ALGORÍTMICA
PRÁCTICA 3: ALGORITMOS VORACES

UNIVERSIDAD DE GRANADA

Grupo MT™

Mario Soriano Morón
Miguel Torres Alonso
José Teodosio Lorente Vallecillos

Índice

1. Ejercicio 1

El problema lo modelamos como un grafo, donde cada vértice es un sensor y cada arista un enlace entre dos sensores. El grafo resultante cumple con las propiedades de ser no dirigido, conexo y ponderado con pesos no negativos. El objetivo consiste en encontrar un subconjunto de aristas que unan todos los vértices del grafo y cuya suma de pesos sea mínima. Por tanto, nos encontramos con que puede ser resuelto como instancia del Árbol Generador Minimal. Sus componentes de diseño serían:

1.1 Diseño de componentes.

- Lista de candidatos: Los sensores a los que se puede enviar información.
- Lista de candidatos usados: Los sensores a los que ya se ha enviado la información.
- Criterio de selección: Se seleccionará el sensor n_j no utilizado tal que el tiempo de transmisión de información entre n_j con otro ya utilizado n_i, sea mínimo.
- Criterio de factibilidad: Un sensor se insertará en la solución si al insertarlo no produce ciclos.
- Función solución: Cuando el número de sensores seleccionados es igual al total incluyendo al servidor central.
- Función objetivo: Minimizar el tiempo de transmisión de información de nodos al servidor central.

1.2 Diseño del algoritmo.

```
Algoritmo S=Greedy(G = <V,A>: Grafo de entrada)
    S = {} # Conjunto vacío
    B = {elemento cualquiera de V} // Candidatos usados
    Mientras (|B| != |V|), hacer:
        x = Seleccionar un nodo v tal que existe una arista a= (b, v),
        de peso mínimo, que una un nodo b en B y otro nodo v en V \ B
        S= S U {a}
        B= B U {v}
    Fin-Mientras
    Devolver S
```

1.3 Estudio de optimalidad (demostración o contraejemplo).

El algoritmo utilizado es una instancia del problema del Árbol Generador Minimal, en concreto, el algoritmo de Prim. Demostraremos su optimalidad por reducción al absurdo:

Imaginemos un caso trivial en el cual disponemos de un grafo completo compuesto de dos nodos, entonces la única arista que une ambos nodos es elegida por el algoritmo, dando así la solución óptima. En un caso base de dos subgrafos donde uno contiene 2 nodos unidos por una arista previamente escogida, se deberá escoger otra arista que una los dos subgrafos, donde el algoritmo escogerá la arista de menor peso y por tanto la solución es óptima. Si añadimos otro subgrafo cualquiera el cual se debe unir al subgrafo que teníamos, y suponiendo 2 posibles aristas para unirlos, el algoritmo escogerá la de menor peso. Es por esto que sabemos que el algoritmo es óptimo, puesto que para que no lo fuera, debería existir otra arista en algún subgrafo que hiciese que la suma de los pesos de las aristas seleccionadas en ese caso fuese menor que la suma actual, pero si esta arista existiese, el algoritmo la habría seleccionado con antelación, pero como no ha sido así, la suma actual es la mínima posible.

1.4 Ejemplo paso a paso de la explicación del funcionamiento del algoritmo para una instancia pequeña propuesta por el estudiante.

Al demostrar la optimalidad por reducción al absurdo en el apartado anterior, queda explicado el ejemplo paso a paso para una instancia pequeña.

2. Ejercicio 2

El problema lo modelamos como un grafo, donde cada vértice es una gasolinera, salvo el primero y el último son el origen y el destino de la ruta, y cada arista un enlace entre dos gasolineras, entre origen/destino con una gasolinera ó entre origen y destino si se da el caso. El grafo resultante cumple con las propiedades de ser no dirigido, conexo y ponderado con pesos no negativos. El objetivo consiste en encontrar un subconjunto minimal de aristas que unan los mínimos vértices del grafo y cuya suma sea el mínimo número de paradas. Por tanto, nos encontramos con que puede ser resuelto como instancia del Árbol Generador Minimal. Sus componentes de diseño serían:

2.1 Diseño de componentes.

- Lista de candidatos: Las gasolineras en las que se puede repostar.
- Lista de candidatos usados: Las gasolineras en las que ya se ha repostado.
- Criterio de selección: Se seleccionará la gasolinera n_j no utilizada tal que el autobús pueda recorrer los siguientes kilómetros hasta la siguiente gasolinera, haciendo que n_j con otra ya utilizada n_i, sea máxima.
- Criterio de factibilidad: Una gasolinera se insertará en la solución si al insertarla no produce ciclos.
- Función solución: Cuando el número de gasolineras seleccionadas es igual al total de kilómetros entre origen y destino.
- Función objetivo: Minimizar el número de gasolineras en las que repostar haciendo máximos los kilómetros entre gasolineras(redondeado hacia arriba).

