EDA-2022-ENE.pdf

Anónimo

Estructura de Datos y Algoritmos

2º Grado en Desarrollo de Videojuegos

Facultad de Informática
Universidad Complutense de Madrid

Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022
Examen final. Convocatoria ordinaria **Tiempo: 3 horas**

Instrucciones

- La entrega se realiza en el juez automático de los laboratorios accesible desde la url `http://exacrc` (cada ejercicio en su correspondiente problema del juez, acabados respectivamente en Ej1, Ej2 y Ej3). Para acceder debes usar el usuario/contraseña que has recibido al comienzo del examen.
- Al principio de cada fichero .cpp debe aparecer, en un comentario, vuestro nombre y apellidos, dni y puesto de laboratorio. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, respuestas a preguntas, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- Los TADs, las plantillas y ficheros de entradas de ejemplo para cada ejercicio se descargan desde `http://exacrc/EDA-Enero22.zip`.
- Podéis realizar varias entregas para un mismo ejercicio pero solamente se tendrá en cuenta la última.
- Podéis acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 1 [3 puntos]

Implementa una función (**externa** a la clase) que adelante un número de posiciones dado a un segmento (elementos en posiciones consecutivas) de la lista pasada como argumento (del tipo `list` visto en clase, fichero `list_eda.h`). Por ejemplo, si la lista está formada por los elementos *a b c d e f g*, y adelantamos $k = 3$ posiciones el segmento que comienza en la posición $pos = 5$ (las posiciones se numeran desde 0 hasta n-1, siendo n el número de elementos de la lista) y tiene longitud lon=2, entonces la lista resultante es *a b* ***f g*** *c d e* (donde se han marcado en negrita los elementos desplazados). Si la posición de origen o destino del segmento no es válida (lo que incluye el caso de la lista vacía), o si $lon = 0$ o $k = 0$, la operación no tendrá efecto (ver cuarto caso). Si el segmento se sale de la lista, es decir, si $pos + lon > n$ se tomará el segmento de los últimos $n - pos$ elementos (ver segundo caso). Se puede asumir que los 4 parámetros n, pos, lon y k son no-negativos.

Se valorará la eficiencia y complejidad tanto en tiempo como en espacio del algoritmo implementado, las cuales debes indicar y justificar. En particular, se penalizarán bastante el uso de espacio no constante y los recorridos innecesarios.
La función principal proporcionada para hacer pruebas comienza leyendo el número de casos. Cada uno consta de dos líneas. En la primera aparecen los números n, pos, lon y k, y en la segunda los n elementos de la lista. Una vez leidos e insertados en la lista se llama a la función pedida con los argumentos correspondientes y se muestra por pantalla la lista modificada (ver ejemplos).

Entrada	Salida
4	
7 5 2 3 a b c d e f g	a b f g c d e
7 5 4 1 x y z p r s t	x y z p s t r
7 2 4 2 x y z p r s t	z p r s x y t
7 2 2 3 a b c d e f g	a b c d e f g

Cuestión: ¿Se tendría alguna ventaja al implementar la operación como método interno a la clase `list`? Justifica la respuesta.

Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022

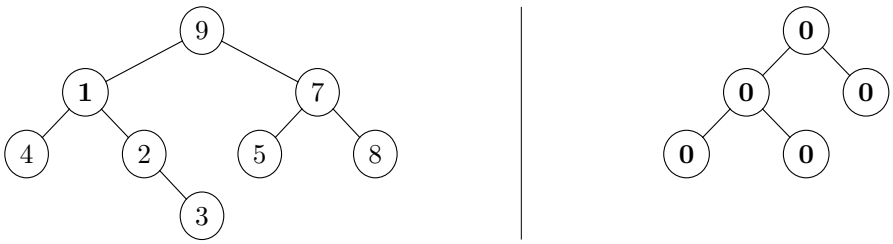
Examen final. Convocatoria ordinaria **Tiempo: 3 horas**

