Introduction to Computer Science
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CH-232-A
Date: 2020-09-11
Due: 2020-09-18

ICS 2020 Problem Sheet #1

Problem 1.1: *minimum spanning trees* (3 points)

We have introduced Kruskal's algorithm for constructing random spanning trees (maze solutions). Edges are selected randomly and added to the spanning tree as long as the nodes connected by the edges belong to different equivalence classes. The original algorithm solves a slightly more difficult problem: Given a graph $G = (V, E)$ and a cost function $c : E \to \mathbb{R}$ that indicates the cost of including the edge $e \in E$ in the spanning tree, calculate the spanning tree $G' = (V, E')$ such that $C = \sum_{e \in E'} c(e)$ is minimal (also called a minimum spanning tree). Kruskal's algorithm solves this problem by selecting in each step an edge that joins two equivalence classes and has the minimum cost of all edges still available.

You are given the graph $G = (V, E)$ with

$$V = \{a, b, c, d, e, f\}$$
$$E = \{(a, b), (a, e), (a, f), (b, c), (b, f), (c, d), (c, f), (d, e), (d, f), (e, f)\}$$

and the cost function $c : E \to \mathbb{R}$ as defined by the following table:

edge:	(a,b)	(a,e)	(a,f)	(b,c)	(b,f)	(c,d)	(c,f)	(d,e)	(d,f)	(e,f)
cost:	7	5	1	2	6	6	5	3	4	5

Construct a minimal spanning tree $G(V, E')$ using Kruskal's algorithm. For each step, write down the set of equivalence classes A and the edges in E'. What is the overall cost C of the resulting spanning tree? You start with:

$E' = \{\}$ start, $C = 0$
$A = \{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}\}$
$E' = \ldots$ step 1, $C = \ldots$
$A = \ldots$
$\ldots$

Problem 1.2: *boyer moore algorithm* (2+2+2 = 6 points)

You have designed a simple robot that can turn left (L), turn right (R), move one step forward (F), and pause (P) for short time. The robot is programmed by a sequence of robot instructions. For example, the sequence FFLFLFRFRFFLFRF will direct the robot through the maze shown on the slides discussing maze generation algorithms. Using the Boyer Moore algorithm, we can determine whether a robot program contains certain movement sequences.

Let $\Sigma = \{L, R, F, P\}$ be an alphabet and $t \in \Sigma^*$ be a text of length n describing a program for the robot. Let $p \in \Sigma^*$ be a pattern of length m. We are looking for the first occurrence of p in t.

Consider the text $t = FFLFLFRFRFFLFRF$ and the pattern $p = FFLFR$.

a) Execute the naive string search algorithm. Show all alignments and indicate comparisons performed by writing uppercase characters and comparisons skipped by writing lowercase characters. How many alignments are used? How many comparisons are done?

b) Execute the Boyer-Moore string search algorithm with the bad character rule only. How many alignments are used? How many comparisons are done?

c) Calculate the lookup table for the bad character rule that indicates the number of alignments that can be skipped if a comparison does not match.

Problem 1.3: *operator precedence and associativity (haskell)* (1 point)

Haskell operators have associativity and precedence. The associativity defines in which order operators with the same precedence are evaluated while the precedence defines in which order operators with different precedence levels are evaluated (higher precedence level first).

a) Some operators are neither left nor right associative. What happens if such operators appear multiple times in an expression (without additional parenthesis defining the evaluation order)? Provide an example and an explanation.

b) Haskell has a very special operator `$`. What is the precedence and associativity of this operator? Write the following prefix expression

```
(^) 2 $ (*) 5 $ (+) 2 3
```

in infix notation without the `$` operator, using parenthesis where necessary.