Exam 9 January 2013, 8:00–13:00, Sparta:C

# 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*

Recall that an *independent set* in an undirected graph $G = (V, E)$ is a subset $W$ of vertices such what no edge in $E$ has both endpoints in $W$. The independent set problem is to find a maximum size independent set in a given graph. We assume that the graph has maximum degree $\Delta = 3$, i.e., every vertex has at most 3 neighbours.

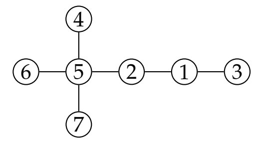



Figure 1: An instance to Independent set.

►*1a* (1 pt.) Find a maximum independent set in the graph in fig. 1.[1]

[1] Your answer is a drawing showing the indpendent set and an integer (the size of your solution).

Consider the following algorithm:

Initially, set $W = \emptyset$.
while $V$ is not empty,
  pick a $v \in V$ (say, the lowest numbered vertex, just to be precise)
  add $v$ to $W$
  remove $v$ and all its neighbours $\{x \colon xv \in E\}$ from $V$

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

►*1c* (1 pt.) Give an example where the algorithm finds an independent set of size only $\frac{1}{3}$OPT on a degree-3 graph. [3]

►*1d* (3 pts.) Show that the algorithm is guaranteed to find an independent set of size at least $\frac{1}{3}$OPT of the optimum for any degree-3 graph. [4]

►*1e* (2 pts.) Prove that unless P equals NP, there cannot be an algorithm for the Independet Set Problem whose solution is at most a factor $(1+\epsilon)$ below OPT and that runs in polynomial time for any choice of $\epsilon > 0$. You can freely use that Independent set is NP-hard.[5]

►*1f* (1 pt.) What is the approximation factor if we run the algorithm on graphs of maximum degree $\Delta = 4$? And $\Delta = 17$?

[2] Your answer is the resulting solution and the solution size.

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

[4] Your answer is a short proof.

[5] Your answer is a short proof. In particular, be precise about how you choose $\epsilon$.

## *Question 2, Parameterized Analysis*

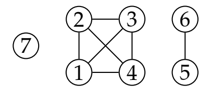


Figure 2: A clean 7-vertex graph (it consist of 3 disjoint cliques, of size 1, 2, and 4, respectively.)

In this exercise, we call a graph *clean* if it consists of a disjoint union of cliques. (Recall that a clique is a complete subgraph, i.e., a vertex subset $C \subseteq V$ such that $uv \in E$ for all $u, v \in C$ with $u \neq v$.)

►*2a* (1 pt.) Describe a simple algorithm to determine if an input graph is clean. [6]

[6] Your answer is some lines of pseudocode and a running time estimate using asymptotic notation. It's easily doable in polynomial time, but even an exponential time algorithm will suffice.

►*2b* (1 pt.) Which of the following 4 patterns on 3 vertices can not appear as an induced subgraph in a clean graph?

►*2c* (1 pt.) How fast can such a "forbidden" 3-vertex pattern be detected in a given graph? [7]

[7] Your answer is an asymptotic running time expression, polynomial in $n$.

A graph is $k$-dirty if it can be turned into a clean graph by adding or removing at most $k$ edges. (Thus, a 0-dirty graph is clean.) The *$k$-cleaning* problem is, given an undirected graph $G = (V, E)$ and an integer $k$, to add or remove $k$ edges such that the resulting graph is clean.

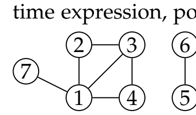


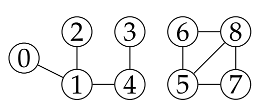
Figure 3: A 2-dirty graph. It can be cleaned by removing the edge between 1 and 7, and adding an edge between 2 and 4.

*Name:* $k$-cleaning.

*Input:* A graph $G = (V, E)$. Integer $k$.

*Output:* A set $R \subseteq E$ and a set $A \subseteq \overline{E}$ such that the graph $G' = (V, E \cup A - R)$ is clean and $|A| + |R| \leq k$, or "impossible" if no such sets exists.

►*2d* (1 pt.) Solve the $k$-cleaning problem for the following 9-vertex graph:



for $k = 3$. What is the answer for $k = 2$?

►*2e* (2 pts.) Write a simple exhaustive search (or "brute force") algorithm for $k$-cleaning and give its running time. [8]

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

►*2f* (1 pt.) Assume $G$ is $k$-dirty for some $k \geq 1$, so it contains at least one occurence of the "forbidden" subgraph(s) that we identified in 2b. Let $u$, $v$, $w$ denote the corresponding vertex names in $G$. Edit the following sentence in order to make it true "If we add or remove [any / a particular / a random / all / some] of the edges $uv$, $vw$, or $wv$, the resulting graph is [clean / $(k+1)$- / $k$- / $(k-1)$- / $\log k$- / $\frac{1}{2}k$ / $\sqrt{k}$-dirty]."

