INF-221 Algoritmos y Complejidad, 2017-2

Tarea 2

Profesor: Diego Arroyuelo
Ayudantes: Fernanda Weiss, Manuel Calquín
`fernanda.weiss.13@sansano.utfsm.cl`,
`manuel.calquin.14@sansano.utfsm.cl`

Fecha de entrega: 27 de noviembre, 2017
Plazo máximo de entrega: 5 días.

Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes **ANTES** de comenzar la tarea. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorio B-032 (LDS). Deben usarse los lenguajes de programación C o C++. Se recomienda compilar en el terminal usando `gcc archivo.c -o output -Wall` (en el caso de lenguaje C) o `g++ archivo.cpp -o output -Wall` (en el caso de C++). Si usa C++, está permitido emplear estructuras de datos de la `stl`, en caso de ser necesarias.

Advanced Drone Delivery System

La *Asociación Mundial de Drones* (AMD) ha decidido entrar al emergente mercado de los despachos de productos automatizados mediante drones, usando su nuevo sistema "Advanced Drone Delivery System$^{\text{TM}}$". Para que este negocio sea rentable, la AMD debe solucionar problemas críticos. Por ejemplo, perder drones por colisiones con edificaciones es poco rentable: se debe reemplazar el dron, así como indemnizar a los dueños del inmueble afectado.

Para solucionar dicho problema a un bajo costo, se han instalado radares de ultrasonido en cada dron. En conjunto, una flota es capaz de detectar la ubicación de las edificaciones que obstaculizan el espacio, en un mapa 2D (una foto). Sin embargo, no se cuenta con un programa que interprete dichos datos. Por este motivo, ha abierto una licitación exclusiva para los alumnos de Algoritmos y Complejidad, quienes deberán implementar un programa que interprete dichos datos. La remuneración por el trabajo se hará a través de un mecanismo de la AMD conocido como *NOn-Traditional Allowance* (NOTA).

Dicho programa debe:

- Generar un *contorno* de los edificios hacia los cuales se dirige una flota de drones. En esta etapa los drones trabajan en conjunto, por lo que es ideal dividir el problema en sub–problemas y luego generar una solución completa. Para evitar colisiones, la complejidad de tiempo de este algoritmo debe ser $O(n \log n)$, siendo n la cantidad de edificios.
- Para cada dron, indicar si se encuentra en una trayectoria de colisión, en base al *contorno* ya generado. Esta etapa es crítica y debe ser realizada en tiempo $O(\log n)$, o el dron colisionará.

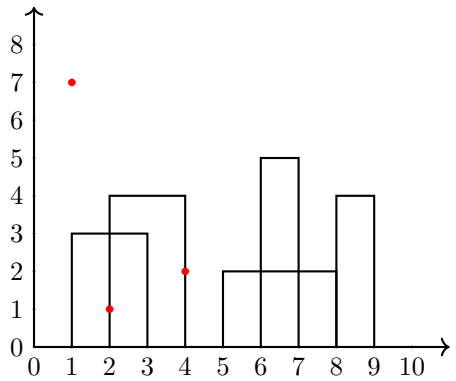
Formato de Entrada

La entrada de datos se hará a través de la entrada standard (`stdin`). La entrada comienza con una línea que contiene un único entero `N` ($0 < \texttt{N} \leq 10.000.000$), indicando la cantidad de edificaciones detectadas por los sensores de los drones. A continuación, le siguen `N` líneas, cada una conteniendo una 3-tupla de valores `L H R` ($0 \leq \texttt{L} < \texttt{R} < 2^{32} - 1$, $0 < \texttt{H} < 2^{31} - 1$) los cuales indican, para cada edificación, la coordenada del lado izquierdo (pared izquierda), la altura y la coordenada del lado derecho (pared derecha), respectivamente. Las 3-tuplas están ordenadas de manera no-decreciente respecto al valor `L`. Posteriormente sigue una línea con un único valor `D` indicando la cantidad de drones de la flota. Finalmente, le siguen `D` líneas, cada una con una 2-tupla `X Y` ($0 < \texttt{X}, \texttt{Y} < 2^{32} - 1$), las cuales indican, para cada dron, su posición en el eje `X` y su altura en el eje `Y` respectivamente. Las 2-tuplas no necesariamente están ordenadas por valor de `X`.

