Institut for Matematik og Datalogi
Syddansk Universitet, Odense — April 12, 2024
JBJ

DM553/MM850 – 3. Exam assignment

Hand in by May 7, 2024 at 10:00.

Rules

This is the third of three sets of problems which together with the oral exam in June constitute the exam in DM553. This set of problems may be solved in groups of up to three. Any collaboration between different groups will be considered as exam fraud. Thus you are not allowed to show your solutions to fellow students, not from your group and you may not discuss the solutions with other groups. On the other hand, you can learn a lot from discussing the problems with each other so you may do this to some extend, such as which methods can be used or similar problems from the book or exercise classes.

It is important that you **argue so that the reader can follow your explanations**. It is not enough just to say that the solution follows from an example in the book or similar. In such a case you should repeat the argument in your own words.

Remember that this counts as part of your exam, so do a good job and try to answer all questions carefully.

How to hand in your report

Your report **should be written in english or danish** and must be handed in via It-slearning by Tuesday May 7 at 10:00.

Hand in one report per group. On the first page you must write the **names** of all participants in the group as well as the first 6 digits of your **CPR-numbers**. Do not write the last 4 digits!

Exam problems

Solve the following problems. **Remember to justify all answers. If you do not justify an answer, you may not get any credit for your answer!**

PROBLEM 1 (30%)

We say that a set of vertices X in a graph $G = (V, E)$ is a **dominating set** if every vertex not in X has at least one edge to a vertex in X. In this problem we study the following problem which we call the **graph domination problem**:

Input: a graph $G = (V, E)$ and a natural number r
Question: Does G have a dominating set with at most r vertices?

Question a:

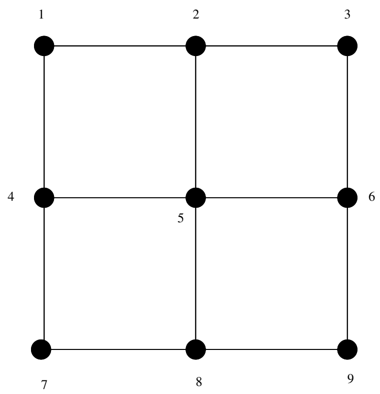

Figure 1: A graph G on 9 vertices.

Let G be the graph in Figure 1. What is the minimum size of a dominating set in G? You must argue for your claim.

Question b:

Describe a greedy algorithm for finding a dominating set of a given graph. You should explain why your algorithm is greedy. Then show how your algorithm would work on the graph G in Figure 1. Does your algorithm find the optimum (smallest possible) dominating set in G?

Question c:

Argue that the graph domination problem is in NP.

Let $I = ((S, \mathcal{F}), k)$ be an instance of the set-covering problem with $\mathcal{F} = \{S_1, S_2, \ldots, S_{|\mathcal{F}|}\}$, where $S_i \subseteq S$ for $i = 1, 2, \ldots, |\mathcal{F}|$. Form a bipartite graph $G = G(I)$ whose vertex set is $V = S \cup U$ where $U = u_1, \ldots, u_{|\mathcal{F}|}$ has one vertex for each set in $\mathcal{F}$ and let G have the following edges:

- An edge from u_i to each $s \in S_i$ for $i = 1, 2, \ldots, |\mathcal{F}|$.
- All possible edges between the vertices in U (so they form a clique).

Question d:

Argue that $G(I)$ can be constructed in polynomial time from I.

Question e:

Argue that $G(I)$ has a dominating of size k if and only if $(S, \mathcal{F})$ has a set-cover of size k (that is. I is a 'yes'-instance of set-cover).

Question f:

Argue that the graph dominating problem is NP-complete.

Question g:

It is a known (and difficult) result that, unless P=NP, there exist a constant c so that there is no polynomial $c \log n$-approximation algorithm for set-cover. Another way of saying this is that, asymptotically, the greedy set-cover algorithm from Cormen has the assymptotically best possible approximation guarantee we can hope to have for the set-cover problem.
Explain why this means that there exists a positive real number d such that there is no $d \log n$-approximation algorithm for the graph domination problem. Hint: show how to use a given $\rho(n)$-approximation algorithm for graph domination to obtain a $\rho(n)$-approximation algorithm for set-cover.

PROBLEM 2 (40%)

For each of the following claims you should say whether they are true or false AND give a short (but sufficient!) explanation for your conclusion. **It is not enough to write that the claim is true or that it is false!**