2.2 Diseño del algoritmo.

Algoritmo R=Greedy(G = <V,A>: Grafo de entrada)
R = {} # Conjunto vacío
G = {elemento cualquiera de V} // Candidatos usados
k = kilómetros máximos que puede recorrer con un repostaje
Mientras (|G| != |V|), hacer:
x = Seleccionar un nodo v tal que existe una arista a= (g, v), de kilómetros máxima, la cual se acerque lo máximo pero sin ser superior a k kilómetros a la vez que tiene en cuenta la siguiente posible parada, que une un nodo g en G y otro nodo v en V \ G
R= R U {a}
G= G U {v}
Fin-Mientras
Devolver S

2.3 Estudio de optimalidad (demostración o contraejemplo).

El algoritmo utilizado es una instancia del problema del Árbol Generador Minimal, en concreto, el algoritmo de Prim. Demostraremos su optimalidad por reducción al absurdo:

Imaginemos un caso trivial en el cual disponemos de un grafo completo compuesto de 4 nodos, entre los cuales uno es el origen otro el destino y el resto gasolineras, entonces la arista que es elegida por el algoritmo es la cual hace mínima las paradas, dando así la solución óptima. En un caso base de dos subgrafos donde uno contiene 2 nodos unidos por una arista previamente escogida, se deberá escoger otra arista que una los dos subgrafos, donde el algoritmo escogerá la arista de menor peso y por tanto la solución es óptima. Si añadimos otro nodo cualquiera el cual se debe unir al grafo que teníamos, y suponiendo 4 posibles aristas para unirlos, el algoritmo si fuera necesario para cumplir con el mínimo número de paradas escogerá la de mayor kilometros sin pasarse claramente de los k kilómetros de tope del tanque. Es por esto que sabemos que el algoritmo es óptimo, puesto que para que no lo fuera, debería existir otra arista en algún subgrafo que hiciese que la suma de los nodos seleccionados en ese caso fuese menor que la suma actual, pero si esta arista existiese, el algoritmo la habría seleccionado con antelación, pero como no ha sido así, la suma actual es la mínima posible.

2.4 Ejemplo paso a paso de la explicación del funcionamiento del algoritmo para una instancia pequeña propuesta por el estudiante.

Al demostrar la optimalidad por reducción al absurdo en el apartado anterior, queda explicado el ejemplo paso a paso para una instancia pequeña.

3. Ejercicio 3

El problema lo modelamos como un grafo, donde cada vértice es un contenedor con su respectivo peso. El grafo resultante cumple con las propiedades de ser no dirigido, conexo y ponderado con pesos no negativos. El objetivo consiste en encontrar un subconjunto máximo de vértices o nodos que maximicen la suma de los pesos de los contenedores a transportar en el barco. Por tanto, nos encontramos con que puede ser resuelto como instancia del Árbol Generador Minimal. Sus componentes de diseño serían:

3.1 Diseño de componentes.

- Lista de candidatos: Los contenedores a cargar.
- Lista de candidatos usados: Los contenedores que ya se han cargado.
- Criterio de selección: Se seleccionará el contenedor n_j no cargado tal que el buque cargue el máximo peso posible.
- Criterio de factibilidad: Un contenedor se insertará en la solución si al insertarla no produce ciclos, y se puede cargar todavía.

- Función solución: Cuando el número de contenedores es máximo y no quedan contenedores que se puedan cargar dentro del buque ya quede espacio o no.
- Función objetivo: Maximizar el número de contenedores que se puedan cargar dentro del buque.

3.2 Diseño del algoritmo.

Algoritmo B=Greedy(G = <V,A>: Grafo de entrada)
B = {} # Conjunto vacío
C = {elemento cualquiera de V} // Candidatos usados
k = peso maximo de la capacidad del buque
Mientras (|G| != |V|), hacer:
x = Seleccionar un nodo v tal que existe un vértice v= (c, a),
de peso máximo, el cual se acerque lo máximo pero sin ser superior a k peso a la vez,
B= B U {v}
C= C U {v}
Fin-Mientras
Devolver S

3.3 Estudio de optimalidad (demostración o contraejemplo).

3.4 Ejemplo paso a paso de la explicación del funcionamiento del algoritmo para una instancia pequeña propuesta por el estudiante.

Al demostrar la optimalidad por reducción al absurdo en el apartado anterior, queda explicado el ejemplo paso a paso para una instancia pequeña.