# Algorithms Exam TIN093/DIT602

**Course:** Algorithms

**Course code:** TIN 093 (CTH), DIT 602 (GU)

**Date, time:** 18 March 2020, morning 8:30-12:30

**Building:** Samhällsbyggnad, SB Multisal - replaced by home-exam

**Responsible teacher:** Birgit Grohe

**Examiner:** Birgit Grohe

**Exam aids:** one A4 paper (both sides) with own notes, dictionary, printouts of the Lecture Notes, lecture slides and assignments (possibly with own annotations). - exam aids replaced by central instructions by the universities due to change of examination form into homeexam (because of Corona virus), see question 0.

**Time for questions:** During the entire exam time by EMAIL birgit.grohe@cse.gu.se, please read all questions before emailing and please try to restrict the number of emails to a minimum since there are over 100 students that signed up for the exam.

**Solutions:** will appear on the course homepage.

**Results:** will appear in ladok.

**Point limits:** CTH: 28 for 3, 38 for 4, 48 for 5; GU: 28 for G, 48 for VG; PhD students: 38. Maximum: 60.

**Inspection of grading (exam review):** Will be announced on the course homepage.

**Instructions and Advice:**

- First read through all problems, such that you know what is unclear to you and what to ask the responsible teacher.
- Write solutions in English.
- Start every new problem on a new sheet of paper.
- Write your exam number on every sheet.
- Write legibly. Unreadable solutions will not get points.
- Answer precisely and to the point, without digressions.
  Unnecessary additional writing may obscure the actual solutions.
- Motivate all claims and answers.
- Strictly avoid code for describing a complex algorithm.
  Instead *explain* how the algorithm works.
- If you cannot manage a problem completely, still provide your approach or partial solution to earn some points.
- Facts that are known from the course material can be used.
  You don't have to repeat their proofs.

**Remark:** The number of points is not always "proportional" to the length or difficulty of a solution, but it may also be influenced by the importance of the topics and skills.

**Good luck!**

# 0 Administration [0 points]

The universities have changed the exam aids for this exam due to the exceptional situation (Corona virus):

*The exam is to be carried out individually, i.e., collaboration is not allowed. Due to the current circumstances, all examination aids are allowed regardless of what is written on the exam. Control for plagiarism will be carried out. For information, the exam cannot be written anonymously."*

That means you may use everything you find both in the course material, including the course book, or e.g. other books or sources. Of course, we discourage searching for solutions, the course is about how to solve problems yourself, not how to search for solutions that someone else created. (Searching for solutions is time-consuming, four hours are enough to solve the problems, but not enough to first search for solutions, find nothing, and then try to solve the problems in the remaining time.)
However, if you look into e.g. a book and happen to find parts of, or entire, solutions, you **must** include the source, otherwise you pretend you came up with the solution yourself and that would be plagiarism and will be reported.

This problem number 0 does not give any points, but you must comply with the statement below in order to get a possibility to pass the exam. No statement and signature means that we will not grade your exam.

Write this exact statement below and sign it: "I have read the text above and understand that I must cite any sources other than the original exam aids[1] or the course book Kleinberg-Tardos. I did not collaborate with anyone and wrote all solutions by myself."

Statement and signature:

[1] One A4 paper (both sides) with own notes and a dictionary. Furthermore printouts of the Lecture Notes, lecture slides and assignments (all possibly with own annotations)

# 1 True or False? [15 points]

1.1 Given a connected, undirected graph $G = (V, E)$. Is it possible to find some node $v \in V$ such that removing node $v$ and all its incident edges will not result in an disconnected graph (i.e. the resulting graph will still be connected)? If you think this is true give an algorithm (and possibly an example graph to show how the algorithm works). If you think it is not true, give a short proof/argument. [4 points]

