Exam 25 October 2013, 8:00–13:00, Sparta:A–B

EDAN55 Advanced Algorithms

The exam consists of 4 large questions; each consisting of a number of smaller subquestions.

1. The exam is "open book," so you can bring whatever material you want, including textbooks, a dictionary, and your own course notes.
2. You can bring an electronic calculator.
3. We try to minimise the dependencies among subquestions. In particular, you can solve them in any order. Also, you are free to *use* the result of subquestion x to answer subquestion y, even if you didn't answer x.
4. Scoring: Answering "I don't know" (and nothing else) scores $\frac{1}{4}$ of a subquestion's points. An empty or wrong answer scores 0 points.
5. You can answer in Swedish or English.

Some tips:

1. Shorter is better.
2. An example is better than a failed attempt at explaining something in general.
3. Drawings, pseudocode, and formulas are good. "Wall of text" is bad.
4. Admit ignorance.
5. Be tidy.

Good luck!

Question 1, Approximation

In the *edge colouring problem*, colours are assigned to vertices such that no vertex is incident on edges of the same colour. Formally:

Name: Edge colouring

Input: A simple, undirected graph $G = (V, E)$ with $|V| = n$, $E = \{e_1, e_2, \dots, e_m\}$.

Output: An integer q and a mapping $f\colon E \to \{1, 2, \dots, q\}$ such that $f(vw) \neq f(vu)$ for each pair of edges vw and vu around the same vertex, with q minimal.

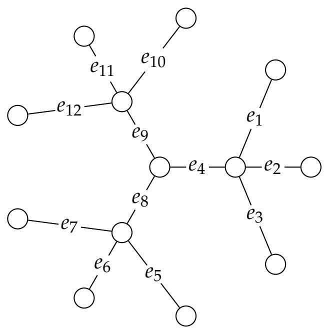

Figure 1: A graph.

►*1a* (1 pt.) Find an optimal edge colouring of the graph in fig. 1. [1]

[1] Your answer is a drawing showing the colouring. Make it clear what q is.

We consider the greedy algorithm for edge colouring:

```
Set $f$ to undefined for each $v$.
For each $i$ from 1 to $m$,
    Let $v$ and $w$ the endpoints of edge $e_i$
    Set $f(e_i)$ to be the smallest integer $\geq 1$ not yet used among the edges around $v$ and $w$.
Return $f$ and $q = \max_{e \in E} f(e)$.
```

►*1b* (1 pt.) Run the algorithm on the graph in fig. 1. [2]

►*1c* (1 pt.) Give an example of a 4-edge colourable graph where the greedy algorithm uses 6 colours. [3]

►*1d* (3 pts.) Show that the algorithm is guaranteed to find colouring using no more than twice the optimum. [4]

►*1e* (2 pts.) Prove that unless P equals NP, there cannot be a polynomial time approximate edge colouring algorithm with guarantee $(\frac{4}{3} - \epsilon)$OPT for any $\epsilon > 0$. [5]

[2] Your answer is the resulting colouring. Make clear what q is.

[3] Your answer is a drawing of a graph, an optimum solution to that instance and the solution found by the algorithm.

[4] Your answer is a short argument. It includes a lower bound on the solution found by the algorithm and an upper bound on the optimum solution.

[5] Your answer is a short proof. You can freely use that it is NP-hard to 3-edge colour a 3-edge colourable graph.

Question 2, Parameterized Analysis

We consider the well-known independent set problem restricted to planar graphs:

Name: Independent set

Input: A planar graph $G = (V, E)$, integer k

Output: A subset of vertices $S \subseteq V$ with $|S| \geq k$ such that for each edge vw, at most one of the endpoints belongs to S.

Recall that a simple branching algorithm solves independent set in time $O(1.3803^n)$ in the general case, but in this exercise we look at parameterized analysis, where the expression involves k as well as n.

►*2b* (2 pts.) Write a simple exhaustive search (or "brute force") algorithm and give its running time in terms of n and k. [6]

[6] Your answer is some lines of pseudocode and a running time estimate using asymptotic notation.

We will use that the average degree $2m/n$ of every planar graph is strictly less than 6. (This follows from Euler's formula. The proof takes a few lines but is not important here.)

►*2c* (3 pt.) Design an algorithm for independent set in planar graphs. The running time must be of the form $f(k)\,\mathrm{poly}(n)$. *Hint:* My solution is a branching algorithm in time $6^k\,\mathrm{poly}(n)$. There may be other ways of doing it. [7]

[7] Your answer is the description of an algorithm, for instance using pseudocode, and a brief analysis of its running time.

