Experimentación

Para el proceso de experimentación del problema se plantearon distintas pruebas para corroborar que el algoritmo propuesto funcionara correctamente y que la cota de complejidad encontrada y justificada en la sección anterior, en la práctica, se cumpliera.

Al igual que en el ejercicio 1, dado que el CPU de la computadora utilizada para tomar los tiempos no está atendiendo únicamente a nuestro proceso, realizar una sola vez cada prueba podría darnos valores que no son cercanos a los reales. Por lo que para minimizar este margen de error, a cada prueba se la hizo ejecutar un total de 10.000 veces y se tomó el mejor valor. Notar que tomar el mejor valor no es una mala decisión, ya que mientras más chico sea el valor, más cerca estamos del valor real de tiempo que toma el algoritmo para una instancia dada.

En cada prueba se tomaron métricas para la posterior evaluación del algoritmo en la práctica. Notar que la medición no contempla tiempos de entrada/salida de datos, sino que contempla solamente el núcleo del algoritmo.

Se representó la información tomada mediante gráficos 2D que permitan ver de una manera más clara los resultados obtenidos en las pruebas. Estos fueron realizados con el software QitPlot que la cátedra proveyó.

En cuanto a qué casos testear, nuestro algoritmo no presenta casos “border”. Es decir, no tiene un peor/mejor caso, sino que para cualquier instancia de edificios cargada, realizará el mismo procedimiento, tomando O(n log n). Ni el tamaño de los edificios, ni su posición en el suelo de la ciudad, ni la posición relativa entre ellos afecta el tiempo de computo del algoritmo, por lo que el único parámetro variable a la hora de realizar pruebas es la **cantidad** de edificios.

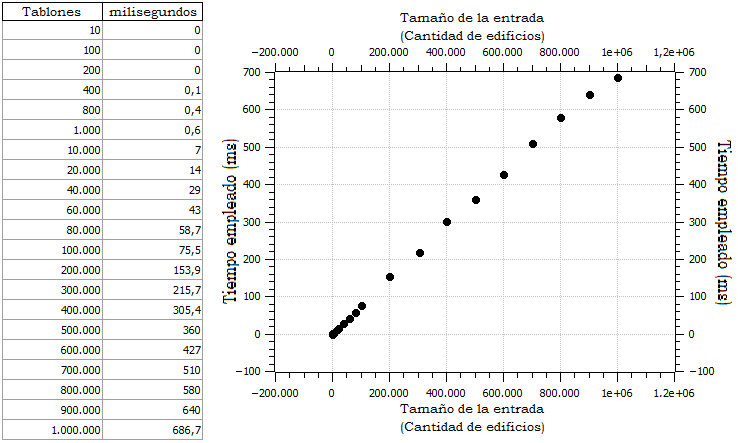
Esto es así ya que el algoritmo que diseñamos, utiliza la estrategia de Divide & Conquer para su resolución, como fue mencionado en insisos anteriores. Esta técnica va dividiendo el problema en varios sub-problemas, sin importar la relación entre los edificios de la instancia. Por lo que, si suponemos que existiera un mejor/peor caso de entrada, de todas formas este sería fraccionado en sub-problemas más pequeños hasta llegar al caso base, y el formato de entrada original se habría perdido. Por este motivo es que el algoritmo no presenta un peor ni un mejor caso de resolución.

Dicho, eso, se diseñó un programa que dado un numero n de edificios, genera n edificios aleatorios para probar el algoritmo. Para facilitar la tarea de experimentación, dicho programa era capaz de generar más de una instancia aleatoria, con distintas cantidades de edificios (cada una elegida en la interfaz de dicho programa).

Para todos los casos, se eligió una precisión de hasta 0,0001 ms (milisegundos). De ser menor, la notamos como 0.

En todos los casos se pudo comprobar que la práctica refleja lo expuesto en incisos anteriores.

A continuación presentamos el gráfico 2D que refleja las pruebas realizadas. Para cada tamaño se realizaron pruebas con instancias distintas y aleatorias y las diferencias muy sutiles (del orden de los microsegundos). A sí mismo, a cada una de esas distintas instancias experimentadas de cada tamaño, se la hizo ejecutar un total de 10.000 veces por los motivos explicados anteriormente.



ALGUNAS CONCLUSIONES

Pudimos verificar en la práctica tal como habíamos mencionado anteriormente, que el tiempo de cómputo del algoritmo depende solamente de la cantidad de edificios, y no de la forma en la que estos estén.