1.2 Given a connected, undirected graph $G = (V, E)$ with non-negative edge costs $c_{ij}$, $(ij) \in E$. If the edge costs are non-unique, then there can be different MSTs with the same value. If the edge costs are unique, then the MST is unique, that is, only one MST with the minimum value exists (these two facts are true, you don't have to prove them). Here is the True-or-False question: If the edge costs are non-unique, then there exist always at least two MSTs with the same value. If you think it is true, give a short proof/argument. If you think it is not true, give a counter example [4 points]

1.3 Given two functions $f(n) = n^k$ and $g(n) = n^{k+1}$, $k \geq 0$ integer. Then $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$. Motivate your answer. [3 points]

1.4 It is possible to sort an array that only consists of 0's and 1's in linear time. If you think it is true, give an algorithm and a short explanation why it is linear. If you think it is not true, give a proof/argument why it cannot be true. [4 points]

## 2 Longest Common Sub-sequence [10 points]

Consider a sequence of letters or numbers. A *sub-sequence* of a given sequence is just the given sequence with 0 or more elements left out. Formally, given two sequences $X = \langle x_1, \ldots, x_m \rangle$ and $Z = \langle z_1, \ldots, z_k \rangle$ (assume $m \geq k$), then $Z$ is a sub-sequence of $X$ if there exists a strictly increasing sequence $\langle 1, \ldots, k \rangle$ of indices of $X$ such that for all $j = i_1 \ldots i_k$ we have $x_{i_j} = z_j$.

For example, consider sequence $X = \langle A, B, C, B, D, A, B \rangle$. Then $Z = \langle B, C, D, B \rangle$ is a sub-sequence of $X$ with indices $\langle 2, 3, 5, 7 \rangle$.

Given two sequences $X$ and $Y$, we say $Z$ is a *common sub-sequence* if $Z$ is a sub-sequence of both $X$ and $Y$.

Now consider that we want to find a *longest common sub-sequence (LCS)* of two sequences $X$ and $Y$, i.e., the common sub-sequence with the maximum length.

2.1 A brute force solution would be to enumerate all sub-sequences of $X$ and compare them to all sub-sequences of $Y$. How many sub-sequences would one have to enumerate and how many comparisons would that result in, in terms of $O()$? [2 points]

2.2 Give a dynamic programming algorithm to solve find the length of the longest common sub-sequence, and analyse its running time in terms of $O()$. You do not need to give a full pseudo-code, a suitable function $OPT()$ and an explanation in words is enough. Do not forget the initial condition(s). [8 points]

# 3 Recurrence Relations [10 points]

Suppose that to solve a given problem, we are given three different divide and conquer algorithms. Let the size of the input be $n$.

- Algorithm $A$ finds a solution by dividing it into five sub-problems, each of half the size of its input problem, recursively solves each of these five sub-problems, and then combines their solution to form the solution of the input problem in linear time.
- Algorithm $B$ finds a solution by recursively solving two sub-problems of size $n-1$ each, and then combines their solutions to form the solution of the input problem in a constant time.
- Algorithm $C$ finds a solution by recursively solving 9 sub-problems of size $n/3$, and then combines their solutions to form the solution of the input problem in quadratic time.

3.1 For each of the algorithms $A, B$ and $C$, write down the recurrence relation. Assume the base case $T(1)$ to be constant. [3 points]

3.2 Solve the recurrence relations and give the running time in terms of $O()$ for each of the algorithms (you may use the master theorem if applicable). [6 points]

3.3 Which of the algorithms would you choose if asked to implement an as efficient algorithm as possible? Give a short explanation of your choice. [1 point]

# 4 Paths for Trucks in a Road Network [8 points]

Assume you work for a company that operates trucks which carry wide load. Your task is to find paths from a starting location to a target location, but instead of finding the shortest or cheapest path you are only interested in paths that contain wide streets.

