

Data Structures and Algorithms  
(Algorithms and Complexity)  
Exam  
9:00 - 11:00, February 13, 2006

Prof. Dr. M. H. Böhlen

*The exam consists of 4 exercises. For each exercise you can get 25 points. It is important that you argue for your answers and that you present your solutions in a readable form. Fully specify all algorithms that you use as part of your solutions.*

### Exercise 1

Consider a trucking company that ships goods between cities. On some roads goods are shipped in one direction, whereas on other roads good are shipped in both directions. A city  $T$  is a *hub* if and only if the goods from any other city can be shipped to  $T$ .

As an example consider cities Bozen, Meran, Sterzing, and Brixen. Between Bozen and Meran and between Bozen and Brixen goods are shipped in both directions. Goods are also shipped from Sterzing to Brixen and from Sterzing to Meran. Goods are not shipped in any other direction. In this example Bozen is a hub, but Sterzing is not.

1. Suggest a mathematical formulation of the problem. Show how the example is represented in your model. A drawing is sufficient.
2. In the above example is Brixen a hub?
3. Give an algorithm that checks if a given city  $T$  is a hub. You may change your model if this makes your algorithm simpler. What is the worst case complexity of your algorithm?
4. Assume the distances between cities is given. The cities that can be reached by driving less than 30km from a town  $T$  belong to the neighborhood of  $T$ . Give an algorithm that determines all cities in the neighborhood a given city. What is the complexity of your algorithm?

## Exercise 2

A *persistent binary search tree* (PBST) stores all versions of a binary search tree as illustrated in Figure 1. A separate root is maintained for each version of the binary search tree.

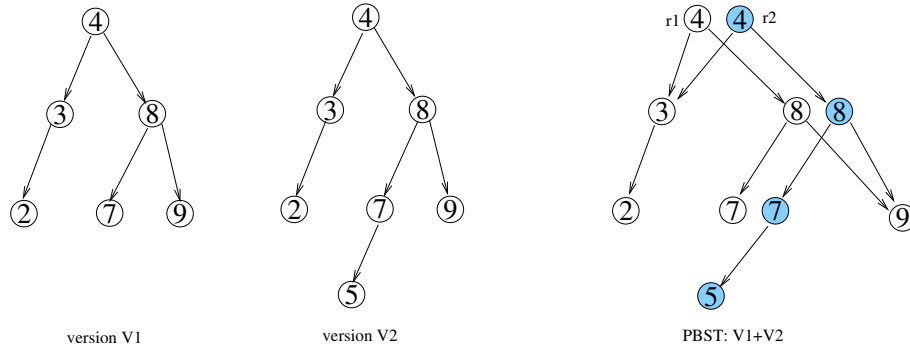


Figure 1: Persistent Binary Search Tree

For example, in order to insert a new node with key 5 into the tree a new node with key 5 is inserted. This node becomes the left child of a new node with key 7. Similarly the new node with key 7 becomes the left node of a new node with key 8 whose right child is the existing node with key 10. The new node with key 8 becomes the right child of a new root  $r'$  with key 4 whose left child is the existing node with key 3. Thus, we copy only part of the tree and share some of the nodes with the original tree.

1. Assume a new version of the tree is created by deleting the node with key 4. Draw the resulting persistent binary search tree.
2. For a general persistent binary search tree identify the nodes that need to be changed to insert a node with key  $k$  or delete a node with key  $l$ .
3. Write a procedure PBST-Insert that, given a persistent binary search tree  $T$  and a key  $k$  to insert, returns a new persistent binary search tree that is the result of inserting  $k$  into  $T$ .
4. If the height of the persistent binary search tree is  $h$  what are the time and space requirements of your implementation of PBST-Insert?

### Exercise 3

Consider the recurrence:

$$T(n) = 2 * T(n/2) + n^3$$

Assume that  $T(n)$  is constant for  $n \leq 2$ .

1. What is the value of  $T(n)$  for  $n = 2, 4, 8$ ?
2. Solve the above recurrence exactly by finding a closed form expression for it (without  $T(n)$  on the right side of the expression).
3. Find the simplest  $f(n)$  such that  $T(n) = \Theta(f(n))$ .

### Exercise 4

Assume hashing with open addressing as a collisions resolution strategy.

Hash function H1 is defined as

$$h(k, i) = (h1(k) + c1 * i + c2 * i^2) \mod m$$

hash function H2 is defined as

$$h(k, i) = (h1(k) + i * h2(k)) \mod m$$

and

$$\begin{aligned} m &= 11 \\ h1(k) &= k \mod m \\ h2(k) &= 1 + (k \mod (m - 1)) \\ c1 &= 1 \\ c2 &= 3 \end{aligned}$$

1. For both hash functions show the state of the hash table after the following operations: insert 10, insert 22, insert 31, insert 15, insert 4, delete 22, insert 59, delete 15, insert 17, insert 88.
2. Write a procedure for initializing the hash table.
3. Write a procedure for inserting an element into a hash table.
4. Write a procedure for deleting an element from the hash table.