Preface

This is a collection of typical problems recommended for the problem solving sessions (Laboratory) of the DALK-part of TDD 56 and of TDD 57 and for self studies. The collection consists of explicitly stated problems and of references to problems in the course book: Michael T. Goodrich and Roberto Tamassia Data Structures & Algorithms in JAVA Fourth Edition, John Wiley & Sons, Inc., 2006.

The collection developed gradually during the previous editions of this course. Many of the problems were included in the previous exams. Most of the problems are of the very basic nature, reflecting the requirements to pass the course.

The problems are divided into 5 sections, corresponding to the five problem solving sessions scheduled for TDD 56 in the academic year 2006-2007. Each section indicates prioritized problems to be solved in the first place, preferably during the session. Each section indicates also a number of selected problems in the course book.

Le 1 – Complexity, Analysis of Algorithms

Prioritized problems: 1, 6, 8, 9, 10, 12, 13

Textbook problems: R-4.11-19, R-4.22-30

1. Prove or disprove the following statements by using the definitions, and by the asymptotic notation technique.

- (a) \((n + 1)^3 \in O(n^2)\)
- (b) \((n - 1)^3 \in O(n^2)\)
- (c) \(2^{n+1} \in O(2^n)\)
- (d) \(3^{n-1} \in O(2^n)\)
- (e) \(2 \cdot \sin(n) \in O(1)\)
- (f) \((n + 1)^2 \in \Omega(n^2)\)
- (g) \((n - 1)^3 \in \Omega(n^2)\)

2. Consider the following functions, where \(k \geq 4\) is a constant, and \(\log\) is the logarithm of base 2. Order them according to their growth rate, i.e., put them in the sequence \(f_1, \ldots, f_k\), such that \(O(f_i) \subseteq O(f_{i+1})\) for \(i = 1, \ldots, 7\). Does the set include functions with equal growth rate?

3. Let \(f(n)\) and \(g(n)\) be functions such that for all \(n\) \(f(n) \geq 0, g(n) \geq 0\). Show:

- (a) \(O(f(n) + g(n)) = O(\max\{f(n), g(n)\})\)
- (b) \(f(n)g(n) \in O(f(n)g(n))\)

4. Which inclusion relations, if any, hold between the following expressions:

- (a) \(O(\log_2 3n)\) and \(O(\log_2 2n)\)
- (b) \(O(\max 2m + 2)\) and \(O(n + 2)\)

5. Is it true that if \(\log f(n) \in O(\log g(n))\) then \(f(n) \in O(g(n))\)? Justify your answer.

6. Prove that if \(f \in O(g)\) and \(g \in O(h)\) then \(f \in O(h)\). What are \(n_0\) and \(c\) for \(f\) and \(h\) in terms of those for \(f\) and \(g\), and for \(g\) and \(h\)?

7. Show that \(f \in \Theta(g)\) iff \(g \in \Theta(f)\). (Thus \(\Theta\) actually partitions the class of all functions into equivalence classes.)
8. Algorithms A and B have worst case time complexity, respectively, \(O(n^2)\) and \(O(n \log n)\). On some data the running time of A is shorter than the running time of B. Give three different possible reasons of this phenomenon. Is this phenomenon still possible if the worst case time complexity are \(O(n^2)\) and \(O(n \log n)\)?

9. Provide justified answers to the following questions:
   (a) The worst case time complexity of an algorithm is \(\Omega(n^2)\). Is it possible that the algorithm executes in time \(T(n) \in \Omega(n^2)\) for some input data?
   (b) The worst case time complexity of an algorithm is \(\Omega(n^2)\). Is it possible that the algorithm executes in time \(T(n) \in \Omega(n^2)\) for every input data?
   (c) For the best case data the algorithm takes time \(\Omega(n)\). Is it possible that the algorithm executes in time \(T(n) \in \Omega(n^2)\) for some input data?
   (d) For the best case data the algorithm takes time \(\Omega(n)\). Is it possible that the algorithm executes in time \(T(n) \in \Omega(n^2)\) for every input data?