More formally: Given a directed graph $G = (V, E)$ with edge weights $w_{ij} \geq 0$, $(i, j) \in E$ (where $w_{ij}$ denotes the width of corresponding street in the problem described above). Given further a start node $s$ and a finish node $t$. Then the *Truck-with-wide-load Problem* is about finding a path from $s$ to $t$ where the minimum $w_{ij}$ is as large as possible. Let the minimum value among all $w_{ij}$ of a path $P$ be denoted by $w_{min}(P)$

Example: Given two $s - t$ paths $P_1$ and $P_2$, and let the edge weights of $P_1$ be $(5, 7, 3, 9, 11)$ and those of $P_2$ be $(6, 4, 4, 8)$. Then $w_{min}(P_1) = 3$ and $w_{min}(P_2) = 4$. The path we were looking for is thus $P_2$.

Design an efficient algorithm that solves this problem and analyse its running time in terms of $O()$. Describe your algorithm both in words and in pseudo-code. (Hint: maybe this problem reminds you of another prominent problem within the field of algorithms.)

# 5 A Healthcare Problem [10 points]

Given a set $A = \{a_1, ..., a_n\}$ and an accompanying collection of subsets of $A$, $\mathbf{B} = \{B_1, ..., B_m\}$ (meaning that $B_i \subseteq A$ for each $i = 1, ..., m$). You can imagine the set $A$ to be different drugs and the collection of subsets are groups of drugs that work well for a certain disease (one subset represents one disease). We would like to find a subset of drugs so that we are prepared for any of the diseases.
We define a **Efficient Healthcare set** $H$ for the collection $\mathbf{B}$ to be a subset of $A$ such that $H \cap B_i \neq \emptyset$ for each $i = 1, ..., m$. What this entails is that for each $i = 1, ..., m$ in the collection $\mathbf{B}$, $H$ must contain at least one element from $B_i$.
The Efficient Healthcare Problem ($EHP$) asks the following question: given a set $A$ and a collection of subsets $\mathbf{B}$, and given a number $k$, is there an Efficient Healthcare set $H \subseteq A$ for $\mathbf{B}$ such that $|H| \leq k$?

5.1 Show that $EHP$ is in NP. [2 points]

5.2 For some NP-complete problem $Y$, show that $Y \leq_p EHP$, i.e.: give a polynomial-time reduction from the problem $Y$ of your choosing to $EHP$. Do so given the following steps:

5.2.1 Provide a translation from $Y$ to $EHP$: create an instance of $EHP$ given the input to $Y$. Show that the translation can be done in polynomial time.[4 points]

5.2.2 Show that if the translated instance of $EHP$ has a solution, then $Y$ has a solution. That is to say:
*"If $EHP$ gives an answer **yes** for the translated instance, then $Y$ also gives an answer **yes**."* [2 points]

5.2.3 Show that if there is a solution for $Y$, then the translated instance of $EHP$ has a solution. That is to say:
*"If $Y$ gives an answer **yes**, then $EHP$ also gives an answer **yes** for the translated instance."* [2 points]

As suggestions, you could take one of the following NP-complete problems:

**3-SAT** Given a set of (disjunctive) clauses $C_1, ..., C_k$ over a set of Boolean variables $X = \{x_1, ..., x_n\}$, being each clause $C_i$ of length 3. Is there a satisfying truth assignment for $X$ satisfying all clauses $C_1, ..., C_k$?

**Vertex Cover** Given a graph $G = (V, E)$, a set $S \subseteq V$ is called a vertex cover if every edge $e \in E$ has at least one end in $S$ (i.e.: if $e = (u, v)$ then at least one of $u, v$ is in $S$). Given a number $k$, does $G$ contain a vertex cover of size at most $k$?

**Set Cover** Given a set $U = \{u_1, ..., u_n\}$ of $n$ elements, a collection $S_1, ..., S_m$ of subsets of $U$ ($S_j \subseteq U$ for all $j = 1, ..., m$), and a number $k$. Is there a collection of at most $k$ of these sets whose union is equal to all of $U$?

# 6 Running time analysis [7 points]

Let $k$ be a positive integer. The following procedure computes $2^k$:

```
power(k) {
    pow := 1;
    for i := 1 to k do
      pow := pow * 2
    return pow
}
```

4.1 Assume that multiplication can be done in constant time. What is the running time of this algorithm in terms of the size of the input? (That is, first think of what is the size of the input?) Explain. [2 points].

