República Bolivariana de Venezuela
Universidad Central de Venezuela - Facultad de Ciencias
Escuela : Computación
Asignatura: Algoritmos y Estructuras de Datos.
Semestre: 2-2019
Alumno: Yunior Moreno C.I. 26465131
Sección: C1

Proyecto #1. Análisis.

Para este proyecto se pide simular un tanque averiado que intenta escapar del campo de batalla, para ellos se debe simular un mapa con muros y puntos de reparación que habrá que recorrer con la intención de hallar la salida, nuestro tanque en un principio solo podrá moverse hacia arriba y hacia la derecha, si se encuentra un punto de reparación el tanque podrá moverse en el resto de las direcciones(abajo y a la izquierda). La cantidad de casos a simular así como la características del mapa para cada caso se deben leer desde un archivo de entrada y como respuesta se deberá escribir en un archivo de salida el porcentaje de muertes seguras para cada mapa particular cuando se parte desde cada casilla válida del mapa así como una lista de los caminos que si llevaron al tanque a escapar con éxito.

La entrada:

Para la entrada se crean dos dos variables de tipo fstream (una *ifstream* y otra *ostream*) que serían los archivos de entrada y salida (*TankEscape.in* y TankEscape.out respectivamente), luego desde el archivo de entrada "*entrada*" leer un **entero** correspondiente al número de casos y guardarlo en una variable n.

Luego con un ciclo ***for*** principal de n iteraciones se leeran los datos correspondiente a cada uno de los casos. Lo primero es leer desde el archivo de entrada tres enteros (f, c, w), los dos primeros correspondientes al tamaño del mapa y el restante a la cantidad de muros, luego, comprobar que sean válidos, es decir que las filas (f) y las columnas (c) no sean menores que 1 y que la cantidad de muros (w) no sea negativa.

Si la comprobación falla se saltará el resto caso y pasaría a la siguiente iteración del ciclo (de haberlo), asumiendo que los datos que leerán en ese punto son correctos (Esto es así por consideraciones del problema).

Si la comprobación va bien se creara una matriz dinámica de tamaño f x c que corresponderá al mapa, me decidí por una matriz dinámica para poder pasar el mapa por parámetros de forma más sencilla. Una vez creada la matriz que representa el mapa se inicializan todas sus posiciones a '.' que representan una casilla vacía.

Luego se construirán los muros en el mapa, para ello se crea otro ciclo ***for (for** muros) dentro de del for principal**,** este **for** muros irá* desde 0 hasta w-1 y para cada iteración del ciclo se leerán tres variables de tipo entero (fila, col1, col2) que corresponden a la fila en la que irá el muro, la columna donde empieza y la columna donde termina. Es necesario revisarar que las posiciones sean válidas, es decir que f corresponda a una fila del mapa y que los columnas 1 y 2 también correspondan con columnas del mapa, además que la segunda columna sea mayor o igual que la primera.

Si todos los datos son válidos se agregan los muros al mapa (con ayuda de una variable *aux),* sino, se salta el resto del caso y continua con la siguiente iteración del ***for*** principal si aún no se ha terminado.

Una vez leídos y construidos los w muros se leerá desde el archivo entrada un entero (*fixPoint*) que indica la cantidad de puntos de reparación que habrá en el mapa y deberán ser mayor que 0 y menor que cuatro (0 <= ***fixPoint*** <= 4), y luego se crea un ciclo ***for*** desde 0 hasta ***fixpoint*** (***for fixPoint***) y dentro del ciclo de leerán desde la entrada 2 variables por iteración (f, c) que serán las coordenadas donde se colocará un ***fixPoint***, se válida la coordenada (que el (f,c) sea parte del mapa y dist**int**o de un muro '#') y luego, si la coordenada es valida se agrega el ***fixPoint*** en ese punto.

Una vez leídos y agregados los puntos de reparación (***fixPoints***) el mapa esta completo y se llama una una función que procese este mapa, luego se libera la memoria del mapa y se pasa al siguiente caso.

Una vez terminados todos los casos, se cierran los archivos de entrada y salida.

Para encontrar la salida:

Ya que para poder llegar a la salida desde el punto de inicio se deben recorrer varios caminos hasta encontrar el correcto se hace uso del *backtraking* para ir moviéndose en el mapa e ir registrando el camino recorrido, esto se hará con una función llamada BT que se encargara de aplicar el *backtracking.*

Para ello lo primero que hace la función BT es comprobar si ya hemos llegado a la salida, en cuyo caso retornara ***True*** en ese punto.

