DS 6^{th} homework

58121102 Jiang Yuchu

November 27, 2022

P_{340} 5. Obtain the adjacency-matrix, adjacency-list, and adjacency-multilist representations of the graph of Figure 6.16.

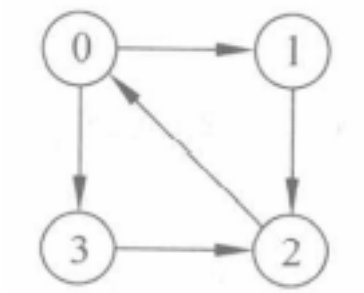

Figure 6.16 A directed graph

Answer:

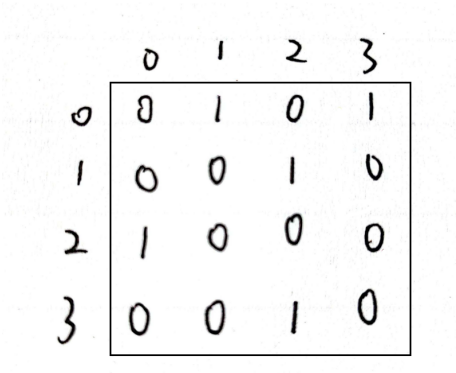

Figure 1: Adjacency-matrix representation

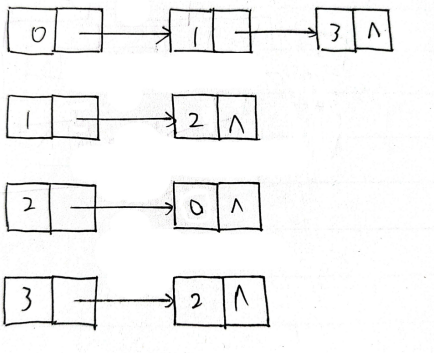

Figure 2: Adjacency-list representation

The graph given in the title is a directed graph, so it cannot be represented by an adjacency multilist. So 3 is a directed graph that I represented with a cross-linked list.

P_{340} 9. Write a C++ function to input the number of vertices and edges in an undirected graph. Next, input the edges one by one and to set up the linked adjacency-list representation of the graph. You may assume that no edge is input twice.What is the run time of your function as a function of the number of vertices and the number of edges?

Answer:

See *p340-9.cpp*. I tested the running time when inserting n ($n = 100, 250, 500, 1000, 2000$) edges (given in pair form, set to (a, b), then a, b belong to $[0, n]$) . Note that if either of the two vertices connected by the edge does not exist, this vertex will be added first.

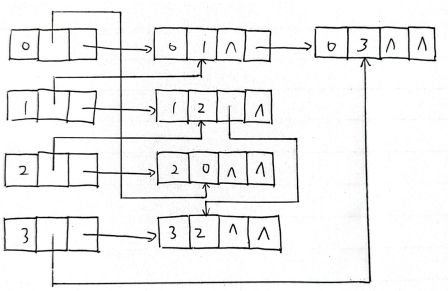

Figure 3: Adjacency-multilist representation

n	100	250	500	1000	2000
total time(us)	530.6	1127.2	1729.1	2666.6	4007.8

P_{352} 5. Write a complete C++ function for breadth-first search under the assumption that graphs are represented using adjacency lists. Test the correctness of your function using suitable graphs.

Answer:

See *p352-5.cpp*. I created a graph shown in figure 4. Then I use bfs to find vertex 7 form vertex 0. The path of its traversal is: $0 \longrightarrow 1 \longrightarrow 3 \longrightarrow 7$

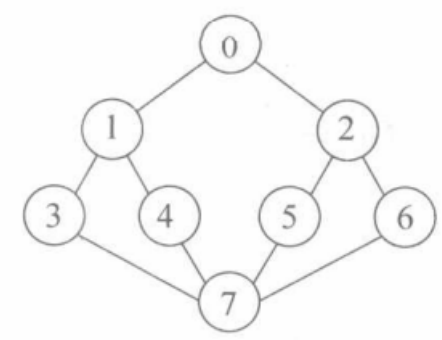

Figure 4: A undirected graph

P_{352} 6. Show how to modify function `DFS`(Program6.1), as it is used in `Components` (Program6.3), to produce a list of all newly visited vertices.

Answer:

I used the same graph mentioned in previous question. The structure of the graph is shown in figure 4. You only need to record the current parent node before entering the next layer of nodes, and then combine the parent nodes after the traversal is complete.