Un ejemplo particular de entrada es el siguiente:

```
5
1 3 3
2 4 4
5 2 8
6 5 7
8 4 9
3
4 2
1 7
2 1
```

Gráficamente se interpreta de la siguiente manera:



Los puntos rojos en la figura indican la posición de los drones.

Hint: para probar su programa de una mejor manera, ingrese los datos de entrada con el formato indicado en un archivo de texto (por ejemplo, el archivo `input.dat`). Luego, ejecute su programa desde la terminal, redirigiendo la entrada standard como a continuación:

```
./tarea2 < input.dat
```

De esta manera evita tener que entrar los datos manualmente cada vez que prueba su programa, y evita errores.

Formato de Salida

La salida de datos se hará a través de la salida standard (`stdout`). Su programa debe generar un *contorno* con los bordes exteriores de los edificios. Definimos un contorno como una colección de pares `L S`, en donde

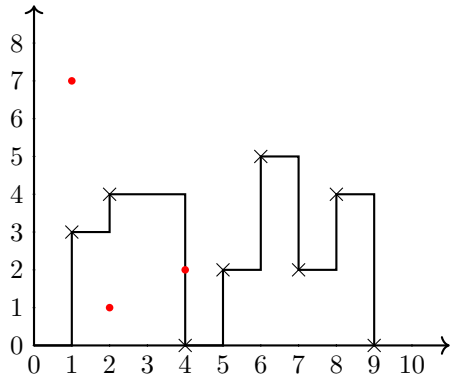
el valor `L` denota la coordenada x, y `S` la altura de cada uno de los puntos que conforman el contorno. Los pares están ordenados de forma creciente por coordenada x.

La salida comienza con una línea que contiene un único valor `M`, que es la cantidad de pares que conforman el contorno. Luego le siguen `M` líneas, indicando de manera ascendente los valores `L S` de cada componente del contorno (ver el ejemplo más abajo para una ilustración). Posteriormente, le sigue una línea indicando únicamente el valor de `D`, seguido de `D` líneas, donde, para cada dron (en el mismo orden del input), se indica si éste se encuentra en una trayectoria de colisión o no respecto al contorno, imprimiendo `true` o `false` respectivamente.

La salida correspondiente al ejemplo anterior es la siguiente:

```
8
1 3
2 4
4 0
5 2
6 5
7 2
8 4
9 0
3
true
false
true
```

La posición de los drones no cambia, sin embargo, note que el contorno puede verse gráficamente así (donde cada punto de la salida es marcado con una $\times$):



Note además que si un dron se encuentra sobre el contorno, se considera que choca con la edificación.

Bonus por Eficiencia

Se dará un bonus de 25 puntos a las 5 tareas que ejecuten más rápido. Dicho bonus se podrá usar en cualquiera de las tareas del curso.

Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea2-apellido1-apellido2-apellido3.tar.xz`

(reemplazando sus apellidos según corresponda) en el sitio Aula del curso, a más tardar el día 27 de noviembre de 2017, a las 23:55:00 hrs (Chile Continental), el cual contenga:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. Los archivos deben compilar!
- `nombres.txt`: Nombre, ROL y qué programó cada integrante del grupo.
- `README.txt`: Instrucciones de compilación u observaciones, en caso de ser necesarias.
- `Makefile`: Reglas de compilación para compilar su tarea con el comando `make`. Se exige el uso del flag `-Wall`.

Restricciones y Consideraciones

- Por cada día de atraso (o fracción) en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.
- Pueden programar la tarea en C o C++ según ustedes consideren conveniente.
- Las tareas deben compilar en los computadores que se encuentran en el laboratorio B-032. **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Por cada *Warning* en la compilación se descontarán 5 puntos.
- Si se detecta **COPIA** la nota automáticamente sera 0 (CERO), para todos los grupos involucrados. El incidente será reportado al Jefe de Carrera (JLML).
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.