Exam in EDAF05 Algorithms, Data Structures, and Complexity

August 20, 2018, 8-13

You may answer in English, på svenska, auf Deutsch, или по-русски.

Examiner: Jonas Skeppstedt

30 out of 60p are needed to pass the exam and your grade is \lfloor \frac{\text{your score}}{10} \rfloor.

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, or old exams. You can’t bring electronic aids (such as a laptop) or communication devices (such as a mobile phone). If you really want, you can bring a pocket calculator, but I can’t see how that would be of any use to you.

Filling out the exam. Some questions are multiple choice. Mark the box or boxes with a cross or a check-mark. If you change your mind, completely black out the box and write your answer’s letter(s) in the left margin. In case it’s unclear what you mean, I will choose the least favourable interpretation. In those questions where you have to write or draw something, I will be extremely unco-operative in interpreting your handwriting. So write clearly. If there is a way to misunderstand what you mean, I will use it.

Scoring. For the free form choice questions, you get between 0 and the maximum number of points for that question. Short, correct answers are preferred. You can, however, get a negative score on a multiple choice question. Each multiple choice question has exactly one correct answer. To get the maximum score for that question, you must check that answer, and only that. However, to reflect partial knowledge, you may check several boxes, in case you’re not quite sure (this lowers your score, of course — the more boxes you check, the fewer points you score). If you check no boxes or all boxes, your score for that question is 0. If the correct answer is not among the boxes you checked, your score is negative, so it’s better to not answer a question where you’re on very thin ice. The worst thing you can do is to check all boxes except the correct one, which gives you a large negative score. For example, assume a question worth maximum 2 points has \( k = 4 \) possible answers (one of them correct).

- If you select only the correct answer, you receive 2 points.
- If you select 2 answers, one of which is correct, you receive 1 point.
- If you select 3 answers, one of which is correct, you receive 0.41 points.
- If you select no answer or all answers, you receive 0 point.
- If you select only one answer, and it is wrong, you receive \(-0.67\) points.
- If you select 2 answers that are both wrong, you receive \(-1\) point.
- If you select 3 answers that are all wrong, you receive \(-1.25\) points.

As a special case, for a yes/no question, you receive 1, 0, or \(-1\) points, depending on whether your answer is correct, empty, or wrong. The precise formula is: if the question has \( k \) choices, and you checked \( a \) boxes, your score is \( \log(k/a) \), provided you checked the correct answer, and \(-a \log(k/a)/(k-a)\) if you only checked wrong answers. Moreover, I have weighted the questions by relevance (not necessarily difficulty), and indicated the maximum points with each question. If you really care why this scoring system makes sense, read Gudmund Skovbjerg Frandsen, Michael I. Schwartzbach: A singular choice for multiple choice. SIGCSE Bulletin 38(4): 34–38 (2006). For example, random guessing will give you exactly 0 points, at least in expectation.
Register

In optimizing compilers, one of the most important tasks is to assign variables to CPU registers, since it is much faster to read or write a variable in a CPU register than in memory. This is called register allocation. Two variables x and y cannot, obviously, be allocated the same register if their values are needed at the same time. For instance:

```c
int f(int x)
{
    int y;
    int z;

    y = x + 1;
    z = y * 2;
    return x * z;
}
```

Assume x is allocated R1. Since y is used (when computing z) while x is not yet finished with R1 (since x needs R1 in the return-statement), it is impossible to let y also use R1.

But y and z can both be allocated R2 without a problem. To see this, we can rename the variables as follows:

```c
int f(int r1)
{
    int r2;

    r2 = r1 + 1;
    r2 = r2 * 2;
    return r1 * r2;
}
```

Register allocation is thus the problem of mapping variables to registers while also taking into account which variable cannot use the same register. That two variables are used at the same time is said that they interfere and compilers represent this information with an interference graph in which variables are nodes and there is an undirected edge between two nodes if the corresponding variables cannot be allocated the same register. When we have the interference graph, variables and uses can be ignored and our only concern is to make sure two nodes which are neighbors in this graph, are not allocated the same register (this ignores the fact that there may be too few registers and what to do then).

Assume you are given an interference graph and that you are going to make a register allocator for a tiny CPU with only two registers. Your task is to determine if this is possible and if it is, map each variable to one of the registers R1 and R2.