The dependency on k in 2c can be improved using the following famous result (the proof of which takes up several hundred pages):

Four Colour Theorem. *Every planar graph can be 4-coloured.*

►*2d* (2 pts.) Design an algorithm for planar independent set with running time better than $6^k\,\mathrm{poly}(n)$. *Hint:* As a first step, consider the case $k < \frac{1}{4}n$. My solution achieves running time $3.63^k\,\mathrm{poly}(n)$, but there's no reason to be so precise in the analysis.

Note that the FPT results is surprising in light of the fact that Indepenent Set in general (without the restriction to planar graphs) is hard for $W[1]$.

Question 3, Exponential Time Algorithms

We consider the 1-in-3-Sat problem

Name: 1-in-3-Sat

Input: A CNF formula in the variables $\{x_1, x_2, \ldots, x_n\}$ with at most 3 literals per clause.

Output: An assignment from $\{x_1, x_2, \ldots, x_n\}$ to $\{\text{true}, \text{false}\}$ such that there is exactly one true literal in each clause.

For instance, the formula

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_5) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\overline{x}_4 \vee \overline{x}_5)$$

is 1-in-3-satisfied by the assignment $x_1 = x_5 = \text{true}$, $x_2 = x_3 = x_4 = \text{false}$.

▶*3a* (1 pt.) Find another 1-in-3-Satisfying assignment for ϕ.

▶*3b* (2 pt.) Explain very briefly how 1-in-3-Sat can be solved using exhaustive search ("brute force") and state the resulting running time, ignoring polynomial factors.

▶*3c* (3 pts.) Construct a simple branching ("decrease-and-conquer") algorithm for 1-in-3-Sat. You running time must be better than 2^n. Be precise about which branching rules you use; for example by writing the algorithm in some form of pseudocode. Give a recurrence relation for the running time of the resulting algorithm and state the resulting running time. *Hint:* I get $O(3^{n/3}) = O(1.45^n)$.

Question 4, Randomized Algorithms

We consider the bichromatic list-colouring problem. List colouring is like standard vertex colouring, except that each vertex has its own list (or "palette") of allowed colours.

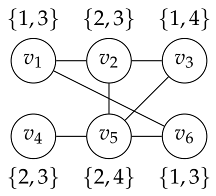

Figure 2: An instance to List colouring, with each list shown next to its vertex. For instance, $L(v_1) = \{1,3\}$.

Name: List-colouring

Input: A simple, undirected graph $G = (V,E)$ with $|V| = n$, $|E| = m$ and for each vertex v a set of colours $L(v)$.

Output: A mapping $f\colon V \to \{1,2,\ldots\}$ such that $f(v) \neq f(w)$ for each edge vw (so f is a vertex colouring) and $f(v) \in L(v)$ for each vertex v (so each colour is chosen from the palette of that vertex.)

►*4a* (1 pt.) Find a list-colouring of the instance in figure 2.

►*4b* (1 pt.) Show that the list colouring problem can be solved in polynomial time when each $L(v)$ has size at most 2. [8]

[8] Your answer is a very short description, either in pseudocode or by referring to some other well-known algorithm. This is supposed to be easy, and has little to do with exponential, parameterized, or randomized algorithms.

From now on you can consider list colouring with $|L(v)| = 2$ to be polynomial-time computable, no matter if you solved question 4a.

We turn to the well-known vertex 3-colouring problem.

Name: Vertex 3-colouring

Input: A simple, undirected 3-colourable graph $G = (V,E)$ with $|V| = n$, $|E| = m$.

Output: A mapping $f\colon V \to \{1,2,3\}$ such that $f(v) \neq f(w)$ for each $vw \in E$.

Consider the following randomized algorithm for this problem:

Algorithm R

1. For each v, choose $L(v)$ to be $\{1,2\}$, $\{2,3\}$, or $\{1,3\}$ independently and uniformly at random.
2. Attempt to solve the resulting list colouring instance in polynomial time.
3. If successful, return the resulting colouring. Otherwise go back to step 1.

►*4c* (1 pt.) What is the probability that $L(v_1) = \{1,2\}$? [9]

[9] Your answer is a value.

▶*4d* (1 pt.) What is the probability that 1 does not occur in any of the $L(v)$? [10]

▶*4e* (1 pt.) What is the probability that $L(v_1) = L(v_2) = \cdots = L(v_n)$? [11]

▶*4f* (1 pt.) Assume G is 3-colourable and let f be a 3-colouring of G. What is the probability that $\forall v \in V \colon f(v) \in L(v)$? [12]

▶*4g* (3 pt.) What is the expected running time of the algorithm R? [13]

[10] Your answer is an expression and a very short argument.

[11] Your answer is an expression and a very short argument.

[12] Your answer is an expression and a very short argument.

[13] Your answer is an expression and an argument for it.