Si la casilla actual no es la salida, entonces lo siguiente sería comprobar si es un punto de reparación, si efectivamente es un punto de reparación y además es el primero que encontramos (*fixed* == ***False***), entonces se reinicializara el mapa_clon, se cambiara *fixed* a ***True***, se actualizara el camino agregando una 'X' y se marcara esa casilla como visitada en el mapa clon. Si no es un punto de reparación o el tanque ya fue reparado simplemente sigue con la función.

Luego en un ciclo ***for*** de 0 a 3 (una iteración por cada camino posible), se preguntara si la siguiente casilla es válida, es decir, es parte del mapa, no es un muro y no lo hemos visitado. Paro los casos de abajo ('S') y la izquierda('A') se debe revisar además que el tanque este reparado (*fixed* == ***True***).

Si se cumple la condición entonces se marcará la casilla a la que pretende mover como marcada en el mapa clon y se agregará el camino correspondiente a la variable camino entrará entonces se llamara recursivamente a BT con el punto de inicio modificado según corresponda (0 = 'W' arriba; 1 = 'D' derecha ; 2 = 'S' abajo ; 3 = 'A' izquierda). Si esta llamada a BT devuelve ***True*** la variable alive se actualizara a ***True*** (en caso de que alguna subllamada de BT la cambiase a ***False***), se regresa el mapa clon al valor original para este punto se se rompe el ciclo.

Si devuelve ***False***, también se regresa el mapa clon al valor original para este punto se se rompe el ciclo y además se elimina de la variable camino el camino que se agrego al principio del if.

Al terminar el ciclo, si se recorrieron todos los caminos y BT no devolvió ***True*** para ninguno, entonces se cambia la variable *alive* a ***False***.

Para terminar se retorna el valor de *alive*.

Para Recorrer Los Caminos:

La función encargar de recorrer el mapa desde un punto de inicio es la función BT(...), pero una función procesar(...) a la que se le pasan como parámetros los el mapa , sus dimensiones y el archivo de salida (este último como referencia), será la que se encargue de llamar inicialmente a BT y lo hará para cada caso inicial valido, en procesar(...), se crearán e inicializaran las variables que se usaran como parámetros en cada llamada a BT:

```
bool alive = True;
bool fixed = False;
string camino;
```

y las variables para construir la respuesta que es enviada al archivo de salida:

```
float casillas_validas = 0.0;
float muertes = 0.0;
string total_caminos = "";
char** mapa_clon;
```

La variable *alive* indica si nuestro tanque sigue vivo, es decir si aun nos quedan caminos por probar para llegar a la salida, al principio de cada iteración deberá ser ***True*** y se pasa por referencia a BT.

La variable *fixed* indica el estado de nuestro tanque, si *fixed* es ***True***, entonces nuestro tanque ya fue reparado, y podremos moverlo hacia la izquierda('A') y hacia abajo('S'), esta variable se para por referencia a la función BT, será inicializada a ***False*** para cada llamada inicial a BT;

La variable camino es una variable acumulativa que irá registrando el camino recorrido en BT desde el punto inicial, al principio de cada iteración contendrá un mensaje indicando el punto de inicio: "Camino desde mapa[i][j]:"

La variable mapa_clon es una copia del mapa original en el que se irán marcando las casillas que ya se han visitado dentro de BT (para cada iteración) para evitar que el tanque se quede dando vueltas entre dos(o más) casillas, se inicializa con una función inicializar_mapa_clon(…) que solo copiara el mapa original sin alteraciones, ya que es necesario reinicializar el mapa una vez encontrado un punto de reparación para que el tanque pueda volver por donde vino de ser necesario empezando a marcar el mapa clon nuevamente desde el punto de reparación.

Una vez las variables son inicializadas se crea un ***for*** anidado que recorrerá cada casilla del mapa y si una casilla no es un muro, inicializara las variables *fixed*, camino y el mapa clon, aumentará en uno la variable casillas_validas y luego un if comprobara el valor de retorno de BT con el punto de inicio correspondiente a los índices (i,j) de las iteración, si BT devuelve ***False***, significa que empezando desde ese punto, no se pudo llegar a la salida y nuestro tanque fue destruido, aumentaremos en uno la variable muertes y pasaremos al siguiente paso del ciclo. Si devuelve ***True*** entonces la variable total_caminos se le agregara el contenido acumulado en la variable camino y se continua con el ciclo.

Una vez terminado el ciclo se calcula el porcentaje de muertes con la formula ***(muertes / caminos_validos *100)*** y solo quedaría construir la respuesta que luego se escribirá en el archivo de salida. (En forma simplificada sería *salida <<caso << porcentaje de muertes << caminos*).