Algorithms Re-exam TIN093/DIT602

Course: Algorithms

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

Date, time: 26 August 2021, afternoon

Location: online

Responsible teacher/examiner: Birgit Grohe

Exam aids: This is an open book exam, all aids allowed except communication with other humans. **VERY IMPORTANT: You must provide references for everything that you use from a source other than your own thoughts!** You may of course use facts from the course book or other course material, in those cases it is ok to give a very short reference e.g. "course material", "course book", or similar.

Time and location for questions: As announced on the Canvas page for this re-exam.

Solutions: will appear on the Canvas 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 Canvas page.

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.
- Write solutions to each problem in a separate document. There will be separate uploads in Canvas for each problem (all uploads must be in pdf format).
- If possible, type your solutions on a computer. Otherwise: write legibly, preferably use a scanner app such as CamScanner, Genius Scan or similar. In any case make sure if you scan/photograph that the result is not too dark and is fully readable. Unreadable solutions will not get points.
- Write your name on each document.
- 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 and use pseudo-code if asked to.
- 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!

1 True or False? [14 points]

1.1 Given an undirected graph $G = (V, E)$ with costs c_{ij} on the edges in E. Assume further that the graph is connected and $c_{ij} = k > 0$ for all edges. Then it is possible to compute the number of shortest paths between two nodes v_1 and v_2 in $O(|E|)$ (note: only the nr. of shortest paths required, not the paths themselves). If you think it is true, suggest a method (description in words or pseudo-code) and explain why the method runs in $O(|E|)$. If you think it is not true, give an argument why it is not possible to do it in $O(|E|)$. [4 points]

1.2 Given a directed graph $G = (V, E)$ with edge costs $c_{ij} \geq 0$. Let e^{max} be the edge with the maximum cost, i.e. $c_{e^{max}} \geq c_e$ forall $e \neq e^{max}$. Let s and t be two of the nodes in the graph. Assume that neither s nor t are start or end nodes of edge e^{max}.

Assume we want to find a shortest path from s to t. Then e^{max} cannot be part of a shortest $s - t$ path. If you think the statement is true, give a proof. If you think the statement is false, provide a counter example. [4 points]

1.3 Given two functions $f(n) = k^n$ and $g(n) = (k + 1)^n$ for $k > 0$ integer. Then $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$. If you think the statements are true, give a proof. Otherwise, argue why it cannot be true. [4 points]

1.4 Assume we run BFS and DFS on a connected graph $G = (V, E)$. Then BFS and DFS will always output different trees. If you think the statement is true, give a proof. Otherwise provide a counter example. [2 points]

2 Searching in an array [8 points]

Assume an array of distinct integers $[a_1 \dots a_n]$, $n \geq 2$. An *array top* is an element a_t of the array where $a_{t-1} < a_t > a_{t+1}$, $2 < t < n-1$, or for $t = 1$ and $t = n$, if the element is strictly large than its neighbour. (Correction given during the exam: it should be $2 \leq 1t \leq n-1$ instead.)
Your task is to design a sub-linear algorithm to find an array top if one exists.

2.1 Design an algorithm and give a description in English. [2 points]

2.2 Give pseudo-code for your algorithm. [3 points]

2.3 What is the running time of your algorithm in terms of $O()$? Explain. [3 points]

3 Maximising points in an exam [15 points]

You are working on a take-home exam that has n tasks. For each task you can get a maximum of p points. There are M hours until the deadline. You probably don´t have time to do all tasks perfectly, so you may have to choose your level of ambition for each task in order to maximise the total number of points.

Assume you are aware if the correlation between your investment of time and the outcome in points for each of the tasks. You know if you e.g. spend 1 hour on task 2 you will get $x_{2,1} \leq p$ points etc. Also assume $x_{i,j} \leq x_{i,j+1} \leq p$ (the more time you spend the more points you get, but not more than p). Investing 0 hours gives 0 points.
The points scale is discrete, there are no half-points. You can only choose to work on a task for whole hours, no fractions of hours are allowed. M and p are positive integers.

