Repeat Exam ID1020 Algorithms and Data Structures

Friday 2016-03-18

Examiners: Johan Karlander, Jim Dowling
Help Material: none
Examination questions do not need to be returned at the end of the exam.

Every submitted sheet should include the following information:

1. Name and personal number
2. Number for the attempted question
3. Page number

The exam consists of two parts. Part I contains questions to obtain grade E. Part II, which is not obligatory, contains four questions for grades A-D. Maximum writing time is four hours.

Every solution will be given a grade from 0-10 marks (points).

The requirement for a passing grade (E) is at least 50 marks in Part I. You have to receive a passing grade in Part I to qualify for a higher grade. You marks from Part I are not transferred to Part II.

On the condition that you have received a passing grade for Part I, the following are the grading ranges for Part II:

- Grade D: 5-10 marks
- Grade C: 11-20 marks
- Grade B: 21-30 marks
- Grade A: 31-40 marks

Solutions will appear on the course homepage.

Part I Questions

The following questions are for grade E. You are required to give complete solutions and motivate your answers for all questions.

1. Give the worst case time complexity as a function of N for the following code fragments. Use the tilde notation. N is a large positive integer. All elements of the array a are distinct.

   a)

```
for (int i=0; i<N; i++)
   for (int j=0; j<N; j++)
     for (int k=j+1; k<N; k++)
         cnt++;
```

   (3 pts)

   b)

```
int i=N, cnt=0;
while (i > 0) {
   i = i / 2;
   cnt++;
}
```

   (3 pts)

   c)

```
// array 'a' has size N and is sorted
boolean find(int x, int[] a) {
  int i=0;
  int j=a.length-1;
  while(i < j) {
    int k= i+j/2;
    if (a[k] == x)
      return;
    if (a[k] < x)
      j=k;
    else
      i=k;
  }
}
```

   (3 pts)

   Which of the above code fragments have the same expected time complexity and worst-case time complexity?
   (1 pt)

2. Answers should be short (a sentence or two).

   a) If we perform a binary search for an element in the array given below, which elements will not be able to be found if we can only examine 3 or fewer elements in the array?
   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
   (2 pts)

   b) What is meant by an in-place sorting algorithm?
   (2 pts)

   c) Is mergesort in-place? Explain why or why not.
   (2 pts)

   d) Which of the following sorting algorithms are stable: selection sort, heapsort, insertion sort, mergesort, and quicksort?
   (2 pts)

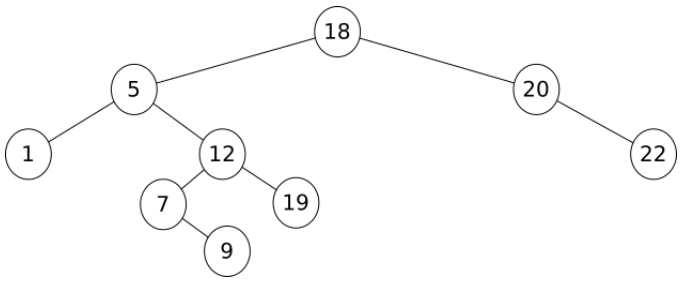
   e) Which of the following sorting algorithms has the lowest expected time complexity for an array that is almost fully sorted: selection sort, insertion sort, mergesort?
   (2 pts)

3. You have access to a linear probing hash table. The table contains N entries and the size of its backing array, på, is $2N$. Calculate both the execpted time complexity and the worst-case time complexity for the following cases:

   a. A search with a hit where all of the elements have their own unique hash code.
      (2 pts)
   b. A search with a hit where all of the elements have the same hash code.
      (2 pts)
   c. A search with no hits where all of the elements have the same hash code.
      (2 pts)
   d. A search with a hit where all of the elements hash to an index that is a multiple of M. The distribution of hashes is the same for the different possible multiples. For example, if $M = 4$, there are the same number of elements that hash to the different multiples of 4 (that is, 4, 8, 12, etc).
      (4 pts)

4. a) Describe briefly what a binary search tree (BST) is, including its properties.
(3 pts)

b) Is the following binary tree a BST or not, and why?
(3 pts)



c) What is the worst case time complexity of a search operation in a red-black binary search tree?
(2 pts)

d) Normally a matrix is represented as an array of arrays. However for large sparse arrays, this consumes too much memory. Describe a more efficient alternative.
(2 pts)

5. a) Draw a hash table with chaining and a size of 8. Use the hash function $key\%8$ (key mod 8) to insert the following keys in your table: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89.
(2 pts)

b) Assume that you have a hash function which uniformly hashes elements to multiples of 32. Assume that you use the standard resizing function. If the hash tables uses linear probing what is the expected number of array accesses for a hit and miss.
(2 pts)

c) Assume that you have a hash function which uniformly hashes elements to multiples of 32. Assume that you use the standard resizing function. If the hash tables uses separate chaining what is the expected number of array accesses for a hit and miss.
(2 pts)

d) Compare using hash tables versus using red-black BSTs (or derivatives thereof) for symbol tables. What are the advantages and disadvantages of each.
(4 pts)

6. Social network connectivity. Given a social network with N members and a logfile that includes M timestamps, where each timestamp includes the earliest time at which a pair of members became friends, design an algorithm that determines the earliest point in time at which all members became connected (that is, every member is a friend of a friend ... of a friend). You can assume that the logfile is sorted by timestamp (starting from the earliest) and that friendship is an equivalence relationship. The expected time complexity should be $Mlog(N)$ or better.
(10 pts)

7. In the figures a), b), c), d) below, there are illustrated a number of different binary trees. Your answer to the question below can be one of: none, one or more than one.