Instrucciones

- La entrega se realiza en el juez automático de los laboratorios accesible desde la url `http://exacrc` (cada ejercicio en su correspondiente problema del juez, acabados respectivamente en Ej1, Ej2 y Ej3). Para acceder debes usar el usuario/contraseña que has recibido al comienzo del examen.
- Al principio de cada fichero .cpp debe aparecer, en un comentario, vuestro nombre y apellidos, dni y puesto de laboratorio. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, respuestas a preguntas, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- Los TADs, las plantillas y ficheros de entradas de ejemplo para cada ejercicio se descargan desde `http://exacrc/EDA-Enero22.zip`.
- Podéis realizar varias entregas para un mismo ejercicio pero solamente se tendrá en cuenta la última.
- Podéis acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 2 [3 puntos]

Un nodo de un árbol binario de enteros (no negativos) se dice que es *singular* si la suma de los valores almacenados en sus nodos antepasados es igual a la suma de los valores almacenados en sus nodos descendientes. Por ejemplo, en el árbol de la izquierda el nodo 1 (en negrita) es el único nodo singular pues la suma de sus antepasados es 9 y la suma de sus descendientes también es 9. En el árbol de la derecha todos los nodos son singulares.



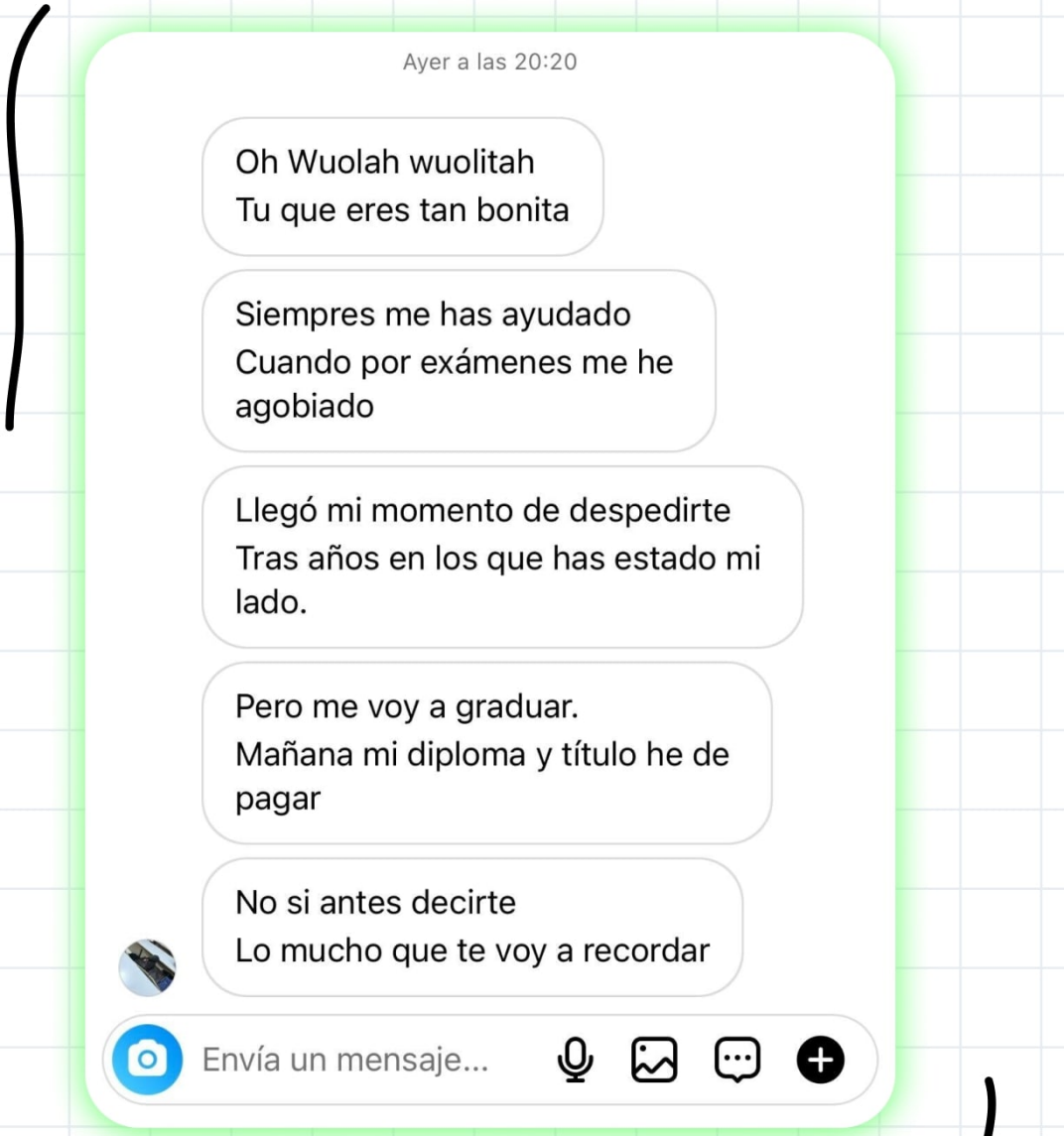
Implementa una función (externa al TAD `bintree`) que dado un árbol binario de enteros no negativos devuelva el número de nodos singulares que tiene. Indica y justifica la complejidad de la función implementada.

