

## 58131 Data Structures (Spring 2009)

Course Examination 2, 29 April 2009

Give your answers to each problem (4, 5, 6) on separate sheets of paper. Write your name and student number on each paper. The problems are numbered 4–6; the problems in the first exam had numbers 1–3 and you will in due time see your points problemwise on the course homepage.

In problems asking for an algorithm, you may use another pseudocode style than the one used in the course material, and you may also use e.g. Java. If you need in your solution an algorithm known from the course (e.g., Prim, Dijkstra, quicksort,...) you have to write it out in your solution.

You can get at most 8 points for each problem, in total at most 24 points.

You are allowed to have an A4-sized sheet of paper of personal notes with you at the exam.

4. Answer for each claim whether it is true or false **and give a short motivation**. Each item gives maximum one point.
  - (a) If the load factor for the hash table becomes high, key search will grow slower.
  - (b) It would be good that the hash function returns a prime number not close to a power of two.
  - (c) Searching a key from a hash table takes time at most  $O(1)$ .
  - (d) A good hash function gives the same hash value to all keys.
  - (e) Open addressing uses chaining.
  - (f) In universal hashing, the hash function is chosen randomly.
  - (g) Open addressing is always the best method.
  - (h) Double hashing is a method for open addressing.
  
5. We are given as input an array that contains  $n$  integers. We want to determine if there are two numbers whose sum equals an integer  $k$ , also given as input. For instance, if the array includes 8, 4, -11, 6 and  $k = 10$ , the answer is yes ( $4 + 6 = 10$ ). A number may be used twice. Thus, if in the example we had  $k = 8$ , the answer would again be yes ( $4 + 4 = 8$ ). Give an  $O(n \log n)$  algorithm to solve this problem.

Note: If your worst-case time complexity is  $\Omega(n^2)$ , you will get zero points for this problem, as it is trivial to construct such an algorithm!

**See next page!**

6. Solve **either** problem (a) **or** (b).

(a) Consider a communications network with  $n$  computers  $a_1, \dots, a_n$ .

If there is a direct communication link (say, cable) between computers  $a_i$  and  $a_j$ , we denote by  $p(i, j)$  the delay associated with the link. The delay  $p(i, j)$  tells how long it takes for a (fixed-length) message sent from  $a_i$  to reach  $a_j$  using the link, and is always a positive real number. All existing links and their delays are known in advance. The links are not symmetrical, so it is possible that  $p(i, j) \neq p(j, i)$ .

Additionally there is a delay  $q(i)$  associated with each computer  $a_i$ . The delay  $q(i)$  is the time that a message passing through  $a_i$  must wait there before it can continue.

When a message is routed through several machines, the total time it requires to reach its destination is the sum of all the delays in the computers and links along its route. This includes the delays at the sending and receiving computer.

The task is to find for  $j = 2, \dots, n$  the fastest route from  $a_1$  to  $a_j$ . Give an efficient algorithm for the task. Explain the basic idea of your solution briefly and analyse its time complexity.

You do not need to present the pseudocode for operations on any auxiliary data structures you wish to use (list, priority queue, balanced search tree, etc.); implementation of such data structures can be assumed to be available. Similarly, you can take the time complexities related to operations on such data structures as known.

(b) For a weighted undirected graph  $G = (V, E)$ , show that if no two arcs have the same weights, the graph has a unique smallest spanning tree.

**Have a nice summer!**