CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
TIN093	Algorithms			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
TIN093-47		2016-10-22	6	5

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	7	
2	X	6	
3	X	7	
4	X	8	
5	X	8	
6	X	13	
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
Total examination points Summa poäng på tentamen		49	

Family name+First name (Blockletters) Efternamn+Förnamn+Initialer(textas)	Signature Namnteckning
THORSELL ERIK E.T	[illegible]

Date of Birth Year Month Day Personnummer år mån dag	Identification no nummer	Programme acronym Program	Year of Admission Antagningsår
[illegible]	[illegible]	MP[illegible]	2016

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	TIN093-47	7/10	1
			Question no. / Uppgift nr: 1

$e(A,B)$ = edit distance of A and B.
AC = A concatenated w C.

1.1
The edit distance between C and C = $e(C,C)=0$ since the words are the same. Given any two words A and B we can concatenate them w C to get AC and BC but we can always align AC and BC s.t. the C-part of the word is a perfect match.
Hence $e(AC,BC) \leq e(A,B)$.

But why can't we align them better? 2/4

1.2

```
A L G O R I T H M
L O G A R I T H M
```

Since the last 5 characters is a perfect match this candidates to be a good alignment of the words. In task 1 we showed that adding a substring to the end of two words does not change the edit distance. Hence we are really looking to solve the edit distance between:

```
A L G O
L O G A
```

5/6

The G in ALGO and LOGA aligns well (and moving the words to get another alignment does not yield a better result). We therefore need to change 3 letters in one of the words for them to be equal.

~~[illegible]~~

Why doesn't it?

1

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	