►*2g* (4 pts.) Write an algorithm based on the above observation, briefly argue for its correctness, and state its running time. We are after "FPT time", i.e., a running time of the form $f(k) \cdot n^{O(1)}$ for some function $f$.

## *Question 3, Exponential time algorithms*

We consider the problem of 2-colouring a 3-uniform family of sets.

*Name:* Bichromatic Balancing of 3-Sets (BB3S)

*Input:* A set $U$ of $n$ elements and a collection $C$ of $m$ subsets $S_1, \ldots, S_m \subseteq U$, all of size $|S_i| = 3$.

*Output:* A partition of $U$ into two not necessarily equal sized parts $R$ and $B$ such that every $S_i$ is not completely in $R$ nor completely in $B$. Formally, $\forall i\colon |S_i \cap R| < 3 \wedge |S_i \cap B| < 3$. If no such solution exists, the word "impossible."

Think of the elements of $U$ as coloured red ($R$) or blue ($B$).

For instance, figures 4 and 5 contain two instances with $n = m = 7$. (One of them is solvable and the other is not.) The task can be viewed as colouring the elements of $U$ so that in no row the three marked elements have the same color.

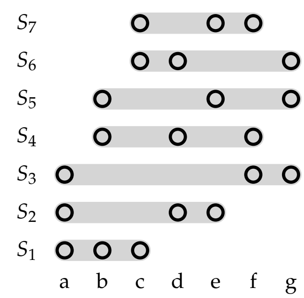



Figure 4: An instance to BB3S. $U = \{a, b, \cdots, g\}$, $S_1 = \{a, b, c\}$, etc.

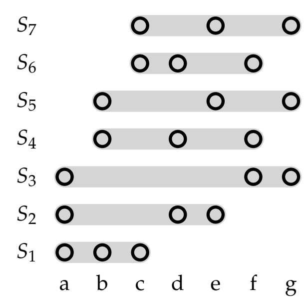



Figure 5: Another instance to BB3S

▶*3a* (1 pt.) Solve the instances in figures 4 and 5. One of them is impossible.[9]

▶*3b* (2 pt.) Explain very briefly how BB3S can be solved by exhaustive search ("brute force") and state the resulting running time (you may ignore polynomial factors in $n$ and $m$).

▶*3c* (2 pt.) Explain very briefly why BB3S can be solved in polynomial time *provided you knew for every $S_i$ the colour of at least one of its elements*. In other words, you are given a colouring of some of the elements of $U$ such that every $S_i$ already has at least on element coloured, and you want to find out if the partial solution can be extended to a full solution by also colouring the remaining elements.

▶*3d* (4 pt.) Construct an algorithm for BB3S with a good worse case run time bound.[10]

[9] Your answer is a list of the sets $R$ and $B$ and a very clear statement of which of the two instances you solve.

[10] Your answer is a description of the algorithm, possibly in pseudocode. Clearly state and prove the running time bound, which has to be better than $2^n$. We know at least 3 different ways to answer this question.

## *Question 4, Randomized Algorithms*

$$\phi = (\overline{x_1} \vee x_2 \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee x_4) \wedge (x_1 \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

Figure 6: A 3-CNF formula $\phi$

We consider the (optimisation version) of the Not-all-equal Satisfiability problem. It's like 3-Sat, except that we also forbid clauses with all three literals *true*. Formally, a clause is *NAE-satisfied* if it contains at least one true literal and and least one false literal.

*Name:* Max NAE-Sat

*Input:* A CNF formula in $n$ variables with $m$ clauses and exactly 3 literals per clause. Let's agree that no variable appears twice in any clause.

*Output:* An assignment that NAE-satisfies as many clauses as possible.

►*4a* Find a maximum NAE-satisfying assignment for $\phi$ in figure 6.

Consider the following randomized algorithm for this problem:

1. For every variable $x_i$ ($i = 1, \ldots, n$), pick its truth value uniformly and independently at random.

►*4b* (1 pt.) Run the algorithm on $\phi$ in figure 6. Use the random values $t, f, f, t$. How large is the resulting solution?[11]

►*4c* (1 pt.) Consider the clause $(x_1 \vee x_6 \vee \overline{x_{16}})$. What is the probability that this clause is NAE-satisfied?

►*4d* (2 pt.) Consider the clauses $C_1 = (x_1 \vee x_2 \vee x_3)$ and $C_2 = (x_1 \vee x_4 \vee x_5)$. Let $E_i$ denote the event that $C_i$ is NAE-satisfied. Compute $\Pr(E_1 \cup E_2)$, $\Pr(E_1 \cap E_2)$, and $\Pr(E_1 \mid E_2)$. Are $E_1$ and $E_2$ independent?

►*4e* (2 pt.) Compute the expected solution size, i.e., the number of NAE-satisfied clauses.[12]

►*4f* (1 pt.) Determine the approximation ratio of the algorithm.[13]

[11] Your answer is an integer.

[12] Include the calculation, be explicit about assumptions such as independence, linearity, etc.

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