Trabajo Practico 1

De Rosa - Schapira - Guerrero

Primer Cuatrimestre 2017

Contents

1 Asignacion de residencias

1.1 Objetivo

Solucionar el problema de la asignación de residencias utilizando el algoritmo de Gale-Shapley de Matching Estable.

1.2 Conclusiones

1.2.1 Complejidad del algoritmo

Se prueba que el algoritmo tiene una complejidad $O(n(n-1)+1)$ pero dado que es un análisis de peor caso, en la práctica se obtuvo un rendimiento mejor. Si se hacen pruebas con cantidades de estudiantes (n) y de hospitales (m) iguales se obtienen los siguientes resultados:

Table 1: Tiempo de resolución del problema

n	m	t
10	10	0.6ms
100	100	30ms
500	500	2.5s
1000	1000	19.1s
3000	3000	8m 34s

Como se puede ver, estos tiempos representan valores mucho menores a los esperados por un algoritmo $O(n(n-1)+1)$.

1.2.2 Reduccion del problema de los casamientos

El problema de la asignación de residencias puede reducirse al problema de los casamientos en tiempo polinomial de la siguiente manera:
Teniendo en cuenta que se dispone de una lista H de listas de preferencias donde en cada indice se encuentra la lista de preferencias del hospital representado por dicho indice, una lista E del mismo estilo pero para los estudiantes y una lista Q que en cada indice tiene un entero que representa las vacantes del hospital de dicho indice, podemos adaptar esta informacion a la necesaria para ser procesada por el algoritmo de formacion de parejas de Gale Shapley adaptando la lista H a una nueva H' con la información de Q.
La manera de hacer esto sería:
H' contendrá las listas de preferencias de H pero repetidas Q[i] veces. Es decir, si H[0] tenia 3 vacantes, entonces en H', H[0], H[1] y H[2] tendrán las mismas preferencias y el hospital 0 pasará a ser representado por el conjunto 0, 1, 2.
Dicho esto, ahora habría que crear una lista E' donde, para cada lista de preferencias E[i], existirá una lista E'[i] donde para cada hospital k, se pondrá el conjunto k = p, q, r. En el ejemplo anterior, si E[i] = [0], entonces E'[i] = [0, 1, 2].

Con estas listas inicializadas, se corre Gale Shapley que devolverá parejas para cada hospital de H'. Es decir, P = [(H'i, E'k), ..., (H'n, E'j)]. Donde solo len(E) estudiantes serán no NULL. Ahora, como ya conocemos en que H'k se convirtió cada Hi, podemos agrupar los E' de las parejas devueltas por Gale Shapley para finalizar el algoritmo y resolver el problema de asignación de residencias. Es decir, con el ejemplo anterior, pondremos para el hospital 0, los estudiantes asignados por Gale Shapley a los hospitales H'[0], H'[1] y H'[2].
Si analizamos la complejidad, podemos ver lo siguiente:
Crear la lista H' es $O(m * q)$, siendo m la cantidad de estudiantes y q la sumatoria de todas las vacantes, pues hay q listas de m elementos (antes eran n pero ahora se repiten q veces).
Crear la lista E' es $O(m * q)$, siendo m y q los mismos de antes, pues hay m listas de q elementos.
Gale Shapley es polinomial.
Adaptar el resultado es $O(q)$, pues se debe recorrer una lista de q parejas, añadiendo el estudiante de cada pareja al hospital que corresponda según los cambios hechos al principio.

2 Puntos de Falla

2.1 Objetivo

Encontrar los puntos de fallas (puntos de articulación) de una red eléctrica mediante el algoritmo de Hopcroft y Tarjan.

2.2 Conclusiones

2.2.1 Funcionamiento del algoritmo

El algoritmo comienza tomando un vértice cualquiera del grafo, a partir del cual se realiza un recorrido dfs. Este genera un grafo del recorrido iniciado en el vértice elegido. En este grafo, será punto de articulación cada vértice que tenga dos o mas conocidos, es decir que tenga dos o mas hijos. Luego para cada vértice en el grafo de recorrido dfs, chequeamos la cantidad de vecinos que tiene. Si se cumple la condición, entonces es un punto de articulación. Por lo tanto el orden del algoritmo será $O(v + e)$.

2.2.2 Tiempo de ejecucion

Haciendo la prueba con una cantidad (v) de vértices obtenemos lo siguiente:

Como se puede observar, los tiempos aumentan linealmente a medida que aumenta la cantidad de vértices. Esto condice con los tiempos esperados del algoritmo.

Table 2: Tiempo de resolución del problema

v	t
10	0.25 ms
100	1.8 ms
1000	9.5 ms
10000	0.093 s
100000	0.99 s
1000000	10.34 s

3 Comunidades en Redes

3.1 Objetivo

Encontrar las componentes fuertemente conexas de las comunidades en las redes implementando el algoritmo de kosaraju.

3.2 Conclusiones

3.2.1 Funcionamiento del algoritmo

El algoritmo comienza realizando un recorrido dfs de cada uno de los vértices. Esto se realiza para ordenar los vértices en sentido decreciente de acuerdo a su tiempo de finalización entre todos los recorridos dfs. Una vez que se obtiene el orden de los vértices, se invierte el grafo, es decir que se invierten todas sus aristas. El tiempo que toma invertir el grafo es $O(v + e)$. Luego se vuelve a realizar un recorrido dfs para cada vértice iniciando por el vértice que tiene mayor tiempo de finalización. Cada grafo que devuelto por cada recorrido dfs es una componente fuertemente conexa. El orden del algoritmo será $O(v + e)$

3.2.2 Tiempo de ejecucion

Haciendo la prueba con una cantidad v de vértices y a de arístas obtenemos:

Table 3: Tiempo de resolución del problema

v	a	t
10	20	0.00084 s
100	250	0.048 s
1000	2500	3.008 s
10000	25000	322.01 s

Como podemos observar, los tiempos de resolución aumentan en medida cuadrática. Esto no concuerda con la predicción esperada, la cuál es lineal. Atribuimos esto a la eficiencia de la computadora en la que se hicieron las pruebas.

4 Comandos

Para correr los archivos, entrar en la carpeta correspondiente al algoritmo y ejecutar:
python matchingEstable.py
python puntosDeArticulacion.py
python componentesConexas.py

ATENCIÓN: Se deben agregar los archivos de prueba para puntos de articulacion y componentes conexas. No se enviaron debido al peso que soporta gmail.