Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Tarea 1 - Intersección de segmentos ortogonales

Profesor: Pablo Barceló
Auxiliar: Manuel Cáceres
Ayudantes: Jaime Salas
Claudio Torres

1 Introducción

El problema de "Intersección de segmentos ortogonales" consiste en, dado un conjunto de segmentos de línea horizontales y verticales, encontrar y reportar todos los puntos de intersección existentes entre ellos. Estamos interesados en resolver instancias en las cuales el problema es tan grande que no puede ser resuelto en la memoria interna del computador, para lo cual usaremos un algoritmo de ordenación en memoria externa y el algoritmo "Distribution Sweep" que logra su objetivo al mezclar los paradigmas de "distribution", usado también para ordenar, y "sweeping", o barrido, utilizado en varios problemas de computación geométrica.

Se espera que se implementen los algoritmos, se realicen los experimentos correspondientes y se entregue un informe que indique claramente los siguientes puntos:

1. Las hipótesis escogidas antes de realizar los experimentos.
2. El diseño experimental, incluyendo los detalles de la implementación de los algoritmos, la generación de instancias y las medidas de rendimiento utilizadas.
3. La presentación de los resultados en forma de una descripción textual, tablas y/o gráficos.
4. El análisis e interpretación de los resultados.

2 Los Algoritmos

2.1 Ordenación

2.1.1 MergeSort

El algoritmo MergeSort para memoria externa visto en cátedras consiste de dos fases:

- Una primera fase de *creación de "runs"*, en la cual se crean $\Theta(n/m)$ "runs" ordenados en memoria interna y puestos nuevamente en memoria externa.
- Una segunda fase de *merge*, donde en cada paso se mezclan $\Theta(m)$ "runs" de igual tamaño poniendo el resultado nuevamente en memoria externa. Para lograr este objetivo, se manejan $\Theta(m)$ buffers de input de los "runs" y un buffer de output (todos de tamaño B).

Este algoritmo alcanza la cota inferior de ordenación en memoria externa $\Theta(n \log_m n)$.

2.1.2 DistributionSort

Por otro lado el algoritmo "DistributionSort" es un algoritmo recursivo que usa $S-1$ pivotes para particionar los datos en S "buckets" disjuntos, es decir, escoge los pivotes $e_1, \ldots, e_{S-1}$ de los datos y distribuye los elementos en estos $|\{(-\infty, e_1),\ [e_1, e_2),\ \ldots, [e_{S-2}, e_{S-1}),\ [e_{S-1}, \infty)\}| = S$ "buckets". Finalmente hace lo mismo recursivamente en cada uno de estos "buckets".
De manera más detallada el algoritmo:

- Escoge $S-1$ pivotes, lo que genera S "buckets". Para encontrar estos pivotes usaremos el siguiente procedimiento probabilístico que asegura que cada "bucket" tendrá $\mathcal{O}(N/S)$ elementos:
  1. Tomamos una muestra aleatoria de $S \log S$ elementos de los datos
  2. Ordenamos estos elementos
  3. Los pivotes serán los $\log S$-ésimos elementos de este orden
- Poner cada dato del input en su "bucket" correspondiente. Para lograr esto tendremos un buffer de input y $\Theta(\min\{m, n/m\})$ buffers de output (todos de tamaño B).[1]
- Recursivamente aplicar el algoritmo en cada uno de los "bucket" hasta que el tamaño de los "buckets" sea $\leq M$ en cuyo caso se ordena en memoria interna.

Este algoritmo también alcanza la cota inferior de ordenación en memoria externa pues en cada nivel de la recursión el tamaño de los "buckets" se reduce en un factor de S, por lo que la profundidad de la recursión es $\log_S n$. Por otro lado, en cada nivel se escanean todos los elementos del input, por lo que su complejidad es $\Theta(n \log_S n) = \Theta(n \log_m n)$.

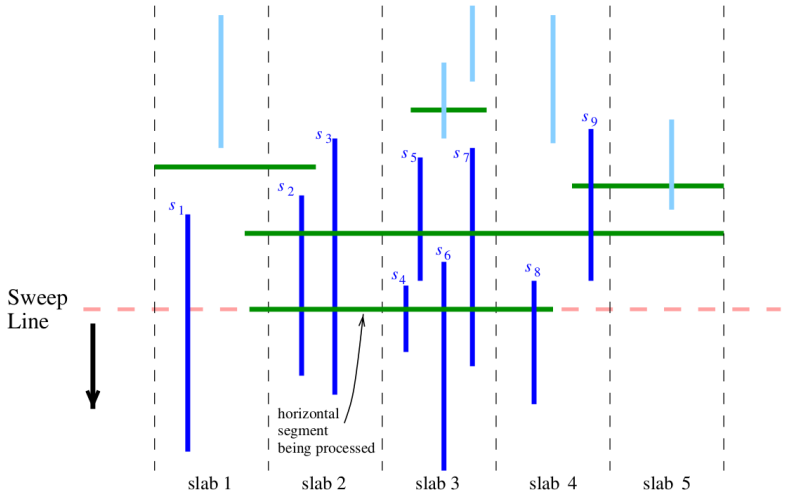
2.2 Distribution Sweep

Distribution Sweep es un algoritmo de computación geométrica para memoria externa que resuelve el problema de encontrar todas las intersecciones entre segmentos de línea ortogonales. Para esto divide la dimensión horizontal en "slabs" (paradigma distributivo) y recorre/barre los datos de arriba hacia abajo con una "Sweep Line"(paradigma de barrido).
El algoritmo queda descrito por el siguiente procedimiento:

- Generar lista X de segmentos ordenados por su coordenada x.
- Generar lista Y de segmentos ordenados por su coordenada y. Sea cuidadoso en este paso, preocúpese que para segmentos de igual coordenada aparezcan primero los segmentos verticales y luego los horizontales.[2]

[1] Notar que esto significa que $S = \min\{m, n/m\}$
[2] Esto último es para garantizar un correcto funcionamiento del algoritmo.

