Grafos

Algoritmos y Estructuras de Datos (CB100) - FIUBA
Martin Klöckner - mklockner@fi.uba.ar

Los grafos son una estructura de datos compuestos por vértices (o nodos) y unidos por aristas. Tanto los vertices como las aristas pueden contener datos.

Los grafos pueden ser dirigidos o no, los primeros se representan gráficamente con sus aristas como flechas, mientras que los segundos sus aristas se representan como lineas, sin principio ni fin.

(1) Grafo dirigido (2) Grafo no dirigido

Figura 1: Representación gráfica de grafos

Una posible implementación de las estructura de datos grafo es mediante una matriz en la cual en las filas y columnas se ponen los vértices y en la intersección entre vértices se pone 0 o 1 en función de si existe una arista que los conecte. La contra de esta implementación es que cada vértice puede tener como máximo de aristas la cantidad de vértices total que haya en el grafo.

	A	**E**	**D**
A	0	1	1
E	0	0	1
D	0	0	0

Tabla 1: Representación de un grafo en una matriz

La implementación mas común es mediante una lista enlazada de dos dimensiones, cada nodo representando un vértice, y este a su vez conteniendo una lista de los vertices adyacentes, esta version es mas eficiente que la implementación con matriz.

Definiciones

Camino

Un camino es una secuencia de vértices conectados por aristas, donde cada vértice está conectado con el siguiente mediante una arista del grafo (un vértice y otro se dicen adyacentes si estas conectados por una arista). La **longitud del camino** se define como la cantidad de aristas del camino.

Camino simple

En un camino simple no se repiten vértices (excepto, a veces, el primero y el último si es un ciclo).

Camino cerrado

En un camino cerrado, el primer y último vértice son el mismo, sin importar la repetición de vertices.

Camino abierto

Por el contrario a un camino cerrado, un camino abierto es un camino en el cual el primer y ultimo vértice difieren.

Recorrido

Un recorrido es un camino en el cual no se repiten aristas.

Ciclo

Un ciclo es un camino que contiene vértices distintos excepto el primer y ultimo vértice que son el mismo. Si un grafo no tiene ciclos se dice **acíclico**.

Ciclo Hamiltoniano

Un ciclo se dice que es Hamiltoniano si visita exactamente una vez todos los vértices del grafo, y por ser un ciclo vuelve al punto de partida.

Subgrafo

Un subgrafo de un grafo es un grafo en si que se compone de un subconjunto de vértices y aristas del grafo más grande.

Grafo subyacente

Un grafo subyacente es un grafo no dirigido que se obtiene a partir de reemplazar todas las aristas dirigidas por no dirigidas en un grafo dirigido.

Grafo conexo

Cuando se puede acceder a todos los vértices mediante un camino de aristas. En el caso de grafos dirigidos, se debe definir si es fuertemente conexo o débilmente conexo, no alcanza con decir que es conexo ya que puede resultar ambiguo.

Grafo dirigido fuertemente conexo

Un grafo dirigido es fuertemente conexo si y solo si entre cualquier par de vértices existe un camino que los une en ambos sentidos, no necesariamente el camino de vuelta tiene que ser entre los mismos vértices, puede ser que entre dos vértices haya una arista dirigida pero para volver haya que realizar un camino de mayor longitud.

Grafo dirigido débilmente conexo

Un grafo dirigido es débilmente conexo si no es fuertemente conexo, es decir, no existe un camino entre cualquier par de vértices, pero si es conexo su grafo subyacente, es decir aquel que se obtiene de reemplazar las aristas dirigidas por no dirigidas. Por ejemplo el grafo de la figura 1.1 es débilmente conexo, ya que no se puede acceder al nodo A o E desde el nodo D debido a las direcciones de las aristas, y ademas su grafo subyacente si es conexo.

Punto de articulación

Un punto de articulación (también llamado vértice de corte) de un grafo no dirigido conexo es aquel vértice que al eliminarlo del grafo este deja de ser conexo

Árbol libre

Se dice que un grafo es árbol libre si es no dirigido, conexo y sin ciclos.

Recorridos de un grafo

Los dos recorridos básicos en un grafo son en profundidad y en anchura, cada uno visita o procesa cada vértice del grafo, visitando una y solo una vez cada uno.

En profundidad

En el recorrido en profundidad (en ingles Depth-First Search o DFS) se toma el primer vértice (como esté almacenado, ya que no hay un "primer vértice" por definición) y se recorre en profundidad los vértices adyacentes de este, marcando en cada visita los vértices como visitados, es decir, para el primer adyacente se recorren los adyacentes de este y así sucesivamente hasta que se llegue a un vértice que no tenga vértices adyacentes o que ya hayan sido visitados. Cuando se llega a este punto se sigue con el próximo vértice que no este visitado (de acuerdo a como están almacenados) y se vuelve a repetir el algoritmo sobre este nodo, así sucesivamente con todos los vértices del grafo.

La implementación típica es mediante una función recursiva, aunque se puede implementar también de manera iterativa utilizando una cola.

