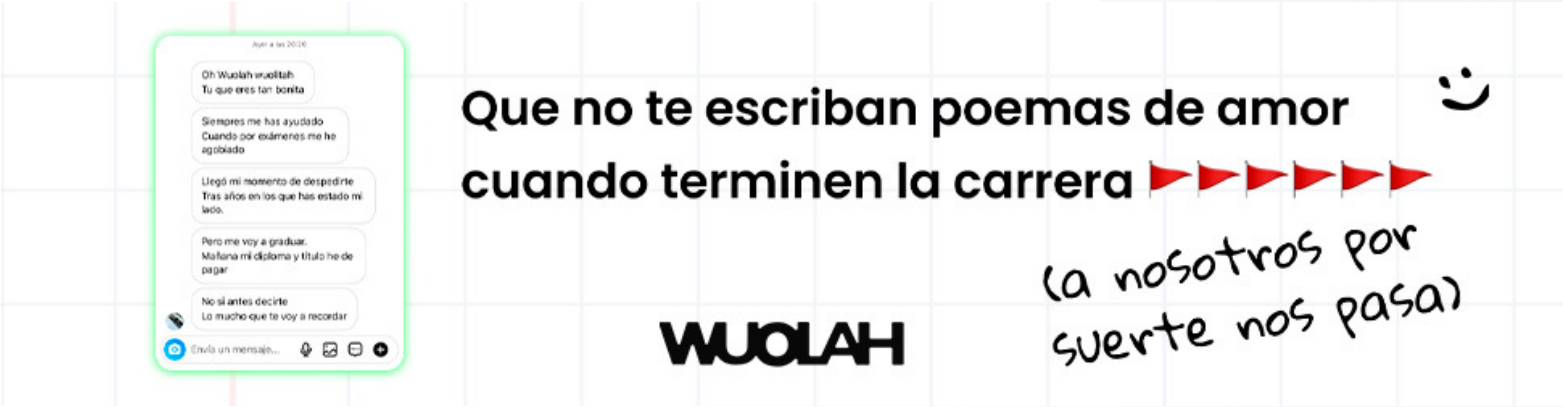
EDA-2021-JUN.pdf

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 2020-2021

Examen final. Convocatoria extraordinaria **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.
- Los TADs, las plantillas y ficheros de entradas de ejemplo para cada ejercicio se descargan desde `http://exacrc/EDA-Julio21.zip`.
- Podéis realizar varias entregas para un mismo ejercicio pero solamente se tendrá en cuenta la última.
- Puedes acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 1 [2.5 puntos]

Implementa una función (**externa** a la clase) que de la vuelta a un segmento (elementos en posiciones consecutivas) de la lista pasada como argumento (del tipo `list` de la STL). Por ejemplo, si la lista está formada por los elementos 1 2 3 4 5 6 7 8, e invertimos el segmento que comienza en la posición 3 (las posiciones se numeran desde 1 hasta N, el número de elementos de la lista) y tiene longitud 4, entonces la lista resultante es 1 2 *6 5 4 3* 7 8 (donde se han marcado en cursiva los elementos invertidos). Se puede asumir que la posición P de comienzo del segmento es válida ($1 \leq P \leq N$); y que el segmento está incluido en la lista, es decir, $P + L - 1 \leq N$. Se valorará la eficiencia y complejidad tanto en tiempo como en espacio del algoritmo implementado, las cuales debes indicar y justificar.

La función principal proporcionada para hacer pruebas lee de una línea los tres números N, P y L, después de la siguiente línea lee e inserta en la lista los N elementos, llama a la función pedida con los argumentos correspondientes y muestra por pantalla la lista modificada (ver ejemplos). El proceso se repite indefinidamente mientras haya datos en la entrada que procesar.

Entrada	**Salida**
8 3 4 1 2 3 4 5 6 7 8	1 2 6 5 4 3 7 8
8 1 8 1 2 3 4 5 6 7 8	8 7 6 5 4 3 2 1
8 3 1 1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8

Ejercicio 2 [3 puntos]

Se tienen dos vectores, a y b, de enteros ordenados y distintos entre sí con n y $n-1$ elementos respectivamente. Los elementos de b son los mismos que tiene a excepto uno que falta. Se pide implementar un algoritmo eficiente que encuentre ese valor que falta y justificar su coste.
La función principal proporcionada para hacer pruebas comienza leyendo el número de casos a tratar. Cada caso consta de 3 líneas: en la primera se da el número de elementos del primer vector, en la segunda los elementos del primer vector y en la tercera los elementos del segundo vector. Para cada caso de prueba se escribe en una línea el número que falta.

