Andrea Reyes

20190265

Path algorithms

1. Shortest Distance in DAGs:

El DAG es un algoritmo más rápido que corriendo en tiempo lineal puede encontrar los caminos más cortos desde un nodo fuente dados todos los demás vértices de un grafo. Con el DAG no tenemos que preocuparnos por los ciclos negativos porque es acílico.

* Pueden ser fácilmente clasificados topológicamente.
* Primero se debe clasificar topológicamente el DAG
* Después se establece la distancia a la fuente en 0 e infinito a todos los demás vértices
* Luego para cada vértice de la lista pasamos por todos sus vecinos y buscamos el camino más corto.

Pseudocódigo:

1. Topologically sort G into L;

2. Set the distance to the source to 0;

3. Set the distances to all other vertices to infinity;

4. For each vertex u in L

5. - Walk through all neighbors v of u;

6. - If dist(v) > dist(u) + w(u, v)

7. - Set dist(v) <- dist(u) + w(u, v);

1. Bidirectional search:

Es un algoritmo que encuentra un camino más corto desde un vértice inicial hasta un vértice objetivo. Ejecuta dos búsquedas simultaneas:

* Una hacia adelante desde el estado inicial
* Y otra hacia atrás desde el objetivo, deteniéndose cuando ambas se encuentran.

1. MST (Prim´s minimum spanning tree):

Un árbol de expansión significa que todos los vértices deben estar conectados.

* Crear un conjunto de mstSet que lleve la cuenta de los vértices ya incluidos en MST.
* Asignar un valor clave a todos los vértices del grafo de entrada. Inicializar todos los key values como infinito. Asignar el key value como 0 para el primer vértice, de modo que sea elegido primero.
* Escoge un vértice que no esté en mstSet y que tenga un valor mínimo.
* Incluye el vértice en el mstSet.
* Actualiza el key values de todos los vértices adyacentes del vértice. Para actualizar los key values, itera a través de todos los vértices adyacentes.

Pseudocódigo:

Prim()

S = new empty set

for i = 1 to n

d[i] = inf

while S.size() < n

x = inf

v = -1

for each i in V - S // V is the set of vertices

if x >= d[v]

then x = d[v], v = i

d[v] = 0

S.insert(v)

for each u in adj[v]

do d[u] = min(d[u], w(v,u))

Aplicaciones:

* Diseños de red (teléfono, electricidad, cable, etc.)

1. Single-source shortest path:

Encuentra los caminos más cortos desde un vértice de origen específico a todos los demás vértices en un grafo ponderado y dirigido.