Trabajo Practico 1

De Rosa - Schapira - Guerrero

Primer Cuatrimestre 2017

Contents

1 Asignacion de residencias

1.1 Objetivo

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

1.2 Conclusiones

1.2.1 Complejidad del algoritmo

Si bien el ciclo principal del algoritmo (sin tener en cuenta llamadas a funciones internas dentro de él) tiene un orden de complejidad $O(nk)$, donde k es la cantidad máxima de vacantes de cada hospital, el algoritmo tiene un orden de complejidad $O(n^2)$. Esto se da por la creación de la matriz de preferencias, pues es necesario al calcularla analizar la posición de cada estudiante en la lista que contiene a cada uno de los hospitales.

1.2.2 Tiempo de ejecucion

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 $O(n^2)$. Consideramos que esto es resultado de la forma en que se crea la anteriormente mencionada matriz de preferencias. Suponemos que la implementación de los diccionarios por comprensión de Python permiten un rendimiento mucho mejor que $O(n^2)$.

1.2.3 Reduccion del problema de los casamientos

El algoritmo implementado permite resolver el problema de matching estable cuando el grupo de *reviewers* puede aceptar a más de un *aplicante*. Por lo tanto, si consideraramos que cada reviewer puede aceptar solo a un aplicante, tenemos el problema de la formación de parejas. Entonces, vemos que ese problema puede ser reducido al ya resuelto si la lista Q de vacantes es tal que $Q = [1] * n$.

2 Puntos de Falla

2.1 Objetivo

Encontrar los puntos de fallas (puntos de articulacion) 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 vertice 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 articulacion. 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:

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

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

3 Comunidades en Redes

3.1 Objetivo

Encontrar las componentes fuertemente conéxas 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 órden de los vértices, se invierte el grafo, es decir que se invierten todas sus arístas. 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 conéxa. El órden 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 cuadratica. Esto no concuerda con la predicción esperada, la cuál es lineál. Atribuimos esto a la eficiencia de la computadora en la que se hicieron las pruebas.