4.2 Assume that multiplication cannot be done in constant time, but "costs" depending on the size of the numbers to be multiplied. Assume "$x * y$" costs $O(\text{size}(x) * \text{size}(y))$. What is the running time of this algorithm? Explain. [3 points]

4.3 Is this a polynomial algorithm or not? Why? [2 points]

## Short solutions (attached after the exam)

1.1 True. Run BFS, then any leaf node of the BFS tree can be removed.

1.2 False. Counterexample e.g. Four nodes $A, B, C, D$ with edge costs $AB = 1$, $BC = 2$, $CD = 3$ and $DA = DB = 4$. The MST is unique.

1.3 f$(n) \in O(g(n))$ but not the other way around (e.g. for $k = 1$).

1.4 True, e.g. with a Bucket-sort-style algorithm:

Pass through the array of size $n$ and count how many 0's and 1's there are (can be done in linear time), suppose there were $k$ 0's and $p$ 1's ($n = k+p$). Then write 0's in the first $k$ entries of the array and 1's in the remaining (also linear time).

2.1 For $|X| = m$ and $|X| = n$, there exist $2^m$ and $2^n$ sub-sequences, respectively. Number of comparisons required is thus $O(2^{n+m})$.

2.2 Let $X_i = \langle x_1, \ldots, x_i \rangle$, i.e., the first $i$ elements of $X$. Define $opt[i, j]$ to be the length of an LCS of sequences $X_i$ and $Y_j$.

Initial conditions: $opt[i, j] = 0$ if $i = 0$ or $j = 0$.

For the recurrence, there are two cases, either the last elements are the same (then they both will be in the common sub-sequence) or they are not (then only one of them will be):

$$opt[i,j] = \begin{cases} opt[i-1, j-1] + 1 & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max\{opt[i, j-1], opt[i-1, j]\} & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

The algorithm first initialises and then fills the 2-dim array $opt(i, j)$. Assume wlog $n \geq m$, then the running time is $O(n^2)$.

3.1 A: $T_A(n) = 5T(n/2) + cn$

B: $T_B(n) = 2T(n-1) + c$

C: $T_C(n) = 9T(n/3) + cn^2$

3.2 A: $O(n^2, 322)$ (master theorem)

B: Enrolling $T(n) = 2T(n-1) + c = 2(2T(n-2) + c) + c = 2(2(2T(n-3) + c) + c) + c = \ldots = 2^n T(1) + c\sum_{i=0}^{n-1} 2^i$

Since the first term grows fastest: $O(2^n)$

C: $O(n^2 \log n)$ (master theorem)

3.3 Since log grows slower than any polynomial (see course book KT chapter 2 statement (2.8)), algorithm C is fastest and should be chosen.

4 A more common name of this problem in the literature is shortest paths with bottlenecks. Efficient solutions can use MST or Shortest path algorithms (e.g. Dijkstras) with small adaptions that do not change $O()$.

E.g., use Kruskals algorithm, but instead sort the edges by largest first. Add edges as long as no cycles are created. The resulting tree will have an s-t-path, and this path will be a path with the maximum $w_{min}$.

5 The problem is the same as a well-known standard problem from the literature: The Hitting set problem (HS)

5.1 Check $|H| \leq k$, takes linear time. Then, for each $i = 1, ..., m$: check some element in $H$ is in $B_i$, takes at most $\mathcal{O}(mn)$.

5.2 Reductions from any of the suggested NP-complete problems are possible and can be found in the standard literature. A more detailed suggestion on how to do the translation from Set Cover will follow.

6.1 Input size is $n = \log k$. The running time is $O(k)$ or $O(2^n)$.

6.2 Given $size(x) = \log_2 x$ and thus $size(2) = 1$, the running time is $\sum_{i=0}^{k-1} size(2^i) \leq k \cdot size(2^{k-1}) = k(k-1)$ and thus $O(k^2)$.

6.3 The algorithm not polynomial in the input size. The algorithm is pseudo-polynomial.