La función principal proporcionada para hacer pruebas comienza leyendo el número de casos. Cada caso viene en una línea que contiene el recorrido preorden del árbol en el que el valor -1 representa al árbol vacío. Para cada uno se imprime en una línea el número de nodos singulares que tiene.

Entrada	**Salida**
5	
9 1 4 -1 -1 2 -1 3 -1 -1 7 5 -1 -1 8 -1 -1	1
0 0 0 -1 -1 0 -1 -1 0 -1 -1	5
-1	0
2 -1 -1	1
4 5 1 -1 -1 3 -1 -1 2 -1 -1	1

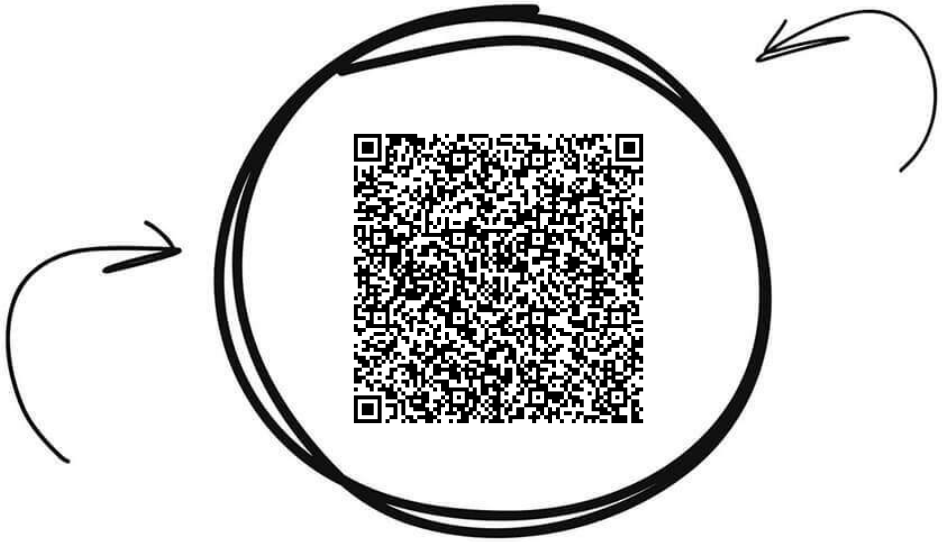
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Estructura de Datos y Algori...



