

operaciones

Arbol: 🡪 arbol {genera un árbol vacío}

getTamanno: arbol 🡪 int {Calcula el tamaño del árbol, la cantidad de nodos}

contarHabImpares: arbol 🡪 int {Cuenta la cantidad de nodos que contienen un paciente que reside en una habitación impar}

altura: arbol 🡪 int {Calcula la altura del árbol}

subArbolIzq: arbol 🡪 arbol {Devuelve el subárbol Izquierdo del árbol}

subArbolDer: arbol 🡪 arbol {Devuelve el subárbol derecho del árbol}

mínimo: arbol 🡪 arbol {Devuelve el subárbol cuyo paciente reside en la habitación de número mínimo}

máximo: arbol 🡪 arbol {Devuelve el subárbol cuyo paciente reside en la habitación de número máximo}

insertar: paciente, arbol 🡪 arbol {Inserta un paciente en el árbol}

eliminarPaciente: arbol, int 🡪 arbol {Elimina el nodo cuyo paciente resida en la habitación introducida (Si existe)}

buscarPaciente: arbol, int 🡪paciente {Devuelve el paciente que reside en la habitación introducida (Si existe)}

inorden Izq: arbol 🡪 {Muestra el inorden del subárbol izquierdo}

inordenDer: arbol 🡪 {Muestra el inorden del subárbol derecho}

inorden: arbol 🡪 {Muestra el inorden del árbol}

preorden: arbol 🡪 {Muestra el preorden del árbol}

postorden: arbol 🡪 {Muestra el postorden del árbol}

mostrarHojas: arbol 🡪 {Muestra todos los nodos hojas del árbol}

dibujar: arbol 🡪 {Dibuja un grafo que representa el árbol}

finoperaciones

finespec

A la hora de incluir el TAD en el código, se utiliza también la clase NodoArbol, el cual tiene los siguientes métodos:

hayHabImpares() 🡪 bool {Calcula si hay habitaciones impares en algún subárbol del nodo}

esHoja() 🡪 bool {Calcula si el nodo se trata de un nodo hoja}

getPaciente() 🡪 puntero a paciente {Devuelve un puntero al paciente contenido por el nodo}

Dificultades encontradas: soluciones adoptadas

He encontrado una dificultad a la hora de crear pacientes con IDs y Números de Habitación aleatorios para resolver el problema he utilizado el objeto vector para almacenar en 4 vectores de enteros distintos los IDs que había libres en cada momento para pacientes con cada enfermedad, y de igual forma con las habitaciones en las que pueden residir los pacientes.

Con el uso de vectores, al asignarle un ID y una habitación a un paciente, se extrae un número de las listas correspondientes a su enfermedad, y sólo se volverá a insertar en el caso de que un paciente reciba el alta médica. Tanto al inicio de la ejecución como en el momento de eliminar un paciente del sistema, la lista se reordena aleatoriamente, para garantizar que la elección del ID y la habitación sea aleatoria.

También otra dificultad fue la de eliminar un Paciente de un árbol, para lograr superar el problema, utilicé los siguientes tres métodos dentro de la clase Arbol.  
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Con estos métodos (el primero público y los otros dos privados) a partir de un número de habitación dado, conseguía eliminar el paciente que residiese en esa habitación. El funcionamiento del código es el siguiente: el método eliminarPaciente() comprueba si el valor de la habitación introducida debe de estar en su subárbol izquierdo, el derecho o en su mismo nodo. Esto se logra gracias a la peculiaridad de los árboles binarios que ordenan en su subárbol izquierdo los elementos menores o iguales al mismo nodo, y en el derecho los mayores. En caso de que se deba eliminar el nodo que está siendo evaluado, el método llama al método de eliminarNodo() a partir de ese nodo para devolver el nodo adecuado, si se encuentra en algún subárbol llama al método eliminarPaciente() a partir del subárbol correspondiente, si alguno de estos fuese nulo, devolvería el nodo sin alterarlo.

El método eliminarNodo() devuelve el nodo alterando el contenido del nodo, si el nodo introducido se trata de un nodo nulo, o un nodo hoja, devuelve su nodo izquierdo (para devolver siempre un nodo nulo) en el caso de que no tenga nodo izquierdo devuelve el nodo derecho ya que no es posible encontrar un valor apropiado para reemplazar el nodo alterado,   
y en caso contrario, colocaría el paciente máximo del subárbol izquierdo y lo eliminaría.