Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022

Examen final. Convocatoria extraordinaria 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-Junio22.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 [4.5 puntos]

Ha comenzado el curso escolar y los profesores tienen que asignar los diversos cargos y tareas de cada clase: quién será el delegado? quién anotará los deberes en la agenda? quien borrará la pizarra entre clases? quién se encargará de recoger y repartir los trabajos? Para ello han solicitado a los alumnos que les indiquen sus preferencias. Darán una puntuación entre 1 y 10 a cada tarea, a mayor puntuación más ganas de hacerla. Por otro lado, debe haber siempre dos alumnos responsables para cada tarea, para que trabajen en equipo y para asegurar que la tarea no se quede sin hacer cuando uno de los dos falte. También, para intentar que participen el mayor número posible de alumnos y evitar que unos pocos acaparen todas las tareas, ningún alumno podrá ocuparse de más de t tareas. Lógicamente, la idea es tener en cuenta las preferencias de los alumnos para que estén lo más contentos posible, pero rápidamente se dan cuenta de que no es nada fácil dar con la asignación óptima (aquella que maximice la suma de puntuaciones). Como saben que estamos estudiando en la Facultad de Informática, nos piden ayuda para encontrar dicha asignación óptima. Se pide:

- Implementa un algoritmo de vuelta atrás que resuelva el problema.
- Explica claramente cuál es la tupla solución y los marcadores que has utilizado.
- Explica detalladamente cómo se podría implementar una poda por estimación (no es necesario llegar a implementarla en el algoritmo) y pon un ejemplo de aplicación.

La función principal proporcionada para hacer pruebas procesa casos de prueba hasta encontrar uno que empiece por 3 ceros. Cada caso de prueba consta de varias líneas. En la primera se muestra el número de tareas n que hay que realizar, el número de alumnos de la clase a y el número máximo de tareas que puede realizar un alumno t. En las a líneas siguientes se muestran n valores que representan las preferencias del alumno para cada tarea. El número de tareas es un entero, $1 \leq n \leq 6$, el número de alumnos es un entero, $2 \leq n \leq 8$ y el número máximo de tareas que puede realizar un alumno $1 \leq t \leq 6$. Se garantiza que siempre existe una asignación posible, es decir $a * t > 2 * n$. No es necesario comprobar ninguna de estas condiciones, simplemente asumimos que siempre se cumplen. Para cada caso de prueba se muestra en una línea la suma de las preferencias de los alumnos que realizarán las tareas. Para depurar se recomienda calcular e imprimir las asignaciones concretas. Por ejemplo, para los tres casos de la derecha, serían 0 3 0 2 3 4, 0 1 0 1 0 1 0 1 y 0 1 1 2 0 2, respectivamente.

Entrada	Salida
3 5 2 5 10 2 1 8 4 3 10 3 4 7 7 1 7 6	42
4 2 4 5 5 5 5 7 7 7 7	48
3 3 2 10 10 8 8 9 1 1 7 8	50
0 0 0	

Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022

Examen final. Convocatoria extraordinaria 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-Junio22.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]

Implementa una operación interna en la clase `list` vista en clase (fichero `list_eda.h`) que adelante un número de posiciones dado a un segmento (elementos en posiciones consecutivas) de la lista pasada como argumento. 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 operación pedida con los argumentos correspondientes y se muestra por pantalla la lista modificada (ver ejemplos y plantilla proporcionada).

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

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-Junio22.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 [2.5 puntos]

Dado un vector no vacío, ordenado de forma creciente y que comienza con una serie de números consecutivos, se pide implementar un algoritmo eficiente que encuentre el último elemento de la serie que se encuentra en el vector. Indica y justifica la complejidad del algoritmo implementado.

Entrada La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba consta del número de elementos del vector y el contenido del mismo.

Salida Por cada caso de prueba el programa escribirá el último elemento de la serie que se encuentra en el vector.

Ejemplos de Entrada/Salida:

3 5 2 3 34 78 99 1 6 4 1 2 3 4	3 6 4