MTH6412B

Implementación de algoritmos de investigación de operaciones

Árboles de expansión mínimos II

Dominique Orban
dominique.orban@polymtl.ca

Matemáticas e Ingeniería Industrial
École Polytechnique de Montréal

Dos heurísticas

Técnicamente, la unión de dos componentes relacionados podría crear largas cadenas.

Para acelerar las operaciones en componentes conectados, proponemos las siguientes dos heurísticas: 1. unión vía rango; 2. compresión de trayectoria.

El objetivo es encontrar la raíz de un árbol en un bosque de conjuntos disjuntos más rápido (en un tiempo casi lineal).

Heurística 1: unión vía rango

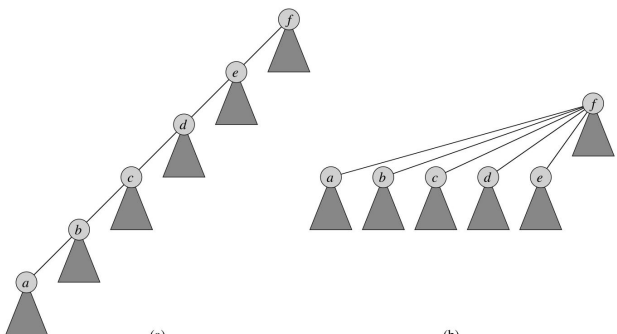
Cada nodo en un bosque de conjuntos disjuntos tiene un atributo de rango cuya función es dar un límite superior a la altura de este nodo en su árbol.

1. Inicialmente, cada nodo recibe el rango
0; 2. al unir dos conjuntos, se comparan los rangos de las raíces:
   - 2.1 si los rangos son distintos, la raíz con el rango más alto se convierte en el padre de la raíz de menor rango y sus rangos no cambian;
   - 2.2 si los rangos son iguales, una de las dos raíces se convierte en el padre de la otra y su rango aumenta en 1.

Heurística 2: compresión de ruta

Cuando buscamos la raíz de un componente conectado, nos movemos de un nodo a su padre. Los nodos y arcos así tomados forman el camino de búsqueda.

La compresión de ruta consiste en hacer que la raíz sea el padre directo de cada uno de los nodos en la ruta de búsqueda.



Segunda parte de este laboratorio.

1. Modificar la estructura de datos del conjunto disjunto para implementar la unión a través del rango;
2. Muestre que el rango de un nodo siempre será menor que |S| ÿ 1. Próximo espectáculo que este rango de hecho siempre será más bajo que blog2 (|S|)c;
3. modificar el procedimiento de subir a la raíz para implementar la compresión caminos.

algoritmo de Prim

El algoritmo de Prim acumula aristas de G = (S,A) en un solo subárbol de expansión mínima de G y este subárbol crece a lo largo de las iteraciones. Cualquier vértice que no sea parte de este subárbol se aísla.

algoritmo de Prim

El algoritmo de Prim acumula aristas de G = (S,A) en un solo subárbol de expansión mínima de G y este subárbol crece a lo largo de las iteraciones. Cualquier vértice que no sea parte de este subárbol se aísla.

En cada paso, agregamos un ligero borde entre este subárbol y un vértice aislado. Por el corolario del laboratorio 3, el subárbol así obtenido sigue siendo un subárbol de expansión mínima.

algoritmo de Prim

El algoritmo de Prim acumula aristas de G = (S,A) en un solo subárbol de expansión mínima de G y este subárbol crece a lo largo de las iteraciones. Cualquier vértice que no sea parte de este subárbol se aísla.

En cada paso, agregamos un ligero borde entre este subárbol y un vértice aislado. Por el corolario del laboratorio 3, el subárbol así obtenido sigue siendo un subárbol de expansión mínima.

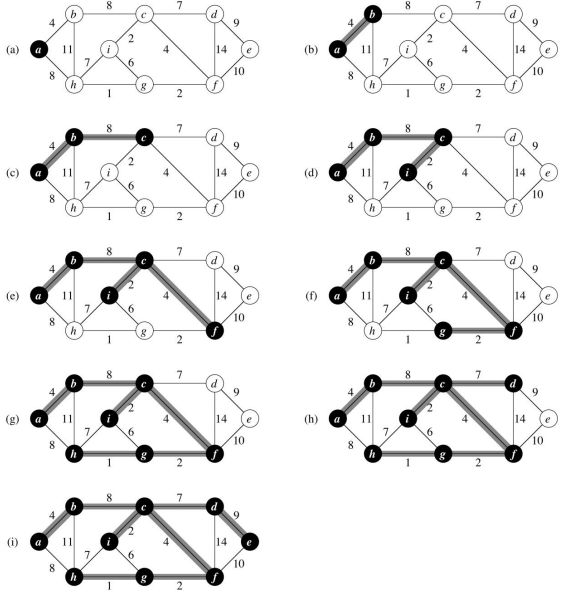
En cada paso, los nodos que forman parte del subárbol determinan un corte. El algoritmo elige un borde claro que cruza este corte.

Ideas de implementación

1. Cada nodo tiene un atributo **min_weight** que da el peso del borde de peso mínimo que conecta este nodo con el subárbol. Inicialmente, **min_weight = Inf** (es decir, +ÿ); 2. Inicialmente, el padre de cada nodo es **nada ;** 3. el algoritmo comienza en un nodo de origen elegido por el usuario y el atributo

   min_weight de s es 0;
4. una cola de prioridad contiene todos los nodos que aún no se han agregado a tree y **min_weight** dan prioridad;
5. Cada vez que se conecta un nodo al árbol, se deben actualizar los atributos **min_weight** y **parent** de aquellos que aún no se han conectado.

Ilustración del algoritmo de Prim



Tercera parte de este laboratorio

1. Implementar el algoritmo de Prim y probarlo en el ejemplo de las notas del curso; 2. Pruebe su implementación en varias instancias del TSP simétrico.