3.1 Make up a small example with 3-4 tasks, $p = 30$ and $H = 8$. Write down the details of your example including an optimal solution. [2 points]

3.2 For the general case described above, design an efficient algorithm using Dynamic programming (DP).

3.2.1 Define the $OPT()$ function, i.e. describe in words what $OPT()$ denotes in terms of the index/indices you choose. [2 points]

3.2.2 Write down a DP recurrence for your $OPT()$ function. [2 points]

3.2.3 Write down initial condition(s). [1 points]

3.2.4 Describe the entire algorithm in words or pseudo-code. If you choose to describe in words, give sufficient details so that we can understand your answer in 3.2.5. [4 points]

3.2.5 What is the running time of your algorithm in terms of $O()$? [2 points]

3.3 Can you think of a special case of this problem that can be solved with a very simple algorithm (that runs in linear time or in $O(n \log n)$? [2 points]

4 A hiking problem [11 points]

You aim for a pandemic-friendly vacation and want to go hiking with a friend in the middle of nowhere. You have a list of n items that you need to bring to survive in the woods. Each item has weight $w_i \geq 0$. Your friend insists you split the total weight exactly equally, otherwise (s)he will refuse to join the hike.
You and your friend try a few algorithmic ideas, but nothing seems to work. Maybe this hiking problem is NP-complete?

Let us be more precise and describe the following decision problem HIKING: Given n items and two backpacks of equal capacity C, where C is half of the total weight of all items. Is there a way to fill (but not overfill) the backpacks and not leave any items behind?

4.1 Make up a small example with 4-6 items including w_i and write down how the two backpacks can be filled (if it is possible to fill them evenly). [1 point]

4.2 Show that HIKING is in NP. [2 points]

4.3 Pick a known NP-complete problem X and suggest a polynomial time reduction. You can choose any NP-complete problem, but if you want a hint try the Subset-sum problem (see course material). [4 points]

4.4 To complete the NP-completeness proof, show that X has a Yes-answer if and only if HIKING has a Yes-answer. [4 points]

5 Vaccine delivery scheduling [12 points]

Your friend Charlie works at a startup "Corona vaccine distribution" and is in charge of scheduling delivery of vaccine doses to the different health care centres (HCC´s) in the north of Sweden. The company is small, they have only one truck.
The region consists of $n > 100$ villages. There is at most one HCC in each village. Not every village has a HCC, less that half of the villages have one. Let the total number of HCC´s in the region be denoted by $n_{hcc} < n$.
Your friend Charlie has heard about the shortest path (SP) problem and the Travelling salesperson (TSP) problem. Somehow the delivery problem seems related to those, but Charlie doesn´t understand exactly how, neither which of the algorithms to use. Or maybe none of them can´t be used and a new algorithm has to be designed?
Help your friend to suggest a suitable model for the problem and help to design an efficient method that outputs a plan (sequence in which the HCC´s should be visited) for the truck driver!

You can assume that you have complete knowledge of the road network, i.e. which villages are connected with a road and how long the driving time is for each connection. You also know which villages have HCC´s and which don´t. The vaccine storage and location of the truck (at the beginning and end of each day) is located next to one of those HCC´s. To simplify matters, we neglect driving times within villages and only care about driving times between villages (northern Sweden - small villages, but large distances between villages).
Both the shortest path problem and TSP need a graph as input, so you need to specify the details of the graph if you want to use any of the know algorithms for SP or TSP. Even if you come up with your own algorithm, you need to specify the input to that algorithm.
On any given day, the truck will only deliver to a limited number of HCC´s, assume delivery to at most 10 HCC´s per day.

Below we ask you to give algorithms and running time analyses for two different scenarios. You can choose if you want to describe your algorithms in words or in pseudo-code (or both). Make sure to include enough details in your description/pseudo-code so that we can understand your running time analysis. If you use a SP or TSP solver in your algorithm, you don´t have to repeat details of those procedures in your solution. Make sure to suggest the most efficient method (in terms of $O()$) you can think of!