Entrada	Salida
5 5 10 20 30 40 50 10 30 40 50	20
5 10 20 30 40 50 10 20 30 40	50
5 10 20 30 40 50 20 30 40 50	10
1 6	6
2 1 8 1	8

Ejercicio 3 [4.5 puntos]

Las elecciones presidenciales de Estados Unidos se rigen en base a un sistema de voto indirecto. Los ciudadanos no eligen directamente al presidente, sino que votan a los miembros del llamado Colegio Electoral, formado por 538 compromisarios, y estos últimos son los que votan a los candidatos presidenciales. Cada uno de los estados de EEUU tiene un número determinado de compromisarios en el Colegio Electoral. Por ejemplo, el estado de Arizona tiene 11 compromisarios, el de Kansas tiene 6, el de Georgia tiene 16, etc. Dentro de cada estado, el partido que obtiene la mayoría de votos de los ciudadanos se lleva todos los compromisarios de ese estado, aunque haya obtenido la mayoría por un estrecho margen. Por ejemplo, en las pasadas elecciones del año 2020, el Partido Demócrata obtuvo los 16 compromisarios del estado de Georgia habiendo obtenido un 49,5 % de los votos populares frente al 49,2 % del Partido Republicano. En este ejercicio vamos a implementar un sistema de contabilización de votos para un país imaginario que tenga este mismo mecanismo electoral. Este sistema estará encapsulado en un TAD ConteoVotos que proporcione las siguientes operaciones:

- `void nuevo_estado(const string& nombre, int numCompromisarios)`: Registra un nuevo estado en el sistema, con el nombre y número de compromisarios pasados como parámetro. Si ya existía un estado registrado con el mismo nombre, se lanzará una excepción de tipo `std::domain_error` con el mensaje `Estado ya existente`. Puedes suponer que `numCompromisarios` > 0.
- `void sumar_votos(const string& estado, const string& partido, int numVotos)`: Suma `numVotos` a la cantidad de votos recibida por el partido dentro del estado indicado como parámetro. Si el estado no estaba registrado previamente en el sistema, se lanzará una excepción `std::domain_error` con el mensaje `Estado no encontrado`. Puedes suponer que `numVotos` > 0.
- `string ganador_en(const string& estado) const`: Devuelve el nombre del partido que más votos ha obtenido en el estado indicado. Puedes suponer (sin necesidad de comprobarlo en tu código) que el estado ha recibido previamente al menos un voto para algún partido político, y que el partido mayoritario tiene un número de votos estrictamente mayor que los partidos restantes; es decir, no hay empates. Si el estado indicado no estaba registrado previamente en el sistema, se lanzará una excepción `std::domain_error` con el mensaje `Estado no encontrado`.
- `vector<pair<string,int>> resultados() const`: Devuelve una lista de los partidos que han obtenido al menos un compromisario en alguno de los estados. Cada elemento de la lista es un par `(nombre,numTotal)`, donde `nombre` es el nombre del partido, y `numTotal` es la suma total de compromisarios que ha obtenido en todo el país. La lista debe estar ordenada alfabéticamente según el nombre del partido. Puedes suponer (sin necesidad de comprobarlo en el código) que todos los estados registrados en el sistema tienen un partido ganador, es decir, que no hay empate en ninguno de ellos.

Se pide elegir una representación adecuada e implementar las operaciones del TAD descrito, utilizando las librerías de la STL o los TADs vistos en clase. Indica y justifica brevemente el coste de cada una de las

operaciones. Ninguno de los métodos del TAD debe realizar operaciones de E/S (el manejo de E/S se hace en la función `resuelveCaso`).

Entrada La entrada consta de una serie de casos de prueba. Cada caso está formado por una serie de líneas, en las que se muestran las operaciones a llevar a cabo, una por cada línea: el nombre de la operación seguido de sus argumentos. La palabra `FIN` en una línea indica el final de cada caso. Todos los nombres de estados y partidos políticos son secuencias de caracteres alfanuméricos sin espacios.

Salida Las operaciones `nuevo_estado` y `sumar_votos` no producen salida, salvo en caso de error. Con respecto a las restantes:

- Tras llamar a `ganador_en` debe escribirse la cadena `Ganador en XXX: ZZZ`, donde `XXX` es el estado pasado como parámetro, y `ZZZ` es el resultado devuelto por la operación.
- Tras llamar a `resultados` deben imprimirse tantas líneas como elementos tenga la lista que haya sido devuelta por esta operación. Cada línea debe contener el nombre del partido político y el número de compromisarios total, separados por un espacio.

Cada caso termina con una línea con tres guiones `---`. Si una operación produce una excepción, debe escribirse una línea con el mensaje de la excepción, y no escribirse nada más para esa operación.

Ejemplos de Entrada/Salida:

```
nuevo_estado Arizona 11
nuevo_estado RhodeIsland 4
nuevo_estado California 55
sumar_votos Arizona REP 1000
sumar_votos Arizona DEM 500
ganador_en Arizona
sumar_votos California DEM 1000
sumar_votos RhodeIsland REP 200
ganador_en California
sumar_votos Arizona DEM 600
ganador_en Arizona
resultados
FIN
nuevo_estado Nevada 6
sumar_votos Arizona P1 10
sumar_votos Nevada P2 200
ganador_en Nevada
ganador_en Arizona
FIN
nuevo_estado Illinois 20
sumar_votos Illinois P1 10
ganador_en Illinois
resultados
sumar_votos Illinois P2 20
ganador_en Illinois
resultados
sumar_votos Illinois P3 30
ganador_en Illinois
resultados
FIN
```

```
Ganador en Arizona: REP
Ganador en California: DEM
Ganador en Arizona: DEM
DEM 66
REP 4
---
Estado no encontrado
Ganador en Nevada: P2
Estado no encontrado
---
Ganador en Illinois: P1
P1 20
Ganador en Illinois: P2
P2 20
Ganador en Illinois: P3
P3 20
---
```