En anchura

El recorrido en anchura (en ingles Breath-First Search o BFS) es similar al recorrido en profundidad pero en lugar de recorrer todos los nodos adyacentes en profundidad se recorren los nodos adyacentes, marcándose como visitados, y luego se sigue con los próximos vértices no visitados (de acuerdo a como se tienen almacenados) recorriendo sus vértices adyacentes, así sucesivamente con todos los nodos del grafo.

La implementación típica es utilizando una función iterativa con la ayuda de una cola, la version iterativo no es común ya que es poco eficiente y difícil de implementar.

Topológicos

Los recorridos topológicos se aplican a grafos dirigidos en los que no hay ciclos (acíclicos) y son un proceso de asignación de orden lineal a los vertices del grafo de modo que se respeten las precedencias indicadas por las relaciones de adyacencia. Se pueden plantear recorridos topológicos como variantes del recorrido en profundidad y en anchura.

Estos recorridos permiten linealizar un grafo de manera que se cumplan las relaciones de adyacencia entre cada nodo.

Aplicaciones de DPS

Puntos de articulación

Detección de componentes conexas

La búsqueda en profundidad también puede usarse para determinar con eficiencia las componentes fuertemente conexas de un grafo dirigido. Una componente fuertemente conexa de un grafo es un conjunto maximal de vertices en el que existe camino entre cualquier par de vertices del conjunto.

Detección de ciclos

Se puede usar el recorrido en profundidad para determinar si un grafo tiene o no ciclos (si es acíclico)

Algoritmo de Dijkstra

El algoritmo de Dijkstra tiene muchas aplicaciones, una de ellas se aplica a grafos y sirve para hallar el camino mínimo entre dos vértices cuando la arista tiene peso no negativo.

Este algoritmo se caracteriza por usar la estrategia *voraz* (o *greedy* en inglés) que se basa en tomar la mejor solución local (sin considerar resultados previos) para avanzar con el algoritmo y llegar a un objetivo el cual solo se alcanza mediante sucesivas decisiones, la estrategia termina cuando se evalúan todas las posibles combinaciones de soluciones.

Programación dinámica

La programación dinámica es una técnica de resolución de problemas en computación. Se utiliza cuando un problema puede descomponerse en subproblemas más pequeños y solapados, y sus soluciones parciales pueden reutilizarse para resolver el problema más grande original. Es muy similar a la técnica "divide y vencerás" (en la cual se divide un problema mas grande en otros mas chicos para luego combinar las soluciones) solo que en la programación dinámica los problemas mas chicos a resolver están superpuestos y los resultados se guardan en memoria para evitar repetir cálculos. Un ejemplo de divide y vencerás es el algoritmo de ordenamiento Mergesort, en este se divide el vector a ordenar por partes y cada parte se ordena de por si para luego combinar cada parte ordenada, un ejemplo de programación dinámica es el calculo de Fibonacci, en el cual el calculo para un numero se calcula el Fibonacci de un numero anterior y se guarda en memoria, luego la solución final es la resta de todos los

valores calculados previamente.

Algoritmo de Floyd-Warshall

El Algoritmo de Floyd-Warshall es similar al Algoritmo de Dijkstra en el sentido que se usa para encontrar el camino mas corto entre dos nodos. En este algoritmo se aplica la técnica de programación dinámica. Es poco eficiente ya que tiene una complejidad de $O(n^3)$

Cerradura transitiva

Árbol abarcador de costo mínimo

Un árbol abarcador de costo mínimo (en inglés Minimum Spanning Tree o MST) es un subgrafo de un grafo no dirigido, ponderado y conectado, que conecta todos los nodos con el coste total mínimo, es decir, consiste en eliminar las aristas que no sean mínimas de un grafo ponderado no dirigido.

Para construir un MST se utilizan principalmente dos algoritmos, ambos equivalente, el algoritmo de Kruskal y el algoritmo de Prim.

Algoritmo de Kruskal

El algoritmo de Kruskal es un algoritmo greedy que encuentra el árbol de expansión mínima (Minimum Spanning Tree, MST) de un grafo no dirigido y ponderado.

Construye un árbol con el peso total mínimo, que conecta todos los nodos del grafo sin formar ciclos.

Pasos

1. Inicializa un conjunto vacío para el MST.
2. Ordena las aristas E por peso ascendente.
3. Para cada arista (u, v) en orden, si u y v están en componentes distintas (no conectados):
   - Añadir (u, v) al MST.
   - Unir los conjuntos de u y v.
4. Detener cuando el MST tenga n - 1 aristas (donde n es el número de nodos).

Algoritmo de Prim

El algoritmo de Prim es un algoritmo greedy que encuentra el árbol de expansión mínima (MST) de un grafo no dirigido y ponderado, conectando todos los vértices con el menor costo total sin formar ciclos.

Algoritmo de Ford-Fulkerson

Este algoritmo se utiliza para encontrar el flujo máximo en una red de flujo. Una red de flujo esta compuesta por:

1. Un grafo dirigido
2. Un vértice fuente desde donde se inicia el flujo
3. Un vértice sumidero donde termina flujo
4. Capacidades en las aristas que determinan el flujo máximo a través de ellas

Backtracking

La estrategia de backtracking (o retroceso) es un método de resolución de problemas que pueden ser expresados en términos de elecciones de decisiones secuenciales.