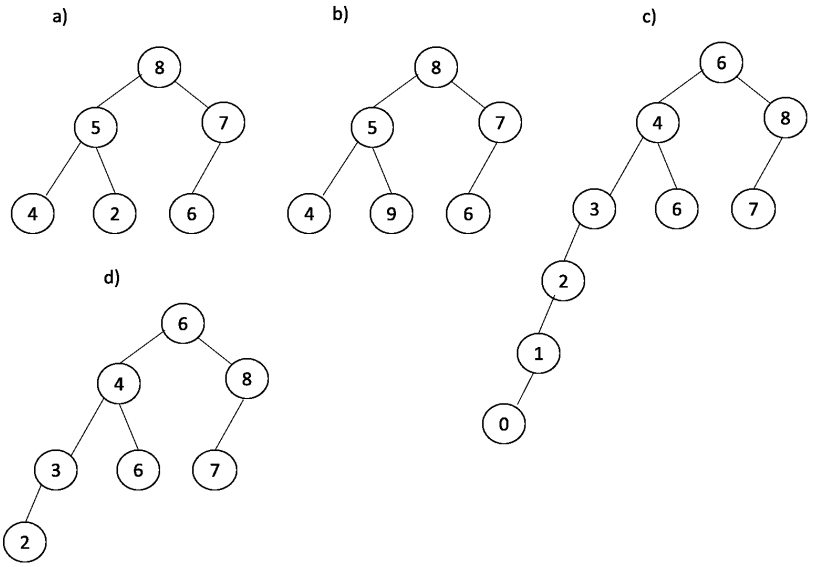
(a) Which one could represent a binary heap?

(b) Which one could represent a binary search tree?

(c) Which one could represent a red-black tree?

(d) Which one could represent the implicit tree in Quick Union?

(10 pts)



8. Assume we have an undirected graph $G = (V, E)$ with a given edge e. Describe an algorithm that decides if e is a member of any cycle in G or not.
(10 pts)

9. Kruskal's algorithm is known to work even if some edges in a graph have the same weight - although the minimal spanning tree is not necessarily unique. The algorithm works even if the edge weights are negative. Let us now assume that we have a connected graph G where all edges have a weight of either 1 or -1 (that is, some edges have one weight, the others have the other weight). Suggest a modification of Kruskal's algorithm that exploits this simple form for edge weights and gives a slightly better running time than the normal version of Kruskal's algorithm.
(10 pts)

10. Assume that we have a graph with weighted edges. Is it possible to use BFS or any simple modification of BFS to calculate the shortest path (with weights representing the distance between nodes)? If not, explain why not.
(10 pts)

Part II Questions

The following questions are for grade A-D. You are required to give complete solutions and motivate your answers for all questions.

1. A Fibonacci function calculates the value of n!, given an input n. Write in Java (or pseudocode) both an iterative and a recursive implementation of the Fibonacci function. Then, for Java, compare the memory complexity of your iterative Fibonacci function with your recursive solution. Which version has (or should have) lower memory complexity? Is there an alternative programming language you could use to improve the more inefficient solution, and if there is what mechanism does it use to reduce its memory complexity compared to Java?

2. a) A number of recurrance relations are listed below. Give the time complexity for each recurrance relation. Then, for each recurrance relation name a sorting algorithm that has the same worst-case time-complexity.

   (a) T(n) = O(1) + T(n/2)

   (b) T(n) = O(1) + T(n-1)

   (c) T(n) = O(n) + 2 T(n/2)

   b) Are you able to make modifications to an array-based implementation of quicksort so that the array is partitioned into three parts, while still being in-place? (The three parts are, respectively, : (1) keys less than the pivot, (2) keys equal to pivot, and (3) keys greater than the pivot.) Justify your answer.

3. a) An array-backed priority queue can store its elements in either (1) sorted order or (2) arbitrary order (random). What is the worst-case time complexity for insertion into an array-backed priority queue for (1) sorted order and (2) arbitrary order? What is the worst-case time complexity for removal of an element from an array-backed priority queue for (1) sorted order and (2) arbitrary order?

   b) Briefly explain the main operations on a symbol table? Symbol tables can be implemented using ordered linked lists or ordered arrays. Discuss the data structures that you would associate with a symbol table maintained as a list of hashtables and how the operations of a symbol table would be implemented in that case.

4. If G is a directed graph, we say that G is semi-connected if, for all pairs of vertices a, b in G, there is either a directed path from a to b or a directed path from b to a or directed paths in both directions between a and b. Note that this term is not the same as strongly connected. How do you determine if a graph is semi-connected? Describe an efficient algorithm for solving this problem. In order to get maximum points/marks, your algorithm should not soley perform a DFS-search from every vertex. You should design a more efficient solution.