

Name: \_\_\_\_\_

Grade: \_\_\_\_\_

### **Exam 1**

CPSC 411 Analysis of Algorithms  
Andreas Klappenecker

- This exam contains 12 problems. You have 60 minutes to earn up to 100 points.
- This exam is closed book.
- You are allowed to use a nonprogrammable calculator, although you will not need one.
- Do not spend too much time on a single problem. Read them all through first and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Recall the Aggie code of honor. Cheating will have severe consequences.

The work shown in this exam is my own.

Signature required: \_\_\_\_\_

**Problem 1** (2 Points)

Write (print) your name on each odd-numbered page.

**Problem 2** (5 Points)

State the definition of  $f(n) \in O(g(n))$  for some functions  $f, g: \mathbf{Z} \rightarrow \mathbf{R}$  (assuming that the argument  $n$  approaches  $\infty$ , as usual).

**Problem 3** (3 Points)

True or false. Is  $4n^2 + 2n + 4 \in O(n^2 + n + 1)$ ?

**Problem 4** (5 Points)

What is the purpose of using a Huffman code?

**Problem 5** (5 Points)

Derive the Huffman tree using Huffman's algorithm for the alphabet  $A = \{a, b, c, d\}$ , when the frequencies are given by

$$f(a) = 1000, f(b) = 3000, f(c) = 2000, \text{ and } f(d) = 5000.$$

The minimum priority queue  $Q$  has initially the state 1) and after each iteration of Huffman's algorithm the states 2), 3), and 4).

1)  $Q =$

2)  $Q =$

3)  $Q =$

4)  $Q =$

**Problem 6** (5 Points)

Peter claims that Prim's algorithm to compute a minimum spanning tree of a connected graph  $G$  is based on the greedy algorithm for matroids. Is Peter right?

**Problem 7** (5 Points)

Which key property is shared by both greedy algorithms and dynamic programming?

**Problem 8** (5 Points)

In the matrix chain order problem, an array  $M$  is maintained to calculate the cost of multiplying the  $i$ th through  $j$ th matrix, that is, the cost of the matrix product  $A_i \cdots A_j$ . The array is initialized in the as follows:

	1	2	3	4
1	0			
2	n/a	0		
3	n/a	n/a	0	
4	n/a	n/a	n/a	0

Explain in which order the remaining entries of the table are generated in the algorithm given in the lecture.

**Problem 9** (5 Points)

on an array  $a[1..n]$ , where  $n$  is some power of 2. The algorithm recursively calls  $DC(a[1..n/2])$  and  $DC(a[n/2+1..n])$  and combines the results in  $\Theta(n)$  time. What is the time complexity (in Big Theta) of the algorithm? Give an example of such an algorithm that was discussed in the lecture.

**Problem 10** (20 Points)

On a string  $s$ , we have the following elementary operations:

- i) *Insertion* of a single letter into the string  $s$ , e.g.,  $BT \rightarrow BAT$ .
- ii) *Deletion* of a single letter in the string  $s$ , e.g.,  $CATE \rightarrow CAT$ .
- iii) *Substitution* of one letter in the string  $s$  by another one, e.g.,  $BAT \rightarrow CAT$ .

The **Levenshtein distance**  $D(s, t)$  between two strings  $s$  and  $t$  is equal to the smallest number of operations i), ii), or iii) to transform the string  $s$  into  $t$ .

Let  $m$  denote an array where the entry  $m[i, j]$  contains the Levenshtein distance of the initial segments  $s[1..i]$  and  $t[1..j]$  of the strings  $s$  and  $t$ , respectively.

(a) Determine the entries and justify your claim:

(a)  $m[i, 0] =$

(b)  $m[0, j] =$

(b) The following facts are easily established:

(F1) If we can transform  $s[1..i]$  to  $t[1..j - 1]$  in  $k$  operations, then we can simply add  $t[j]$  afterwards to get  $t[1..j]$  in  $k + 1$  operations.

(F2) If we can transform  $s[1..i - 1]$  to  $t[1..j]$  in  $k$  operations, then we can do the same operations on  $s[1..i]$  and then remove the original  $s[i]$  at the end in  $k + 1$  operations.

(F3) If we can transform  $s[1..i - 1]$  to  $t[1..j - 1]$  in  $k$  operations, we can do the same to  $s[1..i]$  and then do a substitution of  $t[j]$  for the original  $s[i]$  at the end if necessary, requiring  $k$  or  $k + 1$  operations.

These three facts suffice to determine the value of  $m[i, j]$  from the entries  $m[i, j - 1]$ ,  $m[i - 1, j]$ , and  $m[i - 1, j - 1]$ . Derive the formula for  $m[i, j]$  and justify your result. Explain how the two subcases in F3 are resolved.

$m[i, j] =$

(c) What is the complexity of any dynamic programming approach based on parts a) and b), assume that  $s$  and  $t$  are of length  $a$  and  $b$ . Use Big Omega notation.

**Problem 11** (20 Points)

Recall that a matroid  $M = (S, \mathcal{L})$  consists of

- i) a nonempty, finite set  $S$ ;
- ii) a nonempty family of subsets  $\mathcal{L}$  such that  $B \in \mathcal{L}$  and  $A \subseteq B$  implies  $A \in \mathcal{L}$ ;
- iii) if  $A, B \in \mathcal{L}$  and  $|A| < |B|$ , then there exists an  $x \in B \setminus A$  such that  $A \cup \{x\} \in \mathcal{L}$ .

a) Prove that all maximal sets in  $\mathcal{L}$  have the same cardinality.

b) Prove that  $(S, \mathcal{L}_k)$  is a matroid, where  $S$  is a finite, nonempty set and  $\mathcal{L}_k$  is the set of all subsets of  $S$  of size at most  $k$ , where  $k$  is some nonnegative integer that does not exceed  $|S|$ .

**Problem 12** (20 Points)

Suppose that we have a binary counter with  $k$  bits, where each bit is stored in an array  $A[0..k-1]$ . The operation **Increment**( $A$ ) increments the counter by adding 1 to the content of the counter modulo  $2^k$ . The cost of this operation is proportional to the number of bits that need to be inspected, so it is at most  $O(k)$ .

a) Prove that if a sequence of  $n$  increment operations is applied to a counter that is initially zero, then at most  $2n$  bits are flipped, by counting the number of bit flips. [Hint: Note that  $A[0]$  flips every time,  $A[1]$  flips every other time, etc.]

b) Use now the accounting method to prove that at most  $2n$  bits are flipped, when a sequence of  $n$  increment operations is executed on a counter that is initially zero.