Alumnos: De Giácomo - Ponce
Padrones: 99702 - 99723
Corrector: Agustina Mendez

Informe TP3

Estructuras de datos

Grafo

El grafo se implementa como un diccionario de diccionarios. Se utiliza la libreria randomdict (https://github.com/robtandy/randomdict) para obtener claves aleatorias en O(1) y de esta manera agilizar los random walks. Además se mantiene una variable con la cantidad de aristas (para devolver la cantidad de aristas en O(1)).

Sus primitivas son:

Primitiva	Descripción	Tiempo
Agregar vértice	Agrega un vertice	O(1)
Agregar arista	Agrega una arista	O(1)
Eliminar vértice	Elimina un vértice	O(A)
Eliminar arista	Elimina una arista	O(1)
Son adyacentes	Comprueba la adyacencia de dos vértices	O(1)
Obtener adyacentes	Obtiene los adyacentes de un vértice	O(A)
Existe vertice	Comprueba la existencia de un vértice	O(1)
Obtener vertices	Obtiene todos los vértices	O(V)
Cantidad de vertices	Obtiene la cantidad de vértices	O(1)
Cantidad de aristas	Obtiene la cantidad de aristas	O(1)
Camino minimo	Obtiene el camino mínimo para dos vértices	O(V + E)
BFS	Realiza una búsqueda en anchura	O(V + E)
Vertice aleatorio	Obtiene un vértice aleatorio	O(1)

Alumnos: De Giácomo - Ponce
Padrones: 99702 - 99723
Corrector: Agustina Mendez

Random walk	Realiza un camino aleatorio	O(1)

Nomenclatura:

- V → Cantidad de vértices del grafo.
- E → Cantidad de aristas del grafo.
- A → Cantidad de vértices adyacentes a un vértice.

Algoritmos utilizados sobre el grafo

Similares

Descripción

Dado un vértice, encuentra los vértices más similares..

Tiempo

Random walks [O(1)] + Mayores con heap [O(N*log(K))].
K → Cantidad de similares a obtener.
N → Vertices visitados con los random walks.
Total: O(N*log(K)).

Recomendar

Descripción

Dado un vértice, encuentra los vértices más similares que no sean adyacentes.

Tiempo

Random walks [O(1)] + Mayores con heap [O(N*log(K))] + Descarte de adyacentes [O(N)].
K → Cantidad de recomendados a obtener.
N → Vertices visitados con los random walks.
Total: O(n*log(k)).

Camino

Descripción

Obtiene el camino mínimo entre dos vértices.

Tiempo

BFS [O(V + E)]
V → Cantidad de vértices del grafo.
E → Cantidad de aristas del grafo.
Total: O(V + E)

Centralidad exacta

Descripción

Busca los vértices que aparecen más veces entre todos los caminos mínimos existentes.

Tiempo

BFS para cada vertex [O(V* (V + E))] + Mayores con heap [O(V*log(K))]
V → Cantidad de vértices del grafo.
E → Cantidad de aristas del grafo.

Alumnos: De Giácomo - Ponce
Padrones: 99702 - 99723
Corrector: Agustina Mendez

K → Cantidad de vértices centrales a obtener.
Total: $O(V^2)$

Centralidad aproximada

Descripción

Busca una aproximación de los vértices más centrales.

Tiempo

Random walks [O(1)] + Mayores con heap [O(V*log(K))]
V → Cantidad de vértices del grafo.
K → Cantidad de vértices centrales a obtener.
Total: O(V*log(K))

Distancias

Descripción

Dado un vértice, obtiene los vértices que se encuentran a cada una de las distancias posibles, considerando las distancias como la cantidad de saltos.

Tiempo

BFS [O(V + E)] + Invertir diccionario [O(V)]
V → Cantidad de vértices del grafo.
E → Cantidad de aristas del grafo.
Total: O(V + E)

Estadísticas

Descripción

Obtiene algunas estadísticas del grafo.

Tiempo

Obtener cantidad de vértices [O(1)] + Obtener cantidad de aristas [O(1)]
Total: O(1)

Comunidades

Descripción

Busca las comunidades que se encuentren en el grafo, utilizando el algoritmo de label propagation.

Tiempo

Creo un label para cada vértice [O(V)] + Calculo el máximo de los vértices adyacentes [O(A)] para distintos random walks [O(1)]
V → Cantidad de vértices del grafo.
E → Cantidad de aristas del grafo.
A → Cantidad de vértices adyacentes a un vértice.
Total: O(V + A)

Alumnos: De Giácomo - Ponce
Padrones: 99702 - 99723
Corrector: Agustina Mendez

Conclusiones

Similares, random walk y centralidad

Durante la realización del trabajo se observa la correlación entre los vértices recomendados y los vértices más centrales del grafo. Los vértices más centrales del grafo son más propensos a aparecer como recomendados a una gran cantidad de otros vértices. Esto se debe a que al ser centrales es muy probable que muchos “caminos” pasen por ellos. Y cómo se utilizan random walks para calcular los similares, que estos caminos pasen por un vértice central es muy probable.
Esto puede ser evadido (no completamente) modificando los random walks dependiendo del propósito. Utilizando random walks con muchos pasos para calcular los centrales del grafo y random walks con poca cantidad de pasos para calcular los similares sin salir del entorno cercano al vértice.

Implementación del grafo

La implementación como diccionarios de diccionarios ofrece gran eficiencia para la búsqueda, creación y eliminación de vértices y aristas.
El lado negativo de esta implementación es el coste de memoria, que no está optimizado. Para un grafo no pesado y bidireccional (como este caso), se guardan datos que podrían ser descartados sin mayor complicación. Por ejemplo, para el vértice A se guarda su adyacente B en su diccionario (y viceversa), pudiendo solamente ser guardado de una forma.

BFS

La utilidad del algoritmo BFS resulta clara en este trabajo. Obtener el camino mínimo entre un vértice A y un vértice B se hace en O(V + E) (linealmente), mientras que obtener todos los caminos mínimos entre A y todo el resto de los vértices del grafo, también se hace en O(V + E).

Centralidad exacta y aproximada

Comparando los resultados de ambos algoritmos se puede ver que el resultado de la centralidad aproximada se aproxima de buena manera (a veces incluso se obtiene el mismo resultado) que utilizando una centralidad exacta. Con el ajuste en la cantidad de random walks y el largo de los mismos se podría casi asegurar un resultado correcto, por lo que utilizar el algoritmo de una centralidad exacta se vuelve prácticamente improductivo.