Example 1

Input:

```
   x
      |
      v
       y
           |
           v
        z
```

Example output:

```
x in R1, y in R2, z in R2
```
Example 2

Input:

Output:

impossible
Squares

The Green Squares! That is what we need. They have that Red Square in Moscow but we will have the nicest green squares in the world. Listen now, we will take the biggest rectangle of grass from the White House, and make squares from it.

Really? I mean yes, of course, Mr. President, said the new advisor. How should we cut them?

That’s your job. I want to have a small number of very nice green squares and they should be put at our finest airports.

OK. How many do you want, Mr. President?

As few as possible to make them more exclusive. Can you do that for me? All the squares must have sides which are integers but they can have different sizes, of course.

Yes, Mr. President. Unfortunately, the advisor then thought loud “my sister is a hacker and I will ask her to write a program for it.”

What did you say? Never mind, please go and measure up the biggest grass rectangle out there right away.

The advisor then called his sister and explained his problem.

I will try to do it right away. You can tell me the integer lengths of the sides of the rectangle later. I will start hacking!

Example 1

Input:

\[2 \times 7\]

Output:

2 squares of size 1 x 1
3 squares of size 2 x 2

Example 2

Input:

\[5 \times 6\]

Output:

3 squares of size 2 x 2
2 squares of size 3 x 3

Example 3

Input:

\[107 \times 124\]

Output:

4 squares of size 31 x 31
2 squares of size 4 x 4
2 squares of size 8 x 8
1 square of size 20 x 20
1 square of size 28 x 28
1 square of size 48 x 48
1 square of size 76 x 76
Election

The election laws in Sweden were recently changed and are now as follows. First of all, every citizen of age 18 or older must vote, by going to a voting booth and writing down the name of one of the 39 registered political parties trying to enter Riksdagen. After the election, a computer program will see if any party got a majority, that is, strictly more than half of the votes. If a party got a majority, that party wins and will form the new government. However, if no party gets a majority, the election is repeated the next week, and so on. Since after a few weeks, everyone will be so bored, it is expected that before October, the Swedes will be the first people who agree in advance, using social media, what to vote for, and then everyone can go back to their normal lives again.

The problem now is to find which party, if any, got a majority. Since everybody is mostly worried about whether they will have to go and vote also the next week, the majority detection algorithm needs to be efficient.

The input consists of an array of size \( n \) with objects corresponding to the \( n \) votes. These objects can only be compared for equality (i.e. \( = \) and \( \neq \)) but not other relational operators (i.e. not \(<\), \(\leq\), \(>\), or \(\geq\)). That the objects are shown in the examples as string does thus not indicate they can be compared as strings. It is thus not possible to sort the elements and any solution based on that gives zero points.

Example 1

Input:

A A B A C B A

Output:

Party A is in majority

Example 2

Input:

A A B A C B D A

Output:

No party is in majority
Languages

A new software company, Expert Hackers, in Lund needs to hire programmers for several projects which use various programming languages. The short-listed candidates are all very good but the question now is to what extent the projects can get all competences they require. Each candidate has listed which languages it is skilled in and in how many projects it is willing to work in at the same time, and each project which primary language they use, and how many programmers they need. The input consists of one line with two integers: the number $M$ of candidates and the number $N$ of projects. Then follow $M$ lines, with one line per candidate. Each line contains the name of the candidate, the number of projects that candidate wants to work in, and a list programming languages the candidate is skilled in. Then follows $N$ lines with an identifier of project, the main implementation language and an integer indicating how many programmers the project needs. The output is the sum of the number of programmers employed by the projects — i.e. if one programmer is employed by two projects, it is counted twice.

Example

Input:

```
3 4
Kålle 2 C, C++, Java
Ada 2 Ada, C
Osborn 3 C, Java, Scala

S C 2
T Ada 2
U Java 2
V Scala 1
```

Output:

```
6
```

Easily achieved since all projects except T find all required programmers.
**Workshop**

To get to know each other and to share knowledge about programming languages, the manager of Expert Hackers has decided to make a workshop with all new employees at a hotel which offers up to \( C \) conference rooms of various sizes. A group of programmers will present to each other themselves and the language they are most competent in and what to watch out for when using that language. To achieve the purposes, the manager has two criterions for whether two programmers should not go to the same conference room, and they are:

- they already know each other (which is assumed if they previously worked at the same place or graduated from the same university programme and year), or
- they have listed the same language as their best.