1. If L and L' are non-trivial members of P, that is, there exist both a 'yes'-instance and a 'no'-instance for each (so $\emptyset \neq L, L' \neq \Sigma^*$), then $L \leq_p L'$.
2. Let L, L' be optimization problems whose decision versions are NP-complete, then any k-approximation algorithm for L can be turned into an $f(k)$-approximation algorithm for L' for some finite function f.
3. There exists a polynomial algorithm $\mathcal{C}$ which finds a vertex cover of size at most $\frac{11}{6}$ times the optimum size of a vertex cover in a given graph $G = (V, E)$ with $\max\{d(v)|v \in V\} \leq 3$, where $d(v)$ is the number of edges incident with v in G.
4. There is a $\frac{137}{60}$-approximation algorithm for the special case of the set covering problem in which all subsets have size at most 5.
5. Every comparison based algorithm for finding the two smallest elements in a set of n integers must use at least $2n - 3$ comparisons in the worst case.
6. Every comparison based algorithm for finding the median of a set of integers must use at least 150 comparisons to determine the median of some set of 101 integers.
7. The hamiltonian cycle problem can be solved in polynomial time for graphs of maximum degree 2 (that is, $d(v) \leq 2$ for all vertices $v \in V(G)$).
8. If a graph G on n vertices has an independent set of size k, then G has a vertex cover of size at most $n - 2k$.
9. Not every problem in NP can be solved in exponential time.
10. The halting problem is NP-complete.
11. Recall that a digraph $D = (V, A)$ is strongly connected if it has a directed path from x to y for every choice of vertices $x, y \in V$.
    Let $STRONG = \{< D > |D \text{ is a digraph which is strongly connected}\}$.
    Claim: $STRONG \in P$.
12. Every strongly connected digraph $D = (V, A)$ contains a spanning strong subdigraph $D' = (V, A')$ with $A' \subseteq A$ on at most $2|V| - 2$ arcs.
13. For every fixed k it is NP-complete to decide whether a given graph $G = (V, E)$ has a collection of k disjoint cycles $C_1, C_2, \ldots, C_k$ which together cover all its vertices, that is, $V(C_1) \cup V(C_2) \cup \cdots \cup V(C_k) = V$.
14. The hamiltonian cycle problem is polynomially solvable for graphs with at most $10^{10^{10}}$ vertices. That is, there exists a polynomial algorithm for deciding whether such a graph has a hamiltonian cycle.

15. The decision version of the following problem $\mathcal{Q}$ is NP-complete: given a connected graph $G = (V, E)$ with non-negative weights on its edges; find a minimum weight set of edges $E' \subseteq E$ such that the subgraph $G' = (V, E')$ induced by the edges in E' is connected.

16. The decision version of problem $\mathcal{Q}$ above is polynomial but if we insist that G' must be also 2-edge-connected, then it becomes NP-complete (recall that a graph G is 2-edge-connected if it remains connected after deletion of any edge).

17. There exists no comparison-based sorting algorithm which can sort every set of n distinct integers while using at most $0.49n \log n$ comparisons.

18. Let LONG CYCLE be the problem: Given as input a graph $G = (V, E)$ and a natural number k; does G have a cycle of length at least k? Claim: The problem LONG CYCLE is NP-complete.

19. Does the complexity of LONG CYCLE change if the cycle must have length exactly k?

20. If there exists a polynomial time $(1 + \frac{1}{n})$-approximation algorithm for vertex cover in graphs on n vertices, then P=NP.

PROBLEM 3 (15%)

In this problem we are given a set of $n = 2k$ distinct integers $S = \{x_1, x_2, \ldots, x_n\}$ and we want to find, the following three elements: the maximum, the minimum and the 2nd smallest element.

Let $\mathcal{A}_{min,max}$ be an optimal algorithm (so it uses the minimum number of comparisons) for finding the minimum and the maximum element of a set of distinct integers. Let $\mathcal{A}_{min,2min}$ be an optimal algorithm for finding the minimum and the 2nd smallest element of a set of distinct integers.

Consider the following algorithm $\mathcal{A}$

1. Let $L, H = \emptyset$
2. For $i = 1$ to k:
   if $x_{2i-1} < x_{2i}$ then add x_{2i-1} to L and x_{2i} to H;
   else add x_{2i} to L and x_{2i-1} to H
3. Use $\mathcal{A}_{min,max}$ to find the maximum and the minimum element y_{max}, y_{min} in H;
4. Use $\mathcal{A}_{min,2min}$ to find the minimum and the second smallest element z_{min}, z_{2min} in L;
5. return y_{max}, z_{min} and the smallest of y_{min}, z_{2min} as the maximum, the minimum and the 2nd smallest element of S.

Question a:

Prove that $\mathcal{A}$ is correct and determine how many comparisons $\mathcal{A}$ makes.

Question b:

Show how to use $\mathcal{A}_{min,2min}$ as a subroutine in an algorithm $\mathcal{B}$ which uses fewer comparisons than $\mathcal{A}$ above. Hint: which elements can be the maximum element?

PROBLEM 4 (15%)

This problem is also about lower bounds for comparison based algorithms.

Question a:

Let DUPLICATE be the problem: given a sorted list of integers $x_1 \leq \ldots \leq x_n$; are there any duplicates, that is, is there an index i such that $x_i = x_{i+1}$. First describe an algorithm that uses exactly $n-1$ comparisons to solve the problem and then show that every comparison based algorithm for DUPLICATE must use at least $n-1$ comparisons on some input of size n.

Question b:

A MERGE of two sorted lists $x_1 \leq \ldots \leq x_n$ and $y_1 \leq \ldots \leq y_n$ is the operation that returns one sorted list on the $2n$ elements. Prove that every comparison based algorithm $\mathcal{A}$ for performing a MERGE of two sorted lists of length n must use at least $2n-1$ comparisons in the worst case.
Hint: explain how an adversary can construct two lists $x_1 \leq \ldots \leq x_n$ and $y_1 \leq \ldots \leq y_n$ while answering all queries correctly so that it can force the algorithm $\mathcal{A}$ to use $2n-1$ comparisons before it knows the correct sorted order. Hint: think of an input to the merging part of Mergesort that will require $2n-1$ comparisons.