INF-564 Diseño avanzado de algoritmos, 2019-1

Tarea 2

Profesores: Diego Arroyuelo, Marcelo Mendoza

Fecha de entrega: 21 de junio, 2019

Reglas del Juego

La presente tarea debe hacerse de forma individual. Debe usarse el lenguaje de programación de su preferencia.

Problema 1: Controlador de Vuelos

Un aeropuerto internacional debe administrar miles de vuelos diarios. Entre las tareas de la torre de control se encuentra asignar rutas y alturas de vuelo para evitar que los aviones colisionen. En cualquier momento puede haber una gran cantidad de aviones volando a una misma altura, por lo que para un operador puede ser difícil preocuparse de todos ellos al mismo tiempo. Para ayudar a los operadores con este problema usted debe diseñar un algoritmo que indique, por cada altura, los aviones que están más cerca y que por lo tanto tienen mayores probabilidades de colisionar. Así los operadores podrán saber cuales son los aviones críticos a observar. Dado que los aviones se mueven rápido, su algoritmo debe ser capaz de entregar los resultados rápidamente. Por otro lado, los recursos de la torre de control son limitados (por ejemplo, la memoria de los computadores). Por lo tanto, su algoritmo debe ser eficiente en tiempo y en espacio de memoria usado. Asuma que la distancia entre aviones se calcula usando distancia euclidiana. Diseñe su algoritmo usando la técnica *dividir y conquistar*.

Escriba un informe usando LaTeX, explicando brevemente la idea de su solución, y analizando asintóticamente su algoritmo. Implemente además una solución de fuerza bruta, y compárela experimentalmente con su algoritmo. Incluya los resultados en su informe.

Formato de Entrada

La entrada de datos se hará a través de la entrada estándar (`stdin`). Los datos de entrada consisten en una cantidad indefinida de casos de prueba finalizados con el fin de archivo (`EOF`). Cada caso de prueba corresponde a un conjunto de aviones a una cierta altura y comienza con una línea que indica el número

n de aviones a considerar. A continuación le siguen n líneas, donde cada una contiene las coordenadas x_i e y_i del i-ésimo avión, separadas por un espacio. Un ejemplo de entrada es el siguiente:

```
6
3.0 5.0
6.0 7.0
3.9 4.1
1.0 1.0
4.0 4.0
4.0 1.0
4
9.0 2.0
930.0 1024.0
334.0 144.0
15.5 1.0
```

Si emplea lenguaje `C`/`C++`, puede probar su programa de mejor manera ingresando los datos de entrada con el formato indicado en un archivo de texto (por ejemplo, el archivo `input.txt`). Luego, ejecute su programa desde la terminal, redirigiendo la entrada estándar como a continuación:

```
./tarea2 < input.txt
```

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

Formato de Salida

La salida se hará a través de la salida estándar (`stdout`). Para cada caso de prueba se debe imprimir el par de puntos más cercanos (ordenados por la coordenada x, luego por la y) separados por un salto de línea. Cada caso de prueba debe estar separado por un salto de línea adicional. La salida correspondiente al ejemplo del enunciado es:

```
3.9 4.1
4.0 4.0

9.0 2.0
15.5 1.0
```

Problema 2: Algoritmo Insertion Sort

Proponga mejoras asintóticas al algoritmo `Insertion Sort` estudiado en clases, de tal manera que saque ventaja de ciertas entradas que están casi ordenadas. Asuma que la entrada es un conjunto de números enteros, es decir, no hay elementos repetidos. Algunos tipos de entrada de los que su algoritmo debería tomar ventaja son:

- $\langle \mathtt{i}, \mathtt{i}+1, \cdots, \mathtt{n}, 1, 2, \cdots, \mathtt{i}-1 \rangle$
- $\langle 8, 9, 10, 11, 12, 3, 4, 5, 18, 19, 20, 16, 17, 13, 14, 15, 16 \rangle$
- $\langle 8, 11, 15, 16, 3, 4, 5, 18, 20, 21, 25, 26, 27, 7, 10, 14 \rangle$
- $\langle 8, 11, 15, 16, 19, 5, 4, 3, 1, 27, 23, 21, 20, 18, 17, 13, 14, 22, 24 \rangle$
- etc.

Algunos de esos ejemplos podrían ser casos particulares de otros.

Escriba un informe explicando las mejoras propuestas y un breve análisis de la misma (incluya aspectos como la simpleza del algoritmo, espacio adicional utilizado, etc.). Incluya además el pseudo código de sus variantes del algoritmo, el tipo de entradas del que toma ventaja, y una evaluación experimental de sus variantes. El informe debe ser escrito en LaTeX.

Entrega de la Tarea

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

`tarea2-apellido.tar.gz`

en el sitio moodle del curso, a más tardar el día 21 de junio, 2019, 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!
- `informe.pdf`: Un informe (escrito en LaTeX). Debe estar claramente escrito, con formato de artículo científico.
- `README.txt`: Instrucciones de compilación en caso de ser necesarias.

La tarea deberá ser defendida el día viernes 28 de junio, 2019, en horario de clases. Cada estudiante debe realizar una exposición al resto de la clase, mostrando los resultados obtenidos para ambos problemas.

Restricciones y Consideraciones

- Por cada día de atraso 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.
- Aún cuando se permite discutir los problemas en grupos, **las tareas son individuales**.