

## Questions

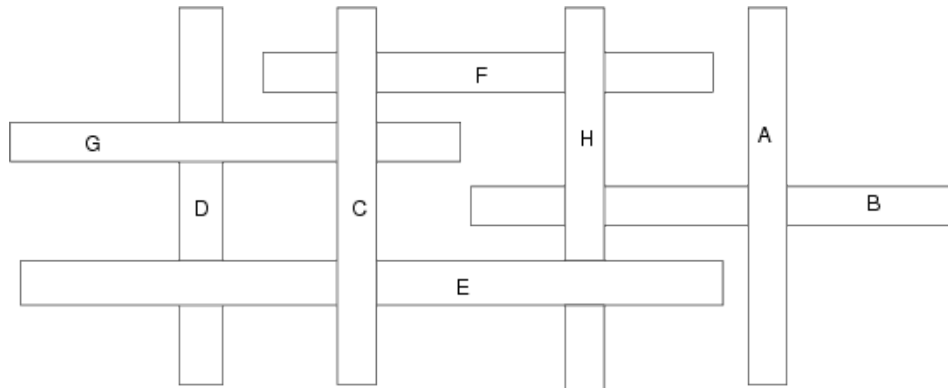
1. A *weighted graph* is a graph such that each edge has a weight (a number). The weights can be represented with an adjacency matrix, whose elements contain the weights (instead of boolean values). Note that this representation does not distinguish the case that there is an edge with weight 0 from the case that there is no edge. For this reason, we interpret a 0 in the matrix to mean that there is no edge.

Let  $A$  be the adjacency matrix of a weighted graph,  $G$ , with the columns and rows labelled, in order, by the vertices of the set  $V = \{v_0, v_1, v_2, v_3, v_4, v_5\}$ . That is, the first row of the matrix represents the adjacencies of vertex  $v_0$ , the second row those of vertex  $v_1$ , etc. Note that there are some non-zero diagonals which means that there are some edges of the form  $(v, v)$ .

$$A = \begin{bmatrix} 0 & 1 & 3 & 0 & 0 & 2 \\ 1 & 1 & 0 & 4 & 5 & 0 \\ 3 & 0 & 0 & 5 & 4 & 0 \\ 0 & 4 & 5 & 2 & 0 & 6 \\ 0 & 5 & 4 & 0 & 0 & 2 \\ 0 & 0 & 0 & 6 & 2 & 3 \end{bmatrix}$$

- (a) Write down the adjacency list for the graph, such that the edges in the list are ordered by increasing weight.
  - (b) Perform a depth first search on  $G$ , beginning at vertex  $v_0$ . Use the ordering in the adjacency list to determine the ordering of vertices visited.
  - (c) Perform a breadth first search of the graph,  $G$ , again beginning at  $v_0$  and using the ordering in the adjacency list.
2. Would you use the adjacency list structure or the adjacency matrix structure in each of the following cases? Justify your choice.
    - (a) The graph has 10,000 vertices and 20,000 edges and it is important to use as little space as possible.
    - (b) The graph has 10,000 vertices and 20,000,000 edges, and it is important to use as little space as possible.
    - (c) You need to answer the query `areAdjacent` as quickly as possible, no matter how much space you use. (Two vertices  $u$  and  $v$  are adjacent if either  $(u, v)$  or  $(v, u)$  is an edge in the graph.)
  3. Explain why the DFS traversal runs in  $O(n^2)$  time on an  $n$ -vertex graph that is represented with an adjacency matrix structure.
  4. Write the depth first traversal from lecture 34 using a non-recursive algorithm, namely use a stack.

5. The figure below shows a set of rectangles whose overlap relationships can be represented with a directed graph. Each rectangle is represented by one vertex, and there is a directed edge whenever one rectangle overlaps another rectangle. For example, there is an edge (A,B) but no edge (B,A).