5.1 Assume Charlie is asked to solve the above problem for one single day only. Suggest an efficient method and provide a running time analysis in terms of $O()$. [6 points]

5.2 Now assume Charlie needs a method that can be applied every day for many weeks ahead. Suggest an efficient method and provide a running time analysis in terms of $O()$ [6 points]

Short solutions (attached after the exam)

1.1 Run BFS starting with v_1. BFS creates levels. Assume v_2 is on level l in the BFS tree, then the length of the SP from v_1 to v_2 is kl. The nr of paths with that length are the sum of the nr of paths to nodes in level $l-1$ that are connected to node v_2 in level l. BFS runs in $O(|E|+|V|)$ and calculating the nr of SP paths for each node in the BFS tree takes in total not more than $O(|E|)$. Since $|V| \in O(|E|)$ for connected graphs, we have $O(|E|)$

1.2 E.g. if the graph consists of a single path from s to t and e^{max} is somewhere on the way (a bottleneck edge).

1.3 $f(n) \in O(g(n))$ but not vice versa, see definition of $O()$ from the course material. Example for the second statement: $k = 2$, $f(n) = 2^n$ and $g(n) = 3^n$.

1.4 False. If the original graph is a tree then BFS and DFS return the same tree.

2.1 Check the borders of the array, if array top found, stop. Check the middle element a_m, if array top found, stop. Else continue searching in the half with the larger neighbour of a_m. Continue until only 1-2 elements left (base cases).

2.3 $T(n) = T(n/2) + c$, gives $O(\log n)$, essentially binary search, reducing the array by half in each step.

3.2.1 $OPT(i, m)$ is the max points we can achieve using the first i tasks and at most m hours.

3.2.2 $OPT(i, m) = \max_{0 \le j \le m}\{OPT(i-1, m-j) + x_{ij}\}$

3.2.3 $OPT(0, m) = 0$

3.2.4 Build up a table of size $n \times M$, for each entry, we compute at most M terms according to 3.2.2. Return $OPT(n, M)$.

3.2.5 $O(nM^2)$

3.3 E.g. for $p = 1$ or if all x_{ij} are equal.

4.2 HIKING is in NP: Given a set of items and their distribution into the backpacks, sum up their w_i and check against C. This can be done in polynomial time (linear).

4.3 and 4.4. HIKING is identical to PARTITION (see Half-half subset sum from the lecture notes). The reduction and proof can either be with PARTITION (trivial) or with Subset-sum (see standard literature).

5 The solution below assumes that the HCC´s to be visited are given.[1]

5.1 Pre-processing: Given $G = (V, E)$ where the nodes are all villages in the region and the edges the known distances. Build graph $G' = (V', E')$ where the nodes are only the villages with HCC´s that we need to visit today plus the village with the truck location (if needed). The edges will be the connections between the HCC´s in V' (possibly passing through some villages in the original graph G). Find the edge costs by calculating the shortest path in G for each pair of nodes that will be in V'.

Alg: Run a TSP solver on G' (enumerating all possible tours in $O(|V'|!)$).

Running time: Max nr of nodes $|V'|$ in the new graph is 11. Assume we run Dijkstra for each pair of the 11 nodes $O(n^2)$. Total running time: Read the entire input to the graph $O(n+|E|)$, then construct G' in $O(n^2)$. The TSP routine is constant since the number of HCC´s per day is limited by a constant. Altogether $O(n^2)$.

5.2 Assume we need to schedule for D days. The 5.1-method is $O(Dn^2)$.

OR: Pre-processing 1: Given a graph $G = (V, E)$ with nodes being all villages in the region and the edges the known distances between the villages. Build $G' = (V', E')$ where the nodes V' are only villages with HCC´s. The edges E' will be direct connections between the HCC-villages. Find the edge costs for E' by running a shortest path solver in G for each pair of HCC´s.

Pre-processing 2: For each new day, construct G'' (sub-graph of G') with max 11 relevant villages to be visited. Construction of G'' is straightforward since G' is a complete graph.

Alg: Run a TSP solver on G'' for each of the D days.

Running time: Reading the input takes $O(n + |E|)$, construction of G' takes $O((n_{hcc}^2 \cdot n^2)$, construction of G'' takes at most $O(|V'| + |E'|)$, where $|E'| \in O(n_{hcc}^2$. The TSP routine takes constant time for each day. In total: $O((n_{hcc}^2 n^2 + n_{hcc}^2 D))$.

The second method is preferable if $D > n_{hcc}^2$.

[1] If the truck driver can choose which HCC to deliver to, that changes the problem (and it makes no sense for 5.2). The free-choice-version can be solved in polynomial time, too, but the details are tricky.