Memoria

La práctica se ha resuelto utilizando el TDA bintree proporcionado, la implementación de los métodos pedidos se describe a continuación:

QuienEsQuien::crear_arbol()

Partiendo de los vectores "atributos" y "personajes", además de la matriz "tablero", se construye el árbol de la siguiente manera:

El árbol se crea a partir de la primera entrada de la matriz, este nodo corresponde a la raíz del árbol y será siempre la primera pregunta que formulará el juego.

Después, para cada fila de la matriz (y comenzando inicialmente en la raíz), formulamos las preguntas según los valores del "tablero" hasta que llegamos a un nodo nulo, en ese caso, insertamos la pregunta en la posición de ese nodo, y repetimos para la siguiente. Al terminar la fila, colocamos el personaje según el valor de la última pregunta.

Para aprovechar el recorrido en el árbol, durante la inserción de los nodos aumentamos los contadores de nº de personajes.

Con este método, creamos un árbol con un bajo número de nodo, pero sin ser equilibrado. Pese a ello, es probable que existan nodos redundantes, por lo que posteriormente se llama al método correspondiente.

QuienEsQuien::eliminar_nodos_redundantes(bintree<Pregunta>::node nodo)

Utilizamos una función recursiva que para cada nodo (comenzando desde la raíz) comprueba si alguno de sus hijos es nulo (obviamente, primero aseguramos que no sea un nodo hoja).

Si al nodo le falta alguno de sus hijos (y por tanto es un nodo redundante), lo eliminamos y subimos al hijo a la posición del padre.

QuienEsQuien::preguntas_formuladas (bintree<Pregunta>::node jugada)

Este método devuelve las preguntas realizadas hasta la jugada actual en una lista.

Consiste en introducir las preguntas en la lista hasta que se alcanza la raíz, yendo de padre en padre.

Por cómo está implementado el método iniciar_juego, la información de las jugadas anteriores no se puede sacar, puesto que ese método trabaja con un árbol auxiliar que es un subárbol del principal.

Francisco José Cotán López
Ignacio Vellido Expósito

QuienEsQuien::profundidad_promedio_hojas ()

Calcula la profundidad promedio de todas las hojas, de la siguiente manera:

Se llama a profundidad_total_hojas (este método se detalla a continuación) con nivel con valor 0, la raíz, y num_hojas igual a 0. Posteriormente dividimos la profundidad total devuelta por el número de hojas, dándonos el valor pedido.

QuienEsQuien::iniciar_juego()

Este método averigua los personajes del jugador hasta que se introduce la opción de salir.

Para cada jugada, formula la pregunta, hasta que llega a un nodo con un nº de personaje igual a 1, lo que significa que ha descubierto a la persona, escribiendo el nombre en pantalla.

QuienEsQuien::información_jugada (bintree<Pregunta>::node jugada_actual)

Devuelve los personajes aún no tumbados en el tablero.

QuienEsQuien::eliminar_personaje (const string &nombre)

Elimina un personaje dado su nombre

QuienEsQuien::aniade_personaje (const string &nombre, const vector<bool> atributos)

Añade un personaje dado su nombre y atributos

Además, se han necesitado los siguientes métodos auxiliares:

- QuienEsQuien::Swap (QuienEsQuien &)

- QuienEsQuien::esHoja()

- Pregunta::operator=(const Pregunta &p)

- QuienEsQuien::profundidad_total_hojas (bintree<Pregunta>::node nodo, int nivel, int &num_hojas)

En este último método, utilizamos el argumento num_hojas (pasado por referencia) para devolver el número de hojas que hay en el árbol. Funciona de la siguiente manera:

Se empieza con una profundidad_total de valor 0, y el nodo es hoja éste se incrementa con el nivel actual, y adicionalmente, aumentamos el número de hojas del árbol.

En caso de que no sea un nodo hoja, llamamos recursivamente a este método tano para su nodo izquierdo como para el derecho incrementando el nivel en 1.