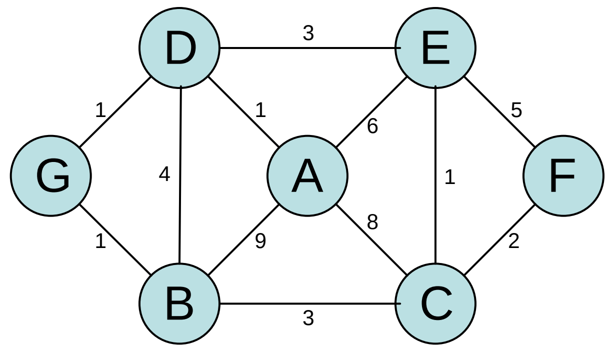
**FINAL EXAMINATION – MAY 2008**
**CSC 112**
**NICHOLAS R. HOWE**

*This is a closed-book exam. You may use two double-sided 8.5x11 sheets of notes.*

*All answers to this exam should be written in your exam booklet(s). Start with the questions that you know how to do, and try not to spend too long on any one question. Partial credit will be granted where appropriate. You will have two hours and twenty minutes. Good luck!*

## 1.) **Graph Traversal** (12 points)

At right is a diagram of a graph, with weights shown on each edge. Simulate a run of Dijkstra's shortest path algorithm starting from node A. In what order are the nodes visited? Show all distance values that are assigned over the course of the algorithm, crossing out old values as they are updated. Indicate the backpointers by an arrow.




## 2.) **Sorting** (8 points)

Consider the array of numbers below, which are to be sorted in increasing order from left to right. Simulate the array version of the algorithms specified, and show the state of the array after **each** swap performed.

3, 1, 6, 5, 2, 4

a.) Selection sort, array implementation, growing the sorted region from left to right.

b.) Insertion sort, array implementation, growing the sorted region from left to right.

## 3.) **Recursion** (14 points)

Describe the development process we have used in class for a recursive algorithm, explaining the purpose of each of the steps. How can we be sure a particular recursive algorithm will terminate? Finally, draw connections between the parts of a recursive routine and the parts of a formally developed iterative loop (again using the methodology presented in class).

## 4.) **Trees** (12 points)

Draw the arithmetic expression trees corresponding to the following **prefix** expressions. Then write corresponding expressions using **infix** and **postfix** notation, employing parentheses only where necessary.

a.) + 3 2

b.) + / 8 2 * 3 5

c.) - + - + 1 1 1 1 1

## 5.) **Hash Tables** (12 points)

Suppose that you create a hash table with seven entries, and use **k mod 5** as your hash function. Your table will use open adressing with linear probing, and relocation to fill gaps on deletion. Draw the state of the table after **each** of the following operations, assuming it begins empty:

a.) Insert **Computer Science I** under key 111.

b.) Insert **Microprocessors and Assembly** under key 231.

c.) Insert **Computer Science II** under key 112.

d.) Delete key 111.

e.) Insert **Computational Geometry** under key 274.

f.) Insert **Computer Networks** under key 249.

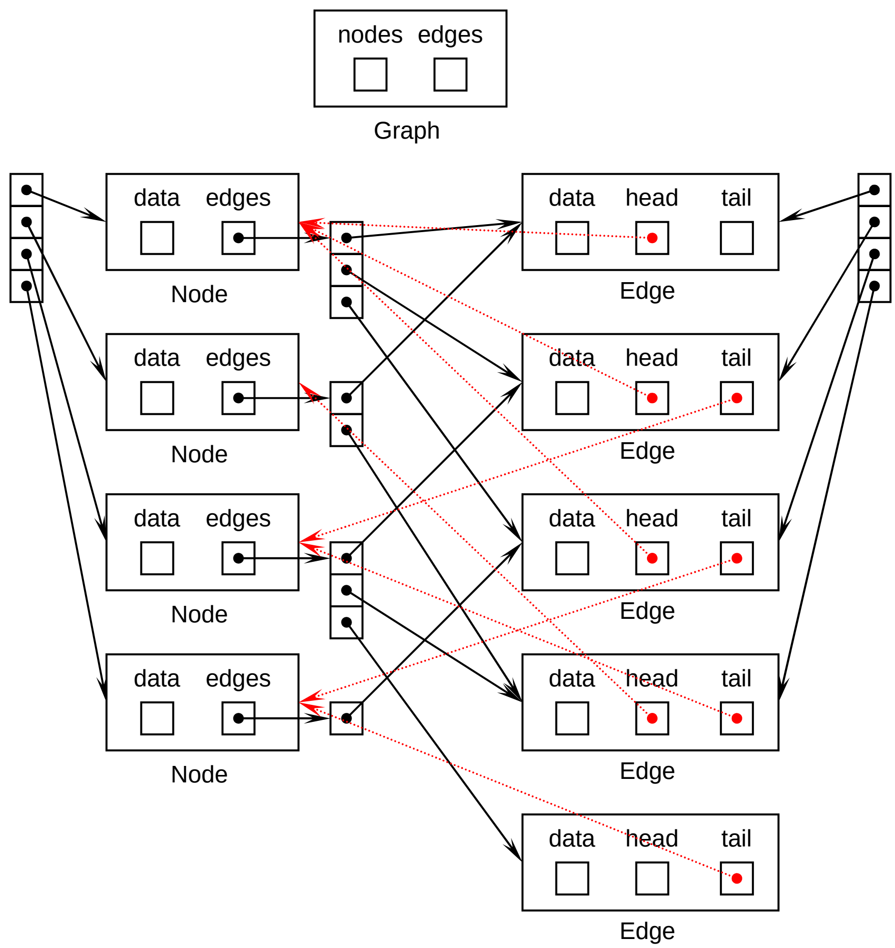
## 6.) **Data Structures** (16 points)

For each of the following data structures and operations, fill in the table with the time complexity of the operation, or that the specified operation is not possible with the specified data structure. You may choose from the following values: $\mathcal{O}(1)$, $\mathcal{O}(\log n)$, $\mathcal{O}(n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2)$, *not possible.*

| | Array | Linked List | Binary Search Tree | Hash Table |
|---|---|---|---|---|
| Insert a new element after an arbitrary element in a linked list. (Assume the element location is given to you.) | | | | |
| Delete an arbitrary element in an array. (Assume the element location is given to you.) | | | | |
| Find the location of an arbitrary element | | | | |
| Produce a sorted listing of all elements | | | | |

## 7.) **Graphs** (8 points)

The **`Graph`** data structure shown at right has lost a number of references that should be present. Assuming that all of the structure shown is correct, draw any additional arrows representing references that should be present in a complete structure.



## 8.) **Classes & Object-Oriented Programming** (18 points)

Define each of the following terms, specifying its role in a program and how/where it would normally be created or defined in a well-structured program.

a.) Constructor

b.) Field

c.) Method

d.) Instance

e.) Accessor

f.) Static method