10. Analyse the time complexity of the following algorithms:
   (a) \textbf{procedure} Mystery\(\text{[integer } n]\): 
   \hspace{1cm} \text{for } i \text{ from } 1 \text{ to } n-1 \text{ do}
   \hspace{2cm} \text{for } j \text{ from } i+1 \text{ to } n \text{ do}
   \hspace{3cm} \text{an instruction that runs in time } \(O(1)\)
   
   \hspace{1cm} \text{for } k \text{ from } 1 \text{ to } j \text{ do}
   \hspace{2cm} \text{an instruction that runs in time } \(O(1)\)

   (b) \textbf{procedure} VeryOdd\(\text{[integer } n]\):
   \hspace{1cm} \text{for } i \text{ from } 1 \text{ to } n \text{ do}
   \hspace{2cm} \text{if } i \text{ is odd then}
   \hspace{3cm} \text{for } j \text{ from } i \text{ to } n \text{ do}
   \hspace{4cm} x \leftarrow x + 1
   \hspace{3cm} \text{for } j \text{ from } 1 \text{ to } i \text{ do}
   \hspace{4cm} y \leftarrow y + 1

   (c) \textbf{procedure} Complicated\(\text{[integer } n]\):
   \hspace{1cm} \text{for } i \text{ from } 1 \text{ to } n \text{ do}
   \hspace{2cm} \text{for } j \text{ from } n-1 \text{ downto } 3 \text{ do}
   \hspace{3cm} \text{for } k \text{ from } 1 \text{ to } 5 \text{ do}
   \hspace{4cm} \text{call } (\text{a procedure running in time } O(\log n))

   (d) \textbf{procedure} PrintStars\(\text{[integer } n]\):
   \hspace{1cm} a \leftarrow 1
   \hspace{2cm} \text{for } i \text{ from } 1 \text{ to } n \text{ do}
   \hspace{3cm} a \leftarrow 1
   \hspace{2cm} \text{for } j \text{ from } 1 \text{ to } a \text{ do}
   \hspace{4cm} \text{print } "*"
   \hspace{3cm} \text{new line}


11. Consider the following procedure:
   \textbf{procedure} foo\(\text{[integer } n]\):
   \hspace{1cm} \text{for } i \text{ from } 1 \text{ to } 4 \text{ do}
   \hspace{2cm} x \leftarrow 1

Which of the following statements hold for its running time \(T(n)\):
\(T(n) \in O(\log n), T(n) \in \Theta(n), T(n) \in \Omega(n), T(n) \in \Theta(n)\)? Justify your answers.

12. Analyse the worst case time complexity and the space complexity of the following algorithm, where input data consists of an integer \(i\) and a sorted table of integers of length \(n\).

   \begin{verbatim}
   function \text{integer } x, \text{table } A[1..n]: boolean
   first \leftarrow 1
   last \leftarrow n
   found \leftarrow false
   while first \leq last and not found do
      index \leftarrow \lfloor (first + last)/2 \rfloor
      if i = A[index] then found \leftarrow true
      else if i < A[index] then last \leftarrow index - 1
      else first \leftarrow index + 1
      return found
   \end{verbatim}

13. Write a recurrence equation characterising the time complexity of the following program:

   \begin{verbatim}
   function \text{Factorial}([integer } n]: \text{integer}
   if \(n = 0\) then return 1
   else return \text{Factorial}(n - 1)
   \end{verbatim}

14. Analyse the time complexity and the space complexity of the following algorithm:

   \begin{verbatim}
   procedure PrintC\(\text{[integer } n]\):
   array A[1..n]
   for i from 1 to n do
      A[i] \leftarrow 0
   loop1:
      i \leftarrow 1
      for j from 1 to n do
         print A[i]
      loop2:
      A[i] \leftarrow A[i] + 1
      if A[i] = 2 then
         A[i] \leftarrow 0
         i \leftarrow i + 1
      else
         exit loop2
      if i = n then
         exit loop1
   \end{verbatim}
Le 2 – Stacks and Queues, Hashing, Trees

Prioritized problems: R-5.3, R-5.6, 1.2, 3, 5-6, R-9.5, 7, 12

Textbook problems: R-5.1-3, R-5.6-9, R-9.5-6, R-9.8

1. The sequence X U Y U V W is read one character by character from left to right. Each character read may be first placed in a temporary memory or sent directly to the output. The reading operations may be interleaved with operations removing characters from the memory and sending them to the output. In this way we obtain at the output a permutation of the input sequence (the leftmost character of the permutation is the character sent first to the output). Consider the permutations X U W V Y and Y U W X V. For each of them check whether it is possible to obtain it if the temporary memory is
   (a) stack
   (b) queue

   If the answer is “yes” show the sequence of the respective operations performed to obtain it. If the answer is “no” explain why no sequence of the operations can generate the required permutation.

2. (a) Explain how to implement two stacks in one table T[0..n] in such a way that neither stack overflows unless the total number of elements in both stacks together is n + 1.
   The stack operation should run in O(1) time.
   (b) A queue Q is implemented as a ring buffer with 3 elements. Show the states of the buffer after each operation of the execution of
   Enqueue(Q,e), Enqueue(Q,b), Enqueue(Q,a), Dequeue(Q), Enqueue(Q,d)

3. A string is a palindrome if it is the same as its reverse (e.g., the string “noon”). Design an algorithm that uses a stack to check whether a sequence of n symbols read from the input is a palindrome or not. What is the running time of your algorithm.

4. Describe how to implement the ADT queue using two stacks.
   (a) Explain the idea.
   (b) Describe it in pseudocode.
   (c) Assume that you enqueue and dequeue n data items. Thus altogether we perform n enqueue operations and n dequeue operations. The order of these operations is such that the queue becomes empty only after the last dequeue operation. What is the worst case time for a single enqueue operation and a single dequeue operation?
   (d) What is the amortized time of all these operations?

5. A double-ended queue (deque) is a sequence that can be modified by adding and removing elements both at the front and at the end of the sequence. Thus, we have the following four abstract operations:
   • addFront(E, D) adds the element E at the front of the deque D,
   • deleteFront(D) returns the first element of the deque D and removes it from D.
   • addEnd(E, D) adds the element E at the end of the deque D,
   • deleteEnd(D) returns the last element of the deque D and removes it from D.

   Explain how deque can be represented in contiguous memory so that each of the four operations takes only a constant time.

6. We use an array with indices 0 to 6 to implement a hash table of size 7. The keys of the inserted elements are integers.

   The hash value is calculated as the key value modulo the table length.

   Show the contents of the initially empty array after performing each of the following operations
   Insert(15, e), Insert(8, a), Insert(14, b), Delete(10), Insert(32, d), Insert(4, c), Insert(7, f)

   when the following technique is used:
   (a) chained hashing,
   (b) open addressing with linear probing,
   (c) open addressing with double hashing, where h(k) = 5 - k mod 5

   The deletion is to be handled by the delete bit technique.

7. Consider hashing based on open addressing with linear probing. What is the disadvantage of using the deleted bit marking technique instead of the delete technique based on re-hashing of the probe sequence? Illustrate both techniques on an example.

8. The police plans to implement a register of stolen cars in form of a hash table using the car identification numbers as keys. Each number consists of three letters followed by three digits. Assume that all combinations appear with equal probability. The value of the hash function is determined by two characters of the number. Which of the following variants of the hash function give the best distribution in the hash table:
   (a) The last two letters,
   (b) The last two digits,
   (c) The last letter and the last digit.

9. Consider the following binary tree.

   ![Binary Tree Diagram]

   Which of the four "usual" kinds of binary tree traversals visits the nodes in the alphabetical order of the labels?
10. Consider the following tree traversal algorithm, where the argument \( n \) refers to the root of the tree.

\[
\text{procedure TraverseTree}(n: \text{tree node})
\]

\[
\begin{align*}
\text{var} & \quad S: \text{ADT Stack} \\
& \quad \text{MakeEmptyStack}(S) \\
& \quad \text{Push}(n) \\
& \quad \text{while not IsEmptyStack}(S) \quad \text{do} \\
& \quad \quad n = \text{Pop}(S) \\
& \quad \quad \text{Print label of } n \\
& \quad \quad \text{foreach child } c \text{ of } n \text{ in reverse order do} \\
& \quad \quad \quad \text{Push}(c)
\end{align*}
\]

(a) Which of the traversal orders is implemented by this algorithm?

(b) Transform the algorithm, so that it uses ADT queue instead of ADT stack. Which of the traversal orders is implemented by the transformed algorithm?

11. Show which of the following binary trees are

(a) full

(b) complete

(c) perfect

\[
\begin{align*}
a & \quad b & \quad c & \quad d \\
e & \quad f & \quad g & \quad h
\end{align*}
\]

12. Show the array-list representation of each of the following binary trees.

\[
\begin{array}{c}
3 & 4 & 5 \\
\end{array}
\]

\[
\begin{array}{c}
6 & 7 & 8 \\
\end{array}
\]

\[
\begin{array}{c}
9 & 10 & 11 \\
\end{array}
\]

Le 3 – Search Trees, Heap, Union/Find

Prioritized problems: R-10.4, R-10.6, R-10.7, R-10.9-10, R-10.23-24

1. Consider the following binary tree

\[
\begin{array}{c}
15 \\
6 & 21 \\
5 & 10 & 18 & 25
\end{array}
\]

(a) Justify that this is a binary search tree.

(b) Show the trees: \( T_1 = \text{Insert}(12, T) \), \( T_2 = \text{Insert}(7, T_1) \), \( T_3 = \text{Insert}(9, T_2) \), \( T_4 = \text{Delete}(15, T_3) \).

(c) Is this also an AVL tree?

2. Consider the sequence of keys obtained by a traversal of a binary search tree. Is it possible to reconstruct the tree, provided that the traversal was:

(a) preorder

(b) postorder

(c) inorder

(d) levelorder

3. Which of the following tree topologies can appear in correct AVL trees?

\[
\begin{array}{c}
A & \text{B} & \text{C} & \text{D} \\
E & F
\end{array}
\]

4. Is it true that every AVL tree of height \( h \) has less nodes than an AVL tree of height \( h + 1 \)? Prove or disprove.
5. (a) Which of the following are AVL trees? Justify your answers.

(b) Consider the following AVL tree:

Show step by step how the operations Insert(41), Insert(54), Delete(68) are performed on the tree above (all operations are applied to the same tree). What notations are performed?

(c) i. Insert 4 in the following splay tree:

ii. Discuss advantages and disadvantages of splay trees with respect to AVL trees.

6. Following the definition of (2,4) tree in p. 455 and the definition of (a,b) tree in p.455 of the course book define a notion of (2,3) tree. Give an example of a (2,3) tree of height 2, with all internal nodes being 3-nodes, and show how an example insertion operation will be performed on it. Discuss representation of (2,3) trees in the memory and analyze the time complexity of insertion.

7. Consider the following tables:

\[ A = \begin{bmatrix} 1 & 2 & 1 & 2 & 2 & 1 & 2 & 2 & 2 & 1 \end{bmatrix} \]

\[ B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix} \]

Which of them, if any, can be a heap?

8. Extend the notion of complete tree to ternary trees, where each node has at most three children. Define an array-list representation of such trees and show how to access the children of a node in \( O(1) \) time.

9. Consider the usual heap-based implementation of the ADT priority queue with DeleteMin operation. In this implementation there is no efficient way for accessing and deleting the maximal element of the queue. Design an algorithm which transforms the heap of a priority queue so that the maximal element can be efficiently accessed and deleted. Analyze the time- and space complexity of your algorithm.

10. Consider the following disjoint sets represented by up-trees (p.230 of the textbook):

Show how this tree is transformed by the following sequence of Find and Merge operations (called in the course book PathCompress(Find and Union-Ap-Siz): Find(2); Merge(A,C); Find(4); Find(1))
Le 4 – Sorting, Selection

Prioritized problems: 1, 2, 3, 4, 5, 7, 9, R-11.9, R-11.10, 11, 12

Textbook problems: R-11.9-11, R-11.17-19, R-11.21, R-11.25

1. For the following sequence of integers:
   \[ 7, 4, 12, 3, 2 \]
   Illustrate the operation of:
   (a) Insertion Sort,
   (b) Heap Sort,
   (c) Replace Quick Sort.

2. The following table is submitted as input data to a sorting algorithm.

   | 3 | 6 | 5 | 1 | 2 | 1 |

   At some stage of sorting the table is in the following form:

   | 1 | 2 | 3 | 4 | 5 | 6 |

   Could the algorithm be Insertion Sort, Selection Sort or Heap Sort?

3. An array of records indexed 1 to \( n \) provides information about students. Each record includes, among others, the name of a student and the information if the student is a male or a female. The array is sorted according to the names. Design an algorithm running in time \( O(n) \), which puts all female students in alphabetic order of their names before the male students in the alphabetic order of their names.

4. (a) Illustrate the operation of Radix Sort on the following table of keys, where the characters of the keys form the subkeys for sorting.

   \[ \text{John, Mary, Jane, ...} \]

   You need only to show the form of the table after each phase of subkey sorting.

   (b) Give a motivation to why Bucket Sort is less appropriate when the key universe is large (as in the above example).

   (c) Is Radix Sort stable? Justify or give a counterexample.

5. Consider the following sorting algorithm, called \textit{Bubble sort}.

   \[
   \text{procedure BubbleSort(A[1..n])}
   \text{for } i \text{ from 0 to } n-2 \text{ do}
   \text{for } j \text{ from } i+1 \text{ downto 1 do}
   \text{if } A[j] < A[j-1] \text{ then}
   \text{A[j] \leftrightarrow A[j-1]}
   \]

   Is it stable? What is its worst-case time complexity?

6. (a) What is the worst-case data for Insertion Sort?

   (b) We want to sort singly linked list. Why should Quick Sort not be our first choice of sorting algorithms?

   (c) Any comparison-based sorting algorithm takes time \( \Omega(n \log n) \) in the worst case. What can we conclude about the best-case time for such algorithms? Justify your answer.

7. (a) The phases of Radix Sort consist in sorting of the consecutive fragments of the keys, starting from the least significant ones. What would be a disadvantage when starting with more significant fragments?

(b) What requirement should be satisfied by Bucket Sort to use it in Radix Sort with phases starting from the least significant fragments of the keys? Discuss, how to implement Radix Sort with a variant of Bucket Sort which does not fulfill the requirement.

8. We want to sort \( n \) strings of maximal length \( m \) characters and average length of \( \frac{m}{2} \) characters (the average was taken over the possible sets of strings, not over a particular input set).

   (a) Show the worst-case time complexity for Quick Sort and Radix Sort. Is the running time of one of these algorithms better than the other on all worst-case data?

   (b) Repeat the analysis for the average case time complexity.

9. Quick Sort is not efficient for small tables. Therefore it is often combined with other sorting algorithms. For example, consider the following variants of Quick Sort:

   A. We make Quick Sort to ignore all subtables of size \( \leq k \), for some parameter \( k > 0 \), i.e., such subtables are left unsorted. When this algorithm terminates the table is ‘almost sorted’ and is passed to Insertion Sort for final sorting.

   B. When Quick Sort is called for a subtable of size \( \leq k \), Insertion Sort is called instead for this subtable. Thus, the recursive calls to Quick Sort are restricted to the tables of size greater than \( k \).

   For each of the variants above find an expression describing how much time is totally used by the Insertion Sort, as a function of \( k \) and size \( n \) of the sorted table. Which expressions change if we replace Insertion Sort by Selection Sort?

10. Telephone directories sorted in the alphabetic order of names should be produced from collections of raw data sorted by phone numbers. Assuming that all data can be placed in the internal memory use the time complexity arguments to justify whether Quick Sort or Radix Sort are better suited for this problem.

11. For each of the following algorithms discuss what are the respective best case and worst case data and illustrate them by examples for \( n = 5 \):

   - Insertion Sort
   - Quick Sort
   - Quick Select

12. Explain whether each of the following statements is true or false

   - The best case running time of Insertion Sort is \( O(n) \).
   - The worst case complexity of Quick Sort is \( O(n^2) \).
   - The worst case complexity of Quick Select is \( O(n) \).

11

12
Le 5 – Graphs

Prioritized problems: 1, 2, 3, R-13.7, R-13.8, 4, 5, 7


1. In a search starting in vertex A of the following graph the vertices were visited in the order indicated by the numbers.

![Graph Image]

Could it be depth-first search or breadth-first search?

2. Consider the following directed graph with weighted edges

![Directed Graph Image]

(a) Show the order of visiting the nodes by depth-first search from vertex c; the neighbours are selected in the increasing weights of the connecting edges.
(b) Show the order of visiting the nodes by breadth-first search from vertex c; the neighbours are selected in the increasing weights of the connecting edges.
(c) Show a Topological Sort of the graph.

3. Assume that the vertices of a Directed Graph are numbered; the number of a vertex is called index. The indices impose an ordering on every subset of vertices. Consider the following operations of ADT Directed Graph:

- `InsertEdge(v, w, G)` insert the edge (v, w) in G
- `DeleteEdge(v, w, G)` delete the edge (v, w) in G
- `FirstIncG(v, G)` = index of v's first neighbour (empty if v has no neighbours)
- `NextInc(i, G)` = the minimal index of a neighbour of v greater than i (empty if there is no such neighbour).
- `IsIndex(i, G)` = true if i is an index different from the empty index
- `Vertex(i, G)` = vertex determined by index i
- `ExistEdge(v, w, G)` = true if the edge (v, w) is in G

Find lower and upper bounds of the asymptotic time complexity of these operations for graphs represented as adjacency matrices and as adjacency lists. To characterize argument graphs use the following parameters:

- v: number of vertices,
- e: number of edges,
- k: maximal number of edges outgoing from a vertex

Analyze the time complexity of the abstract instruction:

```
foreach neighbour u of v do
```

Hint: Implement the instruction using some of the graph operations above.

4. Design an algorithm that checks existence of cycles in undirected graphs. The algorithm should work in time \(O(n + e)\), where \(n\) is the number of vertices and \(e\) is the number of edges of the graph.
(a) Sketch the idea.
(b) Justify that the algorithm has the required complexity.
(c) Explain how to extend your algorithm so that it can check whether the input graph is a tree.

5. A study curriculum includes the following courses: Discrete Math (DM), Compiler Construction (CC), Data Structures (DS), Programming 1 (P1), Programming 2 (P2), Theory of Programming (TP), Analysis of Algorithms (AA), Formal Languages (FL) and Optimization Techniques (OT). Each of the courses has some (possibly none) other courses as prerequisites for admission. They are listed below:

- DM prerequisites: none.
- P1: none.
- DS: TM, P1.
- TP: DM, P1.
- AA: DS, P1.
- FL: DM, DS.
- CC: P2, FL.
- P2: P1, TP.
- OT: DS, P2.

(a) Represent the information as a graph \(G\) whose vertices correspond to courses and prerequisites correspond to edges. What kind of graph is it?
(b) Johan is a part-time student. He can only take one course at a time. Use a well-known graph algorithm to determine a study plan for Johan. Show the consecutive steps of the computation performed on the graph \(G\).
(c) Each of the courses is taught during a whole term. Several courses may be taught in the same term provided that all prerequisites were offered in previous terms. What is the minimum number of terms needed to schedule all courses in the example curriculum? Justify your answer. Use the notion of path length to give a general answer to this question for an arbitrary graph of courses.
6. Modify DFS algorithm so that it can be used for computing a topological sort of a directed acyclic graph. Analyze complexity of your solution and show an example how it works.

7. Peter is building a holiday house. He has broken the project into the following tasks: preparing the ground, building exterior walls, building interior walls, building the chimney, building the roof, painting the interior walls, laying floor, setting doors, setting windows, covering the roof, insulating the exterior walls, moving in.

(a) There are some restrictions on the ordering of these tasks:
   - The ground must be prepared before building the exterior walls or the chimney.
   - To build the roof Peter must first build the external walls and the chimney.
   - The interior walls are built after the exterior walls.
   - The windows are set after building the exterior walls.
   - The doors are set after building the interior walls.
   - The floor can only be laid when the roof is covered and the interior walls are built.
   - To cover the roof Peter must first build it.
   - The insulation of exterior walls can only be done when the windows and the doors are already set.
   - Painting of the interior walls can only be done when the floor is ready.
   - Peter will not move in until the interior walls are painted and the exterior walls are insulated.

Represent the above restrictions as a directed graph with vertices representing tasks and edges representing their ordering. The graph must not introduce more direct restrictions on task ordering than those stated above.

(b) Peter wants to perform each of the tasks during consecutive weekends. Which of the graph algorithms discussed in the course can be used to schedule the tasks in the way that all restrictions are observed. Explain how this algorithm works for your graph by discussing the intermediate stages of its operation and show the resulting schedule.