Estructura de datos
Algoritmos de búsqueda en gráfos
Actividad 10

Dagoberto Quevedo

3 de abril de 2020

Resumen

En esta actividad se describen métricas y algoritmos de búsqueda y recorrido en grafos. Se procede a realizar la implementación en Python que genera un grafo aleatorio y procede a calcularse su diámetro, densidad, distancia con el algoritmo de Dijkstra y centralidad de grado, adicional se implementa un método de búsqueda en profundidad.

1. Representación eficiente de grafos

Un grafo con n vértices puede se almacenada en un matriz de adyacencias, pero sólo resulta eficiente si la densidad del grafo es alta, dado que almacenar un grafo en una matriz ocupa n^2 de espacio, si $m \ll n^2$ la mayoría de los espacios reservados son cero. La forma de optimizar el espacio de almacenamiento es usar listas de adyacencia, dado un arreglo $a[]$, cada elemento contiene un lista dinámica $a[i]$ con los vértices adyacentes a i; el tamaño de esta estructura es $\mathcal{O}(n+m) \leq \mathcal{O}(m) = \mathcal{O}(n^2)$ [3].

2. Métricas y algoritmos en gráfos

La *densidad* es el número máximo de posible de aristas es,

$$m_{\text{máx}} = \binom{n}{2} = \frac{n(n-1)}{2}, \tag{1}$$

entonces la densidad $\delta(G)$, es,

$$\delta(G) = \frac{m}{m_{\text{máx}}} = \frac{m}{\binom{n}{2}}, \tag{2}$$

por lo que un grafo es denso si $\delta(G) \approx 1$, y un grafo poco denso si $\delta(G) \ll 1$. El *grado* de un vértice i,

$$\eta(i) = |\{j \in V : (i,j) \in E\}| \tag{3}$$

La distancia $\ell(a,b)$ entre a y b es el largo mínimo de todos los caminos de a a b el *diametro* $\gamma(G)$ de un grafo G, el la distancia máxima en todo el grafo,

$$\gamma(G) = \max_{a,b \in V} \ell(a,b). \tag{4}$$

La *centralidad de grado* es dada por,

$$i_{\text{máx}} = \max_{i \in V} \eta(i). \tag{5}$$

2.1. Búsqueda en profundidad

La búsqueda en profundidad (DFS por sus siglas en inglés *Depth-First Search*), explora los posibles vértices (desde un vértice inicial) hacia abajo de cada rama antes de retroceder. Esta propiedad permite que el algoritmo se implemente recursivamente [1]. En general el método realiza las siguientes operaciones: 1) Marcar el vértice actual como visitado; 2) Explorar cada vértice adyacente al nodo actual que no haya sido visitado aún. El algoritmo recursivo se definiria como sigue,

Algorithm 1 Búsqueda en profundidad

procedure DFS(v,L)
  $L = L \cup \{v\}$
  for $w \in \{j \in V : (i,j) \in E\}$ **do**
    DFS(w,L)
  return L

3. Implementación computacional

Se realiza una implementación computacional dado un valor m que determina el numero de elementos en V, generar un grafo no dirigido $G = (V, E)$, donde las aristas $(i,j) \in E, i,j \in V$ se asignan desde una probabilidad uniforme de conexión entre sus elementos definida por p_{ij}. Una vez generado calcular lo siguiente: a) diámetro, b) densidad, distancias, usando el algoritmo de Dijkstra y c) determinar el elemento más y menos central del grafo a partir de la centralidad de grado, donde el grado de un elemento i es dado por $\deg(i) = \sum_{j \in V} A_{ij}$, siendo A la matriz de adyacencias. Adicional se implementa una búsqueda en profundidad recursiva. Finalmente realizar la representación gráfica del grafo generado.

3.1. Resultados

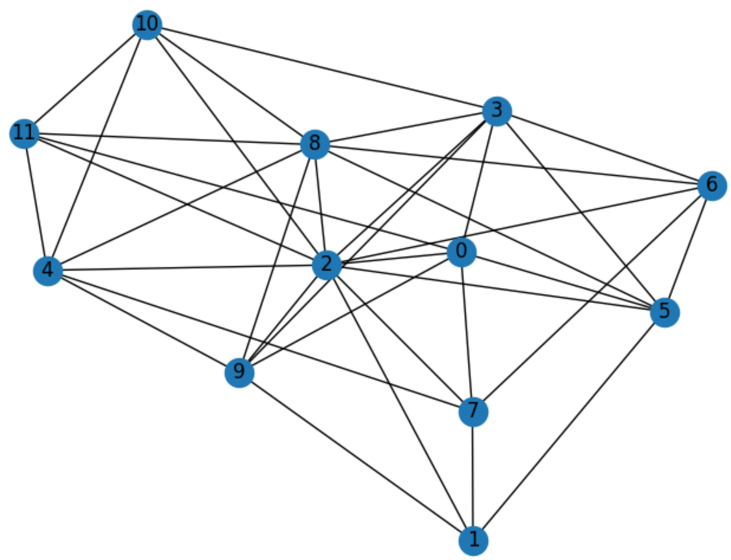

Figura 1: Resultado de la estructura de gráfo generada aleatoriamente

Referencias

[1] Christos H. Papadimitriou, *Computational Complexity*, Addison-Wesley Professional, 1993.

[2] Donald Knuth, *Sorting and searching*, The Art of Computer Programming, Addison-Wesley Professional, 1998.

[3] Elisa Schaeffer, *Modelos computacionales*, Complejidad computacional de problemas y el análisis y diseño de algoritmos, notas de curso, 2020.