Report 3 – AEDs III

Willian de Souza Soares – 2014.1.08.034

March 10, 2020

Abstract

Some of the computational representations that can be used on graphs, including *Adjacency Matrix*, *Incidence Matrix* and *Adjacency List*.

1 Adjacency Matrix

Adjacency: a is **adjacent** to b if a is connected to b.

An adjacency matrix is represented by a square matrix, having n elements, $n \times n$ is the number of Vertices on a graph G.

$G = (V, E)$

$V = a, b, c$

$E = (a, b), (a, c), (b, c), (c, a), (c, b)$

G:

	a	**b**	**c**
a		1	1
b			1
c	1	1	

Advantages:

- $\theta(1)$ for access;
- Efficient when dealing with a complete graph.

Disadvantages:

- $\theta(V^2)$ for memory usage;
- Can be under-used and be made mostly of empty spaces, when dealing with a sparse graph;
- Cannot represent valued **and** parallel edges at the same time.

2 Incidence Matrix

Given a graph $G = (V, E)$, the incidence matrix of G has the size $|V| \times |E|$.

$G = (V, E)$

$V = a, b, c$

$E = (a, b), (a, c), (b, c), (c, a), (c, b)$

G:

	e0	**e1**	**e2**	**e3**	**e4**
a	1	1		1	
b	1		1		1
c		1	1	1	1

Advantages:

- $\theta(E)$ for access. Can be a disadvantage when dealing with too many edges;
- Efficient when dealing with sparse graphs

Disadvantages:

- $\theta(V \times E)$ for memory usage: uses too many space when dealing with graphs with many Edges;

3 Adjacency List

Given a graph $G = (V, E)$, the graph G is represented by an array a_v of linked lists. Each of these linked lists represents an edge e adjacent to v.

$G = (V, E)$

$V = a, b, c$

$E = (a, b), (a, c), (b, c), (c, a), (c, b)$

$$G: \begin{array}{ccccc} a & \rightarrow & b & c & // \\ b & \rightarrow & c & // & \\ c & \rightarrow & a & b & // \end{array}$$

Advantages:

- Uses the lowest memory than the another options, $\theta(V + E)$

Disadvantages:

- Slow for access and operations with edges, $\theta(E)$