- (a) Give the adjacency list for this graph. *The vertices must be ordered alphabetically.*
- (b) Give the ordering of vertices visited in a *breadth first* traversal of the graph, starting from vertex C. *Show the BFT tree.*  
 NOTE: In this question and the next, there is only one answer since you must order the vertices alphabetically.
- (c) Give the ordering of vertices visited in a *preorder depth first search* traversal, starting from vertex C. *Show the DFT tree.*
- (d) Give a *new* example having four rectangles I, J, K, L, such that corresponding graph contains a *cycle*. Draw the rectangles and the graph.

## Answers

1. (a)  $v_0 - v_1, v_5, v_2$   
 $v_1 - v_0, v_1, v_3, v_4$   
 $v_2 - v_0, v_4, v_3$   
 $v_3 - v_3, v_1, v_2, v_5$   
 $v_4 - v_5, v_2, v_1$   
 $v_5 - v_4, v_5, v_3$ 

(b) dft:  $v_0, v_1, v_3, v_2, v_4, v_5$

(c) bft:  $v_0, v_1, v_5, v_2, v_3, v_4$
2. (a) An adjacency list would be better. Why? An adjacency list would have about 20,000 nodes, whereas the adjacency matrix would require  $10,000 \times 10,000 = 100,000,000$  booleans. A boolean is still typically represented with a byte (and the value would be either 0 or 1), so the matrix would require more space.

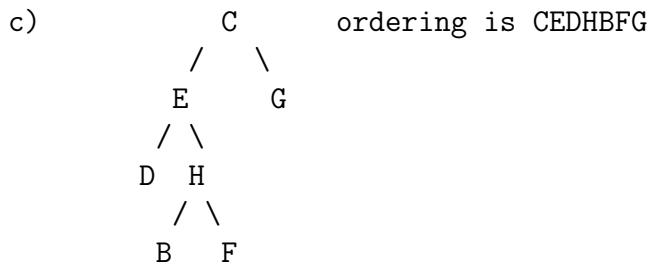
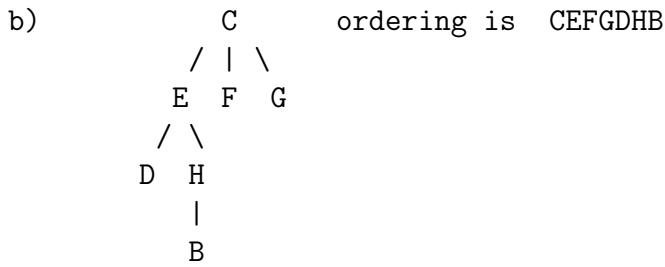
(b) Regarding the space requirement, there is no clear winner. (Note that the exact space usage of the two structures depends on the implementation details, in particular, of a node.) The adjacency matrix structure is much better for operation `areAdjacent`, while the adjacency list structure is much better for operations `insertVertex` and `removeVertex`.

(c) Adjacency matrix. To answer `areAdjacent(i,j)` one need only read off matrix entries  $(i, j)$  and  $(j, i)$ , a constant time operation. In an adjacency list, one would need to search along row  $i$  to find  $j$ , and operation that would take more than constant time. (One can, in principle, speed this up by using a non-linear data structure instead of a linked list.)
3. A DFS traversal will consider every vertex of the graph, eventually, else it would not transverse the graph. (We are assuming a connected graph.)  
 For each vertex that the DFS considers, it must consider the set of neighbours of of the vertex – in order to do so, it must examine the entire row of the adjacency matrix corresponding to the vertex.  
 Thus, for each of  $n$  rows/vertices, the  $n - 1$  non-diagonal entries of the corresponding row must be considered, which makes for  $\mathcal{O}(n^2)$  operations.
4. 

```
depthFirstTraversal(v){
    initialize empty stack s
    s.push(v)
    v.visited = true
    while (!s.empty) {
        if ((w.visited == false) for some w in adjList(s.top)){
            w.visited = true
            s.push(w)}
        else
            s.pop()    }
}
```

5.

- a) A - B
- B -
- C - E,F,G
- D -
- E - D,H
- F -
- G - D
- H - B,F



- d) Use the overlap method for the four (folding) top pieces of a packing box.

