UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Autor: Adrián Aguilera Moreno

Algoritmos II

Resumenes.

0.1 Un algoritmo de barrido de línea para agrupamiento espacial.

Obejetivo: Agrupar un conjunto de puntos P en el plano de manera jerárquica, por medio de la distancia que los separa entre sí.

Desarrollo: El algoritmo consiste en, dadas dos líneas s_1 y s_2, y P ordenado. Entonces barremos P con ambas líneas y dejando una separación d entre s_1 y s_2. Diremos que el espacio entre s_1 y s_2 será llamado "manga".

En la primer iteración, agrupamos los puntos a una cercanía $\frac{d}{2}$ y coloreamos bajo un mismo color los puntos que han sido agrupados en un mismo punto. En esta iteración creamos el FRENTE DE AVANCE (AF), que esta dado por la trayectoria entre los puntos coloreados a este punto.

En la i-ésima iteración, sacamos los puntos de AF que no se encuentren dentro de la manga y reconstruimos AF con los puntos dentro de la manga en este momento. Obtengamos las proyecciones, a AF, de los puntos que vayan entrando a la manga, digamos $P(v)$. Así, asignamos un grupo a los puntos tales que su proyección esté más cerca, por a lo más $\frac{d}{2}$, a un punto con color asignado en AF. Si la proyección no es cercana a nimgún punto en AF y el propio punto no es cercano a ningún punto con color, entonces creamos un nuevo grupo. Si en algún momento dos grupos se ven unidos por un nuevo punto (que esta cerca de puntos con colores distintos), entonces le asignamos color al grupo con menor cantidad de elementos.
Obs. Nótese que $P(v)$ podría no tener imagen cuando no existe una proyección en vértical con AF.

Análisis de complejidad.

1. Ordenar P nos toma $\mathcal{O}(n \log_2 n)$.
2. Mover la manga y realizar las iteraciones nos toma $\mathcal{O}(n \log n)$. Esto gracias a que d es constante respecto a la altura (Y) del conjunto de puntos.
3. Recolorear grupos nos toma c, y a lo más $\mathcal{O}(c \cdot n)$.

Concluimos una complejidad de

$$\therefore \mathcal{O}(n \log_2 n) + \mathcal{O}(n \log n) + \mathcal{O}(c \cdot n) = \mathcal{O}(n \log n).$$

0.2 Un algoritmo de barrido de línea y su aplicación en espirales interiores (pocketing).

Obejetivo: Construcción de espirales internas a un polígono dado.

Desarrollo: Una alternaiva mencionada es usar los diagramas de Voronoi. Sin embargo, la estrategia seguida es:

1. Ordenar nuestro arreglo de puntos.
2. Crear nuestro polígono.

   En este punto nos interesa encontrar polígonos válidos, esto lo logramos búscando en el orden de las manecillas del reloj: empezamos en un punto y recorremos hasta toparnos con que no podemos avanzar y eliminamos esa arista (pues no será parte de nuestro polígono admisible).
3. Eliminar colineales.

   No tiene sentido tener puntos colineales en el polígono, pues construiremos las espirales internas a partir de estos (todos) y solo nos interesa tener segmentos, sin importar si hay más de dos puntos en ellos.
4. Encontrar un extremo (caso partícular del extremo izquierdo).

   Esto lo usamos para empezar el barrido y lo podemos encontrar en tiempo constante si ordenamos nuestro arreglo o en tiempo líneal en otro caso.
5. Barrido de línea.

   Nos toma $\mathcal{O}(\log n)$ por punto, pues empezamos a contruir las espirales.
6. Encontrar intersecciones entre monótonas.

   Las intersecciones serán parte de la construcción de la espiral.
7. Obtener el arreglo de intersecciones.

   Lo obtenemos del paso anterior en $\mathcal{O}(n \log n)$.

Obs. El algoritmos consiste en ir acotando la súperficie interna del polígono e ir formando las espirales en estas áreas. Debemos saber cuándo parar, esto es cuándo acotamos la sección llamada isla, en este punto el polígno debe tener espirales internas y tener estas construcciones terminadas.

Análisis de complejidad. La complejidad esta contenida en

$$\therefore \mathcal{O}(n) + \mathcal{O}(n \log n) = \mathcal{O}(n \log n).$$

0.3 Localización de puntos planos mediante árboles de búsqueda persistentes.

Obejetivo: Localización de puntos.

