Aguirre Ariel *Cañón Cristian* *Algoritmos y Programación II*
100199 *100162* *Cátedra Wachenchauzer*

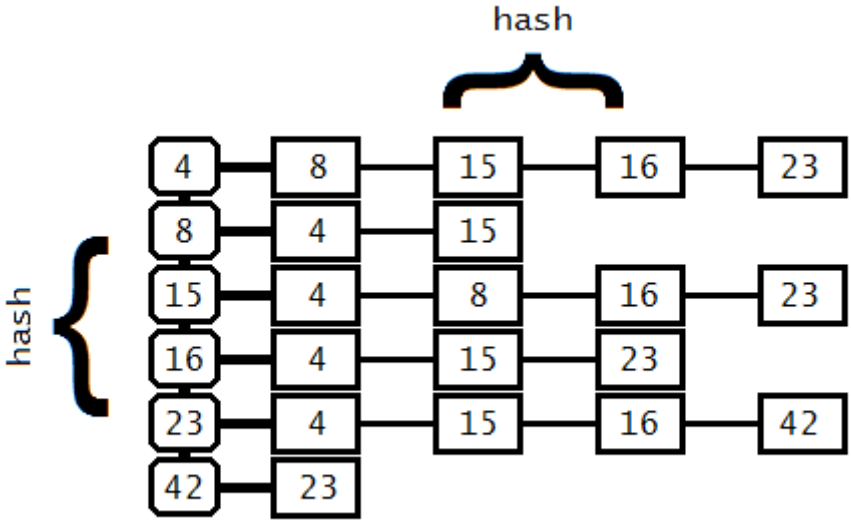
Trabajo Práctico 3

Comunidad de Youtube con grafos

En el presente informe se explicará la implementación propuesta de un grafo no dirigido y los comandos solicitados por la consigna.
Dicho programa se realizó en Python y el archivo de entrada es un flujo de texto tal que por cada línea expresa que la id de un usuario “siga” a la id de otro, de tal manera se puede elaborar un grafo leyéndolo secuencialmente.

TDA Grafo

El grafo que se utilizó consistió en una lista de adyacencias implementándolo como un diccionario de diccionarios, de esta manera las operaciones de agregar vértice, ver adyacentes y agregar y quitar aristas son todas **O(1)**.



En este caso, por ser un grafo no-dirigido, las adyacencias se van añadiendo simétricamente por cada vértice por igual.

Random Walks

Un algoritmo que se utilizó para varias consignas fue el random walks, el cual itera el grafo desde un vértice **v** yendo al azar sobre uno de sus adyacentes **n** veces y devolviendo el camino que realizó.

Aguirre Ariel *Cañón Cristian* *Algoritmos y Programación II*
100199 *100162* *Cátedra Wachenchauzer*

Label propagation

Otro algoritmo utilizado fue el label propagation, se encarga de determinar los vértices que están muy conectados entre sí y pocos conectados con el resto, para esto se le asigna una “etiqueta“ a cada vértice y se pasa a iterar al azar por todos los vértices, asignando en cada ciclo a cada vértice la etiqueta que más se repite entre todos sus adyacentes, de esta manera los vértices quedan conglomerados en contadas comunidades determinadas.

Comandos

El programa una vez cargado acepta 7 comandos que realizan las siguientes operaciones sobre el grafo, para representarlos fueron puestos en una tabla describiendo brevemente su funcionamiento y el orden de cada uno

.

Comando	Algoritmo utilizado	Descripción	Estructuras auxiliares	Orden
Similares	Random walks	Devuelve los n vértices ‘similares’ n ingresado por parámetro, estos son los que más veces aparecen en todos los caminos devueltos	-Hash -Heap	O(V*log(V))
Recomendar	Random walks	Parecido al similares, pero descarta los vértices que ya son adyacentes al ingresado	-Hash -Heap	O(V*log(V))
Camino	BFS	Realizando un BFS devuelve el mínimo camino entre dos vértices	-Hash -Lista	O(V+E)
Centralidad	Random walks	Parecido al similares pero realizando muchos caminos iniciándose en vértices al azar	-Hash -Heap	O(V*log(V))
Distancias	BFS	Parecido al camino, pero guardando en un heap la lista de vértices a todas las distancias	- Heap	O(V+E)
Estadísticas	-	Itera sobre todo el grafo y devuelve las estadísticas	-	O(V+E)
Comunidades	Label propagation	Itera sobre todo el grafo I veces para etiquetar y agrupar los vértices	-Hash	O(I*(V+E))

Aguirre Ariel *Cañón Cristian* *Algoritmos y Programación II*
100199 *100162* *Cátedra Wachenchauzer*

Centralidad y centralidad exacta:

Centralidad O(V*log(v))	Centralidad exactaO(V*V)
Pros: • Mucho mas rápida. • Nos da una buena aproximación. • Para grafos grandes esta buena. • Para grafos con muchos vértices trabaja muy bien.	Pros: • Devuelve los vértices centrales sin error. • En grafos pequeños esta buena. • En grafos con pocos caminos posibles anda bien.
Contras: • No es 100% precisa. • Depende de la cantidad de caminos y el largo de los mismos. • En grafos pequeños puede ser imprecisa en comparación con la exacta. • Al seleccionar un vértice random puede andar mal en grafos no conexos.	Contras: • Toma mas tiempo que la anterior. • En grafos grandes es muy demorada. • Recorre todos los caminos posibles del grafo.

Analizando la tabla anterior, junto con pruebas en tiempo de ejecución decidimos tomar a centralidad como mejor opción para este trabajo práctico, ya que a pesar de que no es 100% precisa, le toma mucho menos tiempo de envolver un resultado que esta bueno y nos puede servir para otra implementación.

Conclusiones finales:

Al trabajar en la implementación de algunas funciones importantes para una red social, nos damos cuenta que en ocasiones no es necesario calcular los datos o alguna información de forma exacta, si no que podemos trabajar con aproximaciones que toman mucho menos tiempo y nos dan un resultado óptimo.
También podemos ver que en una red de información masiva como lo es una red social, en este caso Youtube es muy importante poder hacer cambios u obtener información relevante tanto de los usuarios como de la misma red de forma rápida para así poder controlar y tener información que nos puede interesar para distintos problemas.