Midterm Examination 1
GEM 1501: Problem Solving for Computing

Wednesday 02.03.2011, duration half an hour

Solutions

Rules
This test carries 12 marks and consists of 6 questions. Each questions carries 1 to 3 marks; full marks for a correct solution; a partial solution can give a partial credit.

Question 1 [2 marks].
Punch cards have been used in computing to feed computers with data or programs. Describe how punch cards were used in the 19th century: What machines were invented by Joseph Jacquard and by Herman Hollerith, which used the punch cards? For what purpose needed they punch cards?

Solution. Joseph Jacquard invented a mechanical loom, that is a weaving machine, which produces cloth. This machine could follow various patterns and the weaver could program the patterns and their sequence with punch cards. The holes in the cards coded which colours and threads to take and where to place them.

The 1880 census in the USA took 7 years to evaluate. Therefore the United States Census Office called for a competition to run the census with machinery. Herman Hollerith presented a machine which used punch cards as a data medium and this machine produced statistical tables based on the data read from the punch cards. The machine was called a tabulating machine and won the competition. During the census, the officers recorded for each person the relevant data in a fixed pattern on a punch card and then they were able to evaluate the census with these machines reading the punch cards in six weeks. Later other countries also requested the services of Herman Hollerith for running a census and Herman Hollerith founded the Tabulating Machines Company which eventually merged with some other companies to form the International Business Machines Corporation (IBM) which still exists.

Question 2 [2 marks].
Call a list of n different pairs of numbers $(a_0, b_0), (a_1, b_1), \ldots, (a_{n-1}, b_{n-1})$ sorted iff there are no i, j with $i < j$, $a_i > a_j$ and $b_i > b_j$. An algorithm to sort the pairs can check whether $a_i < a_j$, $a_i = a_j$ or $a_i > a_j$; similarly the algorithm can check whether $b_i < b_j$, $b_i = b_j$ or $b_i > b_j$. Note that two pairs of numbers can be incomparable as for example $(1, 4)$ and $(2, 2)$. Traditional sorting algorithms do not address such data.

Is it nevertheless possible to sort all n pairs in time $O(n \log(n))$? Give reasons for your answer.

Solution. YES, one can still sort the pairs in time $O(n \log(n))$. Let $\sqsubset$ be the partial order above where $(a_i, b_i) \sqsubset (a_j, b_j)$ iff $a_i < a_j \wedge b_i < b_j$. There are incomparable pairs with respect to $\sqsubset$ and therefore it is better to use the lexicographic ordering where $(a_i, b_i) <_{lex} (a_j, b_j)$ iff $a_i < a_j \vee (a_i = a_j \wedge b_i < b_j)$. Note that $(a_i, b_i) \sqsubset (b_i, b_j)$ implies $(a_i, b_i) <_{lex} (a_j, b_j)$ but not vice versa; furthermore, the lexicographic ordering does not have incomparable pairs. Now one can use Mergesort to sort the pairs with respect to $<_{lex}$; note that one can find out with two comparisons whether $(a_i, b_i) <_{lex} (a_j, b_j)$. First one compares a_i and a_j; if they are equal than one also compares b_i and b_j. The sorted list produced by Mergesort is then also sorted with respect to $\sqsubset$, that is, it cannot happen that $i < j$ and $(a_j, b_j) \sqsubset (a_i, b_i)$; the reason is that by the sorting $(a_i, b_i) <_{lex} (a_j, b_j)$ and therefore $a_i < a_j$ or $b_i < b_j$.

There are also some other approaches to sort these pairs; nevertheless, please keep in mind that the pairs should stay together. So a sorted list of $(2, 8), (1, 7), (1, 9)$ is $(1, 7), (1, 9), (2, 8)$ but not $(1, 7), (1, 8), (2, 9)$. Sorting the first and second component independently does not solve the problem.