- Usar la lista X para dividir el input en $k = \Theta(m)$ "slabs" verticales y por cada $slab_i$ inicializar una lista A_i de segmentos verticales activos.
- Usar la lista Y para recorrer los segmentos de arriba hacia abajo, emulando un barrido horizontal de los datos, con la denominada "Sweep Line", de modo que:
  - Si escaneo el inicio de un segmento vertical, encuentro el $slab_i$ en el cual se encuentra y agrego el segmento a la lista A_i correspondiente.[3]
  - Si escaneo el fin de un segmento vertical, lo elimino de la lista A_i correspondiente en la cual se encuentra.[4]
  - Si escaneo un segmento horizontal, encuentro los "slabs" en los que se encuentran los extremos del segmento, $slab_i$ y $slab_j$ (con $i \leq j$) y reporto las intersecciones con todos los segmentos que se encuentran en las listas $A_{i+1}, \ldots, A_{j-1}$ (si existe una intersección en los "slabs" de los extremos, estos serán reportados en los siguientes niveles de la recursión).
  - Repito recursivamente el algoritmo en cada uno de los "slabs" hasta que el tamaño de estos quepa completamente en memoria interna.



En la figura anterior los segmentos verticales en las listas activas están oscurecidos($s_1, \ldots, s_9$), los segmentos s_5 y s_9 no están realmente activos, pero aún no se han eliminado de las listas A_5 y A_9. La "Sweep Line" recién avanzó a una línea horizontal que cubre completamente los "slabs" 2 y 3, por lo que son escaneados todos los segmentos de las listas A_2 y A_3, es decir , $s_2, s_3, s_4, s_5, s_6, s_7$ y

[3]Cada segmento se debe manejar como una pila que mantiene en memoria interna $\leq B$ segmentos y pone en memoria externa cuando se llena.

[4]Las eliminaciones de los segmentos de las listas se debe hacer de forma "lazy", es decir, cuando se procese la lista en busca de intersecciones.

son reportados todas las intersecciones generadas por estos segmentos, excepto s_5 que es eliminado de su lista activa. Las posibles intersecciones que se generan con los "slabs" de los extremos son tratados por los siguientes niveles de la recursión en donde eventualmente se descubre la intersección son s_8.

Si el input son N segmentos de líneas el algoritmo anterior es de complejidad $\mathcal{O}(n \log_m n)$ al igual que la de ordenar.

3 Implementación

Implemente alguna de las versiones de ordenación en memoria externa óptimas en complejidad (como se mostró puede ser MergeSort o DistributionSort) adaptada para ordenar segmentos de líneas según sus coordenadas x o y y utilice esto para implementar el algoritmo "Distribution Sweep".
Asuma que el input viene en la forma de un archivo desordenado de segmentos de línea separados por comas, cada segmento de línea se representa como una tupla (x_1, y_1, x_2, y_2), donde (x_1, y_1) y (x_2, y_2) son los puntos de los extremos del segmento. Las coordenadas de las intersecciones de los segmentos deben mostrarse una única vez en la salida estándar en forma de pares ordenados (x, y).

4 Experimentos

Escoja los parámetros M y B de acuerdo a las características de su máquina y en pro de la obtención de datos significativos. Documente los parámetros anteriores características de su máquina, el sistema operativo, lenguaje, y compilador utilizados, RAM y características del disco duro.
En los experimentos se pide comparar los resultados del algoritmo dependiendo de los siguientes parámetros:

- Tamaño del input, diferentes cantidades de segmentos $N \in \{2^9, 2^{10}, \ldots, 2^{21}\}$.
- Distribución en la coordenada x de los segmentos verticales, considere las siguientes: uniforme, normal.[5]Las demás coordenadas tómelas uniformemente en el espacio.
- Balance entre la cantidad de segmentos verticales y horizontales, considere tener αN segmentos horizontales y $(1 - \alpha)N$ verticales con $\alpha \in \{0.25, 0.5, 0.75\}$.

Para lo anterior considere las siguientes mediciones:

- Tiempo de ejecución y número de operaciones I/O. Separe las mediciones de ordenación de las del resto del algoritmo.
- Número de intersecciones encontradas.

[5]Indique el rango, la media y la desviación de las distribuciones consideradas.

5 Entrega de la Tarea

- La tarea puede realizarse en grupos de a lo más 2 personas.
- No se permiten atrasos.
- Para la implementación puede utilizar C, C++ o Java. Para el informe se recomienda utilizar LaTeX.
- Siga buenas prácticas (good coding practices) en sus implementaciones.
- Escriba un informe claro y conciso. Las ponderaciones del informe y la implementación en su nota final son las mismas. No olvide adjuntar un README con instrucciones de compilación y ejecución de su código. Tampoco olvide documentar las características de la máquina donde fueron realizados los experimentos.
- Tenga en cuenta las sugerencias realizadas en la primera clase auxiliar y en los archivos subidos a U-Cursos sobre la forma de realizar y presentar experimentos.
- La entrega será a través de U-Cursos y deberá incluir el informe junto con el código fuente de la implementación (y todas las indicaciones necesarias para su ejecución).

6 Links

- Paper donde se explica algoritmo de "Distribution Sweep":
  http://www.ics.uci.edu/ goodrich/pubs/ecg.pdf