Asignatura: Fundamentos de algoritmia
Evaluación continua. Problema de algoritmos de vuelta atrás.

Profesor: Isabel Pita.

Enunciado: Ha caído una gran nevada y el ayuntamiento debe limpiar las calles de la ciudad. El alcalde quiere maximizar los metros de calle que se limpian, pero solo tiene una cierta cantidad de sal. Hay que tener en cuenta que las calles se limpian completas. Para poder decidir cuales se van a limpiar requiere a sus empleados información sobre la longitud de cada calle de la ciudad, y de la cantidad de sal que se requiere para limpiarla. La cantidad de sal que requiere una calle varía de unas a otras dependiendo de su orientación. Como además quiere que una mayoría de vecinos estén contentos ha decidido que el total de vecinos a los que se les limpia la calle debe superar un cierto número.

Datos de entrada: Número de calles de la ciudad, cantidad total de sal que tiene el ayuntamiento y cantidad mínima de personas a las que el alcalde quiere que se les limpie la calle. Longitud de cada calle, cantidad de sal necesario para limpiar cada calle y número de vecinos de cada calle del ayuntamiento.

Datos de salida: Obtener una lista de las calles que se deben limpiar, es decir, que limpiando estas calles se maximiza el número de metros de calle que se limpian en total, se tiene sal suficiente para limpiarlas completas y al menos el número mínimo exigido de vecinos quedan con su calle limpia. Si hay varias soluciones óptimas se dará una de ellas.

Se pide:

1. Implementa un programa que resuelva el ejercicio propuesto.

2. Debéis crear vosotros los casos de prueba ya que no se va a utilizar el juez para ello. Crea varios casos de prueba *pequeños* para probar tu programa.

3. La solución del problema debe incluir una solución inicial que permita inicializar la solución mejor. Diseña un caso de prueba *grande* que permita ver la diferencia del tiempo de ejecución cuando se utiliza la inicialización de la solución respecto a cuando se inicializa la solución mejor al máximo/mínimo del tipo. Para ello debes pensar un caso de prueba con el cual la ejecución del programa utilizando la solución inicial consiga podar muchas ramas del árbol de ejecución. Mide el tiempo que tarda el programa en un caso y en otro.

4. La solución del problema debe incluir una estimación del coste de la solución que queda por construir que permita podar adecuadamente el árbol de exploración. Diseña un caso de prueba *grande* que permita ver la diferencia del tiempo de ejecución cuando se utiliza la estimación respecto a cuando no se utiliza. Mide el tiempo que tarda el programa cuando se utiliza la estimación y cuando no se utiliza.

5. Haz un vídeo en el que se explique el árbol de exploración con el que se ha desarrollado el programa y el vector solución que se va a utilizar. El vídeo debe incluir una explicación de la solución inicial que se ha empleado y de la estimación realizada. Se comentarán los tiempos que se han tomado en cada caso. Al principio del vídeo debe salir el alumno para que se vea que lo está realizando él y mostrar su DNI. Los vídeos se destruirán una vez corregidos y no serán empleados con otro fin que evaluar la prueba de vuelta atrás.

6. Debes subir al campus a la tarea abierta para ello el fichero .cpp con la implementación del programa, dos ficheros con los casos de prueba, uno de ellos con los casos de prueba *pequeños* y el otro con el caso de prueba *grande*. Si se necesitan varios casos de prueba grandes se pueden subir varios ficheros uno con cada caso. El link del vídeo realizado se pondrá al comienzo del archivo .cpp con el programa implementado junto con el nombre del alumno. Si el vídeo es pequeño se puede subir directamente al campus en lugar de utilizar la herramienta youtube para compartirlo.

7. Se valorarán los comentarios incluidos en la solución, asi como las decisiones tomadas respecto a los aspectos que no están completamente cerrados en el enunciado.