Banco de apuntes de la UCM

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022
Examen final. Convocatoria ordinaria **Tiempo: 3 horas**

Instrucciones

- La entrega se realiza en el juez automático de los laboratorios accesible desde la url `http://exacrc` (cada ejercicio en su correspondiente problema del juez, acabados respectivamente en Ej1, Ej2 y Ej3). Para acceder debes usar el usuario/contraseña que has recibido al comienzo del examen.
- Al principio de cada fichero .cpp debe aparecer, en un comentario, vuestro nombre y apellidos, dni y puesto de laboratorio. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, respuestas a preguntas, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- Los TADs, las plantillas y ficheros de entradas de ejemplo para cada ejercicio se descargan desde `http://exacrc/EDA-Enero22.zip`.
- Podéis realizar varias entregas para un mismo ejercicio pero solamente se tendrá en cuenta la última.
- Podéis acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 3 [4 puntos]

Tenemos una lista de las películas emitidas durante el último año en una cadena de televisión. De cada película conocemos los actores que intervienen en ella y el tiempo que aparecen en pantalla durante la película. Queremos obtener la película y el actor preferidos por la cadena. La película preferida es aquella que más veces se ha emitido, mientras que el actor preferido es aquel que más minutos aparece en pantalla contando todas las emisiones. Si existen varios actores que han aparecido el mismo tiempo máximo mostraremos todos ellos por orden alfabético. Si existen varias películas que se han emitido el mayor número de veces, mostraremos la que se ha emitido más recientemente de todas ellas.

Se pide lo siguiente:

- Completa la implementación de la función `leerRepartos` de manera que se almacene toda la información de los repartos de todas las películas en el segundo parámetro (ver plantilla proporcionada). Define una estructura de datos adecuada para el tipo `RepartosPeliculas` usando los TADs vistos durante el curso (usa los de la STL). No es necesario crear una clase, basta con definir un alias mediante `using` o un `struct`. Indica y justifica brevemente la complejidad de la función.
- Implementa la función `procesarEmisiones` que calcula la información solicitada a partir de los repartos de las películas (tipo `RepartosPeliculas`) y la secuencia de emisiones (tipo `vector<string>`). Se valorarán de manera notable el uso adecuado de estructuras de datos auxiliares y la complejidad y eficiencia de la implementación, la cual debes indicar y justificar. Puedes mostrar la salida de texto desde la propia función.

Entrada La entrada consta de una serie de casos de prueba. Cada caso se muestra en varias líneas. En la primera se indica el número de películas distintas que se han emitido. A continuación se muestra, para cada película, su título y el número de actores que aparecen en ella. En la línea siguiente se muestra el nombre de cada actor y su tiempo de actuación (un número positivo) en esta película, separados por blancos. A continuación se muestra el número de películas emitidas por la cadena de televisión en todo el año, seguido de los títulos de las películas en el orden en que fueron emitidas. La entrada finaliza cuando haya un caso con 0 películas (que no se procesa). Los títulos de las películas y los nombres de los actores son cadenas de caracteres sin espacios en blanco.

Salida La salida de cada caso se escribirá en dos líneas. En la primera se muestra el máximo número de veces que se ha emitido una película seguido de su título. Si existen varias películas que se han emitido el mismo número máximo de veces, se muestra el título de la última emitida. En la segunda línea se muestra el máximo tiempo que ha aparecido un actor en pantalla seguido de los nombres de los actores que han aparecido ese tiempo máximo en orden alfabético.

Ejemplos de Entrada/Salida:

2 película1 3 actor1 10 actor2 30 actor3 5 película2 2 actor1 10 actor4 30 3 película2 película2 película1 4 película1 3 actor3 5 actor1 10 actor2 20 película3 1 actor5 40 película4 1 actor1 40 película2 2 actor1 10 actor4 15 5 película2 película1 película2 película3 película1 0	2 película2 60 actor4 2 película1 40 actor1 actor2 actor5