There are \( P \) programmers and the manager finds the needed information from the applications, and now starts making groups. The input consists of \( P \) and \( C \) on one line, and then \( P \) lines with one line per employee with name, best language, previous job (or ‘first job’), and graduation. The problem is to create groups, with one group per conference room and the goal is to create as few groups as possible without violating the criteria above.

The output is either the smallest number of rooms required or *impossible* if \( C \) rooms are insufficient.

**Example**

*Input:*

```
4 3
Kålle C++, Sony, Chalmers/D 2016
Ada Ada, Volvo, Chalmers/D 2012
Osborn C, Volvo, Chalmers/D 2013
Nils Java, Sony, LTH/D 2017
```

*Output:*

```
2
```

Ada and Kålle in one group and Osborn and Nils in one group.
Exam Questions

**Pseudo code.** Pseudo code is usually preferable to source code expressed in a programming language. The purpose, obviously, is to describe key ideas as simply as possible. Do not write any code to parse input but instead say for instance, “put each word from the input in an array” (if you want to do that).

**Analysis of Algorithms**

1. Let \( f(n) = (n^4 + n^3) \log n \). True or false?
   
   (a) (2p) \( f(n) = O(2^n) \)
   
   
   (b) (2p) \( f(n) = O(n^2 \log n) \)
   
   
   (c) (2p) \( f(n) = O(n^2) \)
   
   
   (d) (2p) \( f(n) = O(n^4 \log n) \)
   

2. Let \( f(a, n) = a^n \), and assume \( a \) is real and \( n \) is a positive integer. Ignore the limited precision of floating point arithmetic.

   ```
   void f(int n)
   {
   int i;
   int j;
   for (i = 0; i < n; i += 1) {
   for (j = 1; j < n; j *= 2) {
   /* ... */
   }
   }
   }
   ```

   (a) (2p) The time complexity of \( f(a, n) \) is — select the smallest correct estimate
   
   [A] \( O(\log n) \) [B] \( O(n) \) [C] \( O(n \log n) \) [D] \( O(n^2) \)

3. **Graph search.** One of the problems is suitable to solve using a simple graph search.

   (a) (2p) Which one?
   

   (b) (6p) Describe your solution. Use pseudocode.

   (c) (2p) What is the running time of your solution?
4. **Divide-and-conquer.** One of the problems is suitable to solve using divide-and-conquer.

   (a) (2p) Which one?
   
   A Register  B Squares  C Election  D Languages  E Workshop

   (b) (6p) Describe your solution. Use pseudocode.

   (c) (2p) What is the running time of your solution?

5. **Dynamic programming.** One of the problems is suitable to solve using dynamic programming.

   (a) (2p) Which one?
   
   A Register  B Squares  C Election  D Languages  E Workshop

   (b) (6p) Describe your solution. Use pseudocode.

   (c) (2p) What is the running time of your solution?

6. **Network flow.** One of the problems can be solved by a reduction to network flow.

   (a) (2p) Which one?
   
   A Register  B Squares  C Election  D Languages  E Workshop

   (b) (4p) Explain how and illustrate with an example.

   (c) (2p) What is the smallest correct running time? (see (d) below)
   
   A $O(Cm)$  B $O(Cmn)$  C $O(m \log n)$  D $O((m+n)^2)$  E $O(4mn^2)$

   (d) (2p) Which algorithm gives this time complexity, and what do $C$, $m$ and $n$ mean (if anything) in the problem?

7. **Computational complexity.** One of the problems is NP-complete\(^1\).

   (a) (2p) Which one? Call it $P_1$.
   
   A Register  B Squares  C Election  D Languages  E Workshop

   (b) (2p) The easiest way to see this is to reduce from $P_2$ where $P_2$ is
   
   A $k$-coloring  B 3-SAT  C Hamiltonian Cycle  D Vertex Cover  E Independent Set

   (c) (1p) and prove: A $P_1 \leq_P P_2$  B $P_2 \leq_P P_1$

   (d) (5p) Describe the reduction on a separate piece of paper. Do this both in general and for a small but complete example. In particular, be ridiculously precise about what instance is given, and what instance is constructed by the reduction, the parameters of the instance you produce (for example number of vertices, edges, sets, colors) in terms of the parameters of the original instance, what the solution of the transformed instance means in terms of the original instance, etc. Start your answer with the words “Given an instance to problem ...,”

---

\(^1\)If $P = NP$ then all these problems are NP-complete. Thus, in order to avoid unproductive (but hypothetically correct) answers, this question assumes that $P \neq NP$. 

9