Institut for Matematik og Datalogi
Syddansk Universitet

December 8, 2010

Peter Schneider-Kamp

Programmering B
5th Weekly Note (E10, Week 49)

Reading for Week 49

Tutorial on Hashing, available from:

`http://research.cs.vt.edu/AVresearch/hashing/`

Lecture: Tuesday, December 7, 10-12 (U133)

We repeat depth-first search and introduce breadth-first search. We also apply these algorithms to a practical problem. Then the 2nd part of the project will be handed out and discussed.

Discussion: see schedule for time and room

1. Perform the following exercises about priority queues:

   - What is the output of the following sequence of operations on a priority queue?

     insert(5,A), insert(4,B), insert(7,I), insert(1,D), removeMin(), insert(3,J), insert(6,L), removeMin(), removeMin(), insert(8,G), removeMin(), insert(2,H), removeMin(), removeMin()

   - Where can an element with the biggest key be stored in a heap?
   - Illustrate the execution of Heap Sort on the following sequence:

     (2,5,16,4,10,23,39,18,26,15).

   - Explain why the case where a node has a right child but no left child does not need to be handled when updating the pointer to the last node.

- Is there a heap T with seven elements with pairwise different keys such that a pre-order traversal of T will output the keys in ascending or descending order? What about an in-order traversal? What about a post-order traversal? In each case, give an example of such a heap or explain why it cannot exist.
- Draw a heap that contains all odd numbers between (and including) 1 and 59 (without repetitions) such that inserting the key 32 will lead to 32 being bubbled up all the way up to the root (not including the root).

2. Perform the following exercises about graphs:

   - Bob loves foreign languages and would like to plan his language courses for the next year. He is interested in the following 9 language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141 og LA169. The pre-requirements for the courses are as follows:
     - LA15 : (none)
     - LA16 : LA15
     - LA22 : (none)
     - LA31 : LA15
     - LA32 : LA16, LA31
     - LA126: LA22, LA32
     - LA127: LA16
     - LA141: LA22,LA16
     - LA169: LA32

     Find the order of courses such that Bob satisfies all pre-requirements.
   - Let G be a graph where the nodes er the integers between (and including) 1 and 8. Assume that the neighbours er like this:

node	neighbour nodes
1	(2, 3, 4)
2	(1, 3, 4)
3	(1, 2, 4)
4	(1, 2, 3, 6)
5	(6, 7, 8)
6	(4, 5, 7)
7	(5, 6, 8)
8	(5, 7)

Assume that a traversal of G will produce the neighbour nodes in the order given by the above table.

(a) Draw G.

(b) Give the incidens matrix for G as a matrix over 0 and 1.

(c) Give the order in which nodes are visited in a DFS or BFS traversal of G starting in node 1.

- Would you use an incidence matrix or a neighbour list in each of the following cases. Explain your answer.

  (a) The graph has 10,000 nodes and 20,000 edges and it is important to use minimal memory.

  (b) The graph has 10,000 nodes and 20,000,000 edges and it is important to use minimal memory.

  (c) You should answer the question "Are U and V neighbours" as fast as possible. It does not matter how much space you use.

- DFS can be implemented without recursion using a stack.

  Let G be a graph with the nodes A, B, C, D, E, F, G and assume that the neighbours are given as below:

node	neighbour nodes
A	(B, C)
B	(A, D, E)
C	(A, E, F)
D	(B, E, G)
E	(B, C, D)
F	(C)
G	(D)

Determine the order that DFS visits the nodes and give the contents of the stack each time that it changes. Start the search in node A.

Lab: see schedule for time and room

1. Implement a sorting algorithm based on priority queues. Start using Java's built-in PriorityQueue and concentrate on getting the algorithm itself to work. The idea is that all numbers are inserted into the queue and then the minimal elements is extracted repeatedly.

   Then implement a class for priority queues based on a heap represented by an ArrayList using the following idea: The root is on index 0. For a node at index i, the left child is at $2 * i + 1$ and the right child at $2 * i + 2$. The parent can be found by $\lfloor \frac{i-1}{2} \rfloor$.

   Test your priority queue class as an alternative backend of your sorting algorithm above.

2. Start on your 2nd part of the exam project.