

Trabajo Practico 1

De Rosa - Schapira - Guerrero

Primer Cuatrimestre 2017

Contents

1	Asignacion de residencias	1
1.1	Objetivo	1
1.2	Conclusiones	1
1.2.1	Complejidad del algoritmo	1
1.2.2	Tiempo de ejecucion	1
1.2.3	Reduccion del problema de los casamientos	1
2	Puntos de Falla	2
2.1	Objetivo	2
2.2	Conclusiones	2
2.2.1	Funcionamiento del algoritmo	2
2.2.2	Tiempo de ejecucion	2
3	Comunidades en Redes	2
3.1	Objetivo	2
3.2	Conclusiones	3
3.2.1	Funcionamiento del algoritmo	3
3.2.2	Tiempo de ejecucion	3

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 (en el que se resuelve el problema como si) tiene un orden de complejidad $O(nk)$, donde k es la cantidad maxima de vacantes de cada hospital, el algoritmo tiene un orden de complejidad $O(n^2)$ dado por la creacion de la matriz de preferencias, pues es necesario analizar la posicion de cada estudiante en la lista de 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 resolucion 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 implementacion de los diccionarios por comprension 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 mas de un *aplicante*. Por lo tanto, si consideraramos que cada reviewer puede aceptar solo a un aplicante, tenemos el problema de la formacion de parejas. Por lo tanto, 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 elctrica 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 vertice elegido. En este grafo, sera punto de articulacion cada vertice que tenga dos o mas conocidos, es decir que tenga dos o mas hijos. Luego para cada vertice en el grafo de recorrido dfs, chequeamos la cantidad de vecinos que tiene y si se cumple la condicin es un punto de articulacion. Por lo tanto el orden del algoritmo sera $O(v + e)$.

2.2.2 Tiempo de ejecucion

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

Table 2: Tiempo de resolucio 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 vertices. Esto condice con los tiempos esperados del algoritmo.

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 vertices. Esto se realiza para ordenar los vertices en sentido decreciente de acuerdo a su tiempo de finalizacion entre todos los recorridos dfs. Una vez que se obtiene el orden de los vertices, 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 vertice iniciando por el vertice que tiene mayor tiempo de finalizacion. Cada grafo que devuelve cada recorrido dfs es una componente fuertemente conexa. El orden del algoritmo sera $O(v + e)$

3.2.2 Tiempo de ejecucion

Haciendo la prueba con una cantidad v de vertices y a de aristas obtenemos:

Table 3: Tiempo de resolucio 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 resolucio aumentan en medida cuadratica. Esto no concuerda con la prediccion esperada, la cual es lineal. Atribuimos esto a la eficiencia de la computadora en la que se hicieron las pruebas.