Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2020-2021

Examen final. Convocatoria ordinaria Tiempo: 2 horas y 30 minutos

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.
- Las plantillas, TADs y ficheros de entradas para cada ejercicio se obtienen pulsando en el icono del Escritorio "Publicacion docente ...", después en "Alumno recogida docente", y en el programa que se abre, abriendo en la parte derecha la carpeta `.../TODOS/EDA-GDV-Feb21`, arrastrando los ficheros a hlocal (en la izqda).

Ejercicio 1 [3.5 puntos]

Extiende el TAD `Cola` visto en clase (`Queue.h`) con una nueva operación interna y pública cuya cabecera en `C++` es

```
void cuela(const T& a, const T& b);
```

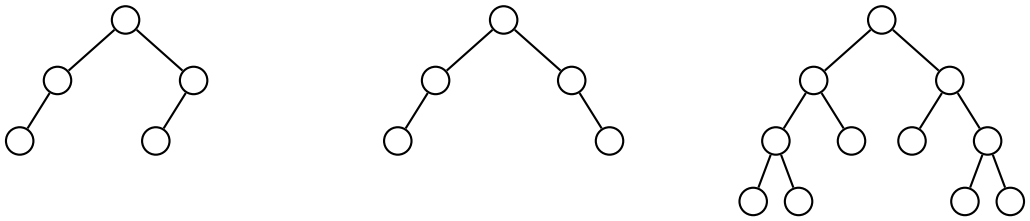
que mueve al elemento `b` de su posición a la posición inmediatamente detrás del elemento `a`. En caso de haber múltiples apariciones de los elementos `a` y/o `b` se considerará: la primera aparición de `a`, y la primera aparición de `b` tras la primera aparición de `a`. Si alguno de los elementos no se encuentra en la cola, o bien, si `b` no aparece detrás de `a`, la operación no tendrá efecto. Indica y justifica la complejidad de la operación implementada. *Requisitos:* No se puede crear ni destruir memoria dinámica, ni tampoco modificar los valores almacenados en la cola.

Entrada	**Salida**
1 2 3 4 -1 1 3	1 3 2 4
1 2 3 4 -1 1 4	1 4 2 3
1 2 3 4 -1 2 1	1 2 3 4
3 1 2 1 3 4 -1 1 3	3 1 3 2 1 4

La función principal proporcionada para hacer pruebas lee la cola de enteros (secuencia de enteros no negativos acabando la lectura con un -1 que no se incluye en la cola), después los enteros `a` y `b`, llama a la función pedida, y muestra por pantalla la cola resultante (ver ejemplos). El proceso se repite hasta introducir una cola vacía (es decir, un -1).

Ejercicio 2 [3 puntos]

Un árbol binario no vacío es *simétrico* respecto al eje vertical que pasa por la raíz si al "doblarlo" por ese eje todo nodo de un lado coincide con un nodo del otro. Por ejemplo, de los siguientes árboles, el de la izquierda no es simétrico pero los otros dos sí lo son. El árbol vacío también se entiende como simétrico.



Implementa una función (externa al TAD `Arbin`) que dado un árbol binario devuelva un booleano indicando si el árbol es o no simétrico. 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 carácter '*'representa un nodo y el '.'representa un árbol vacío.

Entrada	Salida
5	
***...**...	NO
***...*.*..	SI
****..*..*..**..**..*..	SI
*..	SI
..*..	NO

Ejercicio 3 [3.5 puntos]

Se va a celebrar un concierto benéfico de rock como el que tuvo lugar hace 36 años en Wembley. Los artistas participantes ya están confirmados y solamente falta decidir el orden de actuación de los mismos. Los promotores del concierto han realizado una estimación de la cantidad de donaciones que se pueden recibir durante la actuación de cada uno de los n artistas dependiendo del momento 0 a $n-1$ en el que actúan. También disponen de una tabla de "consentimientos" en la que cada artista ha reflejado si admite tocar o no inmediatamente después de cada uno de los demás. Por ejemplo *CoolPlay* no acepta tocar después de nadie mientras que *Mus* acepta tocar solamente después de *Willie Alish*. Ayuda a los promotores a determinar el orden en que han de tocar los artistas para obtener la máxima donación posible según la estimación realizada.

Implementa un algoritmo de vuelta atrás que resuelva el problema. Explica claramente cuál es la tupla solución y los marcadores que has utilizado. Implementa una poda por estimación y pon un ejemplo de aplicación.

La función principal proporcionada para hacer pruebas comienza leyendo el número de casos. Cada caso de prueba contendrá el valor del número de artistas n. A continuación figuran las estimaciones de las donaciones: una fila para cada artista. Después los consentimientos de los artistas: una fila para cada artista i indicando si admite (1) o no (0) tocar después del artista j (habrá un 0 en la posición i). Por cada caso de prueba el programa escribirá una línea con la donación máxima estimada (suma de las donaciones obtenidas por cada artista en el momento que le corresponde tocar). En caso de que no sea posible satisfacer los consentimientos se escribirá la cadena "NEGOCIA CON LOS ARTISTAS".

Entrada	Salida
2 3 10 20 30 140 20 10 160 10 20 0 1 1 0 0 1 0 0 0 3 10 20 30 140 20 10 160 10 20 0 0 1 0 0 1 0 0 0	210 NEGOCIA CON LOS ARTISTAS