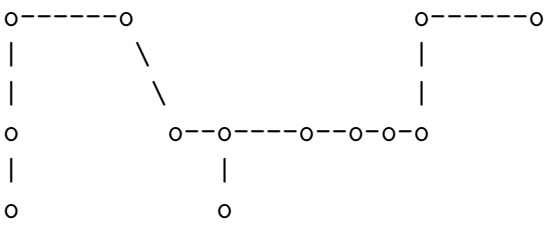
Question 3 [2 marks].
The railroad contractor problem asks for an algorithm to connect nodes in a network such that the resulting network on one hand connects all cities and on the other hand is as short as possible. This algorithm goes in general as follows: Starting with a network consisting of one node, it chooses in each step one node A outsidee the current network and links it to one node B in the network.

Describe how these two nodes A and B are selected? Draw the resulting network into the below graphic for the given points, where the algorithm starts in the left lower corner.

Solution. Node A is selected such that it is nearest to the currently already constructed network and node B is then the nearest node to A inside the network. One could do it like this: for all nodes A outsid the network and nodes B inside the network, determine the distance from A to B and then take that pair (A,B) for which

this distance is shortest. Connect A and B. This step is repeated so long until all nodes are in the network.

Note that it is permitted to have branches in the network. But there should not be any closed circles in the network.



Question 4 [3 marks].
The following Java Script function computes some number for an array called "list":

```
function sumone(list)
  { var n = list.length;
    var i,j; var sum = 0;
    for (i=0;i<n;i=i+1)
      { for (j=0;j<n;j=j+1)
          { sum = sum+list[i]*list[j]; } }
    return(sum); }
```

(a) What is the order of the running time of the program?

(b) Is there also a program doing the same in time $O(n)$?

(c) If no, explain why such a program cannot exist. If yes, please write the corresponding program below.

Solution. (a) The order of the running time is $O(n^2)$. There are two nested loops where in both cases the variables `i` and `j`, respectively, run from `0` to `n-1`.

(b) Yes, it is possible to make a program which runs in linear time. The essential idea is to use that

$$\sum_{i<n}\sum_{j<n} \text{list}[i] \cdot \text{list}[j] = (\sum_{k<n} \text{list}[k])^2.$$

(c) The program is the following.

```
function sumtwo(list)
  { var n = list.length;
    var k; var sum = 0;
    for (k=0;k<n;k=k+1)
      { sum = sum+list[k]; }
    return(sum*sum); }
```

Question 5 [2 marks].
Write a program which does the following: It counts how many numbers of the form `x*(x*x+3)` are between `0` and `y`. So if `y` is `14` then the answer should be `3`; the corresponding numbers are `0`, `4` and `14`. Here `y` is always a natural number, that is, `y` is an element of the set $\{0, 1, 2, 3, 4, \ldots\}$.

Solution.

```
function count(y)
  { var x = 0; var c = 0;
    while (x*(x*x+3) <= y)
      { x=x+1; c=c+1; }
    return(c); }
```

Question 6 [1 marks].
An NP complete problem is satisfiability. Is the following set of clauses satisfiable? Here the clauses are 1. $x_1 \vee x_2$; 2. $x_2 \vee x_3$; 3. $x_3 \vee x_4$; 4. $x_4 \vee x_5$; 5. $x_5 \vee x_1$; 6. $\neg x_1 \vee \neg x_2$; 7. $\neg x_2 \vee \neg x_3$; 8. $\neg x_3 \vee \neg x_4$; 9. $\neg x_4 \vee \neg x_5$; 10. $\neg x_5 \vee \neg x_1$.

Solution. NO, this set of clauses is not solvable. To see this, one tries to satisfy all clauses and will then see that one clause remains unsatisfied. So one starts with one truth-value for x_1. Then, for $m = 1, 2, 3, 4$, the clauses $x_m \vee x_{m+1}$ and $\neg x_m \vee \neg x_{m+1}$ can only be both true when x_{m+1} has the opposite truth-value than x_m so that x_m makes one clause and x_{m+1} makes the other clause true. It follows that x_{m+2} has the same truth-value as x_m for $m = 1, 2, 3$. In particular x_1, x_3, x_5 have the same truth-value. But then x_5 and x_1 are either both true and the clause $\neg x_5 \vee \neg x_1$ is not satisfied or they are both false and the clause $x_5 \vee x_1$ is not satisfied. Hence it follows that there is no satisfying assignment to make all clauses true.