Desarrollo: Segementamos nuestro espacio de búsqueda, por medio de un barrido de línea, en franjas. Ahora necesitamos guardar la información de una manera no costosa y luego realizar una búsuqeda en estas franjas. Nosotros emplearemos un árbol rojo-negro y por tanto bastará descender por el árbol para encontrar el espacio dónde se localiza nuestro punto. Para poder tener nuestras frangas almacenadas de manera eficiente haremos persistente nuestro árbol y con "persistente" nos referimos a que queremos que el árbol guarde de alguna manera los estados del mismo. ¿Cómos hacemos lo anterios? Fusionaremos dos técnicas, estas son

1. Técnica 1. Realizar una copia de nodo por cada eliminación o reordenamiento, con sus respectivas referencias.
2. Técnica 2. Marcar referencias "vivas" (aquellas intactas) y "muertas" (aquellas cuáles nodos han sido borrado, al menos uno de ellos).

con la fusión de ambas técnicas podemos garantizar que tendremos los estados por los que pasa el árbol durante su construcción, sin necesidad de guardar estado por estado (este caso nos llevaría a un orden cuadrático).

Análisis de complejidad. En general, esto nos toma $\mathcal{O}(k + \log n)$ por nodo, donde k es la cantidad de punteros a ligar. Este valor es amortizado. Así, mantener la persistencia nos toma

$$\therefore \mathcal{O}(n \cdot k + n \log n).$$

Obs. La complejidad es solo la de construir nuestro árbol.

0.4 Un problema simple de caminos alternos.

Obejetivo: Resolver el problema de Putman para una cadena círcular partícular y dar el conjunto de posibles caminos en una gráfica que se pueden tomar para llegar a la palabra dada, en caso de ser admisible. Esto es, ¿de cuántas maneras podemos formar la cadena siempre que sea admisible?.

Desarrollo: Nos basamos en las reglas de construcción:

- La palabra vacía es válida.
- abW, dónde W es no terminal (en adelante se omite la explicación).
- aWb.
- bWa.
- Wab.

ahora, debemos contar las posiciones en la palabra círcular c. Enumeremos carácter a carácter con s_i y cuán queramos acceder a un carácter $j > |c|$, entonces buscaremos el carácter en la posición j mod c. Utilizaremos la notación $W(i, j)$ tal que iniciaremos en el carácter i y elijiremos los siguientes $k \cdot j$ carácteres de manera círcular y donde k es el número de carácteres de la palabra por nivel de jerárquia.

¿Qué son los niveles de jerárquia? Les llamo partícularmente a los niveles que se forman en nuestra gráfica, iniciando con la vacía, luego con $W(i, 1), W(i, 2), \cdots$

Ya definidos los niveles, entonces tenemos como "fuente" a la palabra vacía y como "sumidero" a la palabra círcular objetivo, así basta bajar desde la fuente al sumidero en forma de aristas orientadas para saber cuántas maneras hay de formar la palabra. Las palabras relacionadas por una arista serán aquellas que procedan por medio de una regla de producción (de las ya mencionadas).

Si la palabra círcular es no-admisible, entonces no hay manera de llegar por un camino desde la fuente al sumidero.

Análisis de complejidad. Basta ver que en el pero de los casos es necesario construir la gráfica que por cada nodo inicial (donde nace la arista orientada) tenga relación con todos los nodos en el siguiente nivel, en general esto es un orden cuadrático y por tanto la complejidad de nuestro algoritmo es $\mathcal{O}(n^2)$.

0.5 Triangulaciones de Graham y hamiltonianas con centro.

Obejetivo: Triangular una nube de puntos de tal manera que podamos encontrar caminos hamiltonianos que entren por una arista de un triángulo[1] y salgan por otra, finalmente debemos encontrar un ciclo hamiltoniano.

Desarrollo: Dividamos este tema en dos subtemas, estos son:

1. *Caminos hamiltonianos en triangulaciones de Graham.* Definamos un Triángulo separador T_s como un triángulo del cuál sus aristas son parte de aristas en la triángulación y, además, existe al menos un punto contenido por T_s.

   Al tener la triangulación T de Graham sabemos que todo triángulo separador tiene como vértice a v_0 desde dónde se originó la triangulación. Entonces, encontrar la gráfica dual es sencillo desde este punto, pero ¿Qué es la gráfica dual? Es la gráfica que se forma a partir de los centros de triángulos generados en la triangulación y tales que son adyacentes si y sólo si existe una única arista entre ellos[2]. La gráfica dual no es una trayectoria o ábol, entonces nos quedaremos con aristas a manera que cada arista sea de corte[3]

   El árbol generado anteriormente es un camino hamiltoniano tal que pasa por todos los triángulos generados por la triangulación de graham.

2. *Caminos hamiltonianos en triangulaciones basadas en un centro.* A partir de un centro generamos triángulos que contengan una arista en el cierre convexo y que preserven la nube de puntos como una gráfica plana. Queremos buscar trayerctorias que pasen por todos los puntos en cada triángulo y que se conecte en forma de ciclo hamiltoniano. Las trayecotorias se conectan saliendo y entrando por los triángulos generados con el punto central y por tanto tenemos un ciclo hamiloniano.

[1] No cualquier triángulo
[2] Sólo son separados por una arista.
[3] Arista de corte. Sea e una arista tal que al quitarla la gráfica se vuelve inconexa, entonces e es de corte.