Tarea 3: Closest pair of points

Profesores: Benjamín Bustos
Gonzalo Navarro
Auxiliares: Diego Salas
Asunción Gómez

1 Introducción

En esta tarea se estudiará el problema de encontrar el par de puntos más cercano, es decir, con la distancia más pequeña entre ellos. Para esto, se estudiarán tres algoritmos: dos de ellos determinísticos y uno aleatorizado.

El primero de los algoritmos determinísticos consiste en utilizar la técnica "dividir para reinar" y el segundo "sweep line". Ambos algoritmos tienen complejidad $O(n \log n)$. Finalmente, el tercero corresponde a un algoritmo aleatorizado, el cual toma tiempo $O(n)$ esperado. El objetivo es ver cómo se comportan uno de los algoritmos determinísticos y el aleatorizado con diferentes inputs.

2 Algoritmos

Dado n puntos en el plano, encontrar el par de puntos con menor distancia euclideana.

2.1 Divide and conquer

El siguiente algoritmo consiste en ir dividiendo el conjunto de puntos en dos partes iguales e ir resolviendo el problema de manera recursiva. Vamos encontrando el par más cercano en cada mitad y luego el par más cercano que esté en ambas mitades. El algoritmo paso a paso está descrito de la siguiente forma:

1. Ordenar los puntos según su coordenada x.
2. Si el tamaño es 0 o 1, retornar infinito. Si es 2, devolver la distancia entre ellos. Sino, dividir el conjunto en dos partes iguales usando una recta divisoria.
3. Solucionar el problema de forma recursiva para ambas partes y obtenemos $Lmin$ y $Rmin$.
4. Calcular el mínimo entre ambas soluciones: $LRmin$.
5. Combinación de puntos: seleccionamos aquellos pares que tienen un punto en cada mitad y con una distancia menor a $LRmin$ de la recta divisoria entre cada mitad. Como están ordenados los puntos, se seleccionan aquellos del final de la parte izquierda y del principio de la parte derecha.
6. Retornar finalmente el par más cercano.

Pueden ver mayor información del algoritmo aquí.

2.2 Sweep line

Este algoritmo usa la técnica de barrido "sweep line" para ir revisando todo el plano y detenerse en aquellos lugares que nos interesan. Para esto, debemos hacer lo siguiente:

1. Ordenar los puntos por su coordenada x.
2. Vamos haciendo el barrido con una línea vertical de izquierda a derecha y vamos guardando la distancia d que corresponde a la menor distancia vista hasta el momento y el par de puntos.
3. Si encontramos un punto más cercano, actualizamos el valor d y el par de puntos.

Para más información sobre el algoritmo, pueden visitar este sitio.

2.3 Aleatorizado

Para el caso aleatorizado, se tiene el siguente algoritmo:

- Se seleccionan n pares de puntos al azar (permitiendo repeticiones), y se calcula la distancia mínima d entre los puntos de cada par.
- Se divide el plano en una grilla de cuadrillas de $d \times d$, y se agrupan los puntos en función de la grilla a la que pertenecen (para hacer esto de manera eficiente, se debe utilizar una Hash Table implementada por ustedes, donde las llaves son las coordenadas del cuadrado, y en ella se guardan los puntos contenidos en el).
- Para cada punto, se compara la distancia entre este, y todos los puntos pertenecientes a su misma casilla, y a las 8 casillas vecinas, la idea es que, si el d es suficientemente bueno, los puntos con los que se compara cada iteración deben ser pocos.
- Se entrega el par de puntos con la menor distancia calculada en el paso anterior.

Esta solución entrega siempre la respuesta correcta, sin embargo, el tiempo de ejecución puede variar, esto depende de que tan bueno es el d que tomamos (en general, mientras más cercano a la solución del problema, mejor). Para la Hash Table, se deben investigar distintas formas de implementar el hash. En general, al hashear, se utiliza el operador módulo, que resulta ser bastante lento en comparación al resto de operaciones, explore distintas formas de implementarlo, estudiando la rapidez de estas, como mínimo se debe implementar la tabla con los siguientes métodos de hashing:

- Hashing Universal (Sección 6.4.1 del apunte).
- El hash mencionado en **Funciones más rápidas** (final de la sección 6.4.1 del apunte).

- Se debe implementar un hash que utilice los primos de Mersenne con el fin de aumentar la velocidad del cálculo, investigue como se consigue esto.
- Implementear otros métodos relevantes significará un puntaje adicional.

3 Objetivos

Para esta tarea, se deberá implementar uno de los primeros dos algoritmos y el tercero para resolver el problema de "closest pair of points":

1. Divide and conquer
2. Sweep line
3. Aleatorizado

Luego, queremos evaluar el tiempo en cada algoritmo y ver cómo se comportan con un mismo set de puntos.

4 Experimentación

Para probar los algoritmos, deberá entregarle un set de tamaño n de puntos pertenecientes al rango $[0, 1)$ x $[0, 1)$. Pruebe con n de 5 a 50 millones con un paso de 5 millones. Para cada arreglo de tamaño n, deberá ejecutar cada algoritmo una cantidad considerable de veces (> 100).
Luego, deberá generar un histograma con los tiempos de ejecución de cada algoritmo. Debe utilizar el mismo arreglo en las repeticiones y en ambos algoritmos para poder compararlos. También, deben graficar cómo crecen las medias en función del tamaño del input. Por último, analice el tiempo promedio para varios inputs aleatorios y muestre que el costo de tiempo en sus implementaciones se asemejan a lo conocido por la teoría.

5 Instrucciones

- No es necesario implementar ustedes mismos el algoritmo determinístico, lo pueden obtener de Internet. El método probabilístico, por el contrario, sí deben implementarlo ustedes.
- Tendrán puntos extras si buscan otros métodos relevantes para la implementación del hash.
- Se deben probar, como mínimo, los 3 métodos de hashing mencionados para el algoritmo aleatorizado.

6 Entrega

- La tarea puede realizarse en grupos de a lo más 3 personas.
- Para la implementación puede utilizar C, C++ o Java.
- Para el informe se recomienda utilizar Latex.
- Escriba un informe claro y conciso, la entrega debe contar con el código de su implementación y todas las indicaciones necesarias para su ejecución explicadas de forma clara.
- Lea el material docente sobre cómo escribir un buen informe!
- La nota del informe se calculará de la forma:
  - Introducción: 0.8 ptos.
  - Desarrollo: 0.8 ptos.
  - Resultados: 2.4 ptos.
  - Análisis: 1.2 ptos.
  - Conclusión: 0.8 ptos.
- La nota final de la tarea es el promedio del informe y del código.
- Investigar, implementar, experimentar y documentar sobre otros métodos de hashing, significará una bonificación de 0.5 puntos (con tope 0.5 puntos).