

Data Structures and Algorithms

Exam

Prof. Dr. M. Böhlen

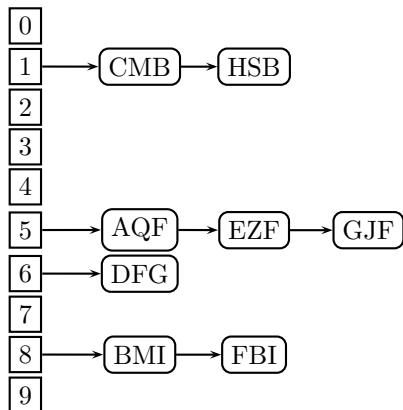
June 22, 2006

14:00 - 17:00

The exam consists of 6 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. As auxiliary material you may use 1 A4 sheet with notes.

1 Hashing

Consider the following hash table with linear chaining as a collision resolution strategy.



1. Show the positions of the elements if open addressing with linear probing is used as a conflict resolution strategy.
2. Is your solution unique? If yes argue for your answer. If no argue for your answer and show an example of another possible organization of the elements.
3. Give algorithms to insert and delete elements into the hash table with open addressing and linear probing as a conflict resolution strategy.

2 Linked Lists

Assume a double-linked non-circular list of integers. The order of the elements in the list is not relevant. A list element is defined as follows:

```
struct List {
    int key;
    struct List* prev;
    struct List* next;
}
```

A list can be accessed through a pointer to the head and end, respectively. Assume two lists. List $L1$ is instantiated as shown in Figure 1.

```
struct List* h1;
struct List* t1;

struct List* h2;
struct List* t2;
```



Figure 1: Original Double-linked List $L1$

Develop a fragment of C code that splits a double-linked list by removing from $L1$ all elements with key values less than or equal to Max . All elements less than or equal Max shall be put into a new double-linked list $L2$, while all other elements shall remain in the original list $L1$. For instance, with $Max = 9$ and list $L1$ from Figure 1, the two lists displayed in Figure 2 are produced.

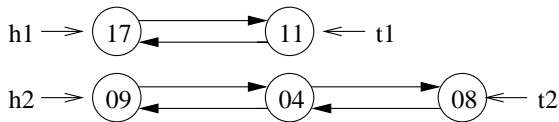


Figure 2: Resulting Double-linked Lists $L1$ and $L2$

Design your algorithm so that no memory is allocated or deallocated. Illustrate your algorithm by showing the lists at different points during the computation.

3 Algorithmic Complexity

1.1. Prove or disprove the following statements about the asymptotic complexity. It is important that you argue for your answer and that you show all steps to arrive at the solution.

- $2^{n+1} = O(n^2)$
- $3^n = O(3^{n+1})$
- $2n^{1.5} = O(n \cdot \log_a n)$

1.2. Solve the following recurrences and determine the simplest $g(n)$ such that $T(n) = \Theta(g(n))$. It is important that you argue for your answer and that you show all steps to arrive at the solution.

- a. $T(n) = 15T(n/4) + n^2$
- b. $T(n) = 6T(n/2) + n^2$
- c. $T(n) = 9T(n/3) + n^2$

4 Dynamic Programming

A paragraph consists of a sequence of words that are separated by spaces. An optimal wrapping splits the paragraph into lines as follows (splits are only allowed between the words): The number of characters per line may not exceed the maximum line length Max and the number of extra whitespace characters is minimal.

We assume an array $w[1..n]$ that holds the words of the paragraph, and a function $lgth$ that returns the number of characters in a word. Thus, the number of extra whitespace characters that are needed on a line with words i through j is defined as:

$$e[i, j] = Max - (j - i) - \sum_{k=i}^j lgth(w[k])$$

We use the number of extra whitespace characters to define the cost of a line holding words i through j . The line cost is infinite if the number of characters exceeds the length of the line. The cost of the last line is zero because the last line does not have to be padded with whitespace.

$$lc[i, j] = \begin{cases} (e[i, j])^2 & \text{if } e[i, j] \geq 0 \text{ and } j < n \\ \infty & \text{if } e[i, j] < 0 \\ 0 & \text{if } e[i, j] \geq 0 \text{ and } j = n \end{cases}$$

Consider the text “this is a quick fox circumventing malicious pitfalls” and its organization into a paragraph with three lines as shown below. At the end of each line we state the number of extra whitespace characters required for the line and the line cost. In this example $Max = 19$.

```
this is a quick fox      (extra = 0, line cost = 0)
circumventing           (extra = 6, line cost = 36)
malicious pitfalls      (extra = 1, line cost = 1)
```

- a. Illustrate the computation of the number of extra whitespace characters ($e[i, j]$), the line cost ($lc[i, j]$), and the computation of the optimal line breaks. Use the text given above as a concrete example.
- b. What happens if in the definition of the line cost $(e[i, j])^2$ is changed to $e[i, j]$?
- c. Give a dynamic programming solution to determine the optimal line breaks.
- d. What is the asymptotic complexity of your algorithm?

5 Graphs

A *separation edge* of G is an edge whose removal disconnects G . Write an algorithm that finds all separation edges in a given undirected graph.

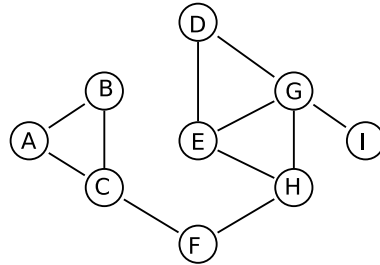
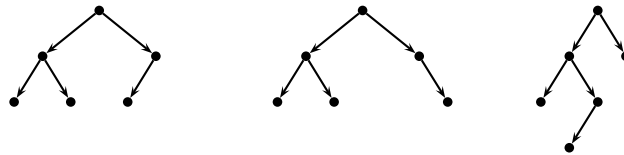


Figure 3: Example Graph

- Identify the separation edges of the graph in Figure 3.
- Give an algorithm that finds all separation edges in an undirected graph.
- What is the complexity of your algorithm?

6 Binary Trees

A *nearly complete binary tree* (NCBT) is a binary tree where the depth of two leaves differs by at most 1 and all leaves are as far left as possible. For example, in the figure below leftmost tree is a nearly complete binary tree, whereas the other two trees are not nearly complete binary trees.



Assume a tree with a node structure defined as follows:

```
struct Tree {  
    int key;  
    struct Tree* lChild;  
    struct Tree* rChild;  
}
```

- Write a function `int isNCBT(Tree* pTree, int n)` that returns 1 if `pTree` is a nearly complete binary tree and 0 otherwise. Explain the idea of your algorithm.
- What is the complexity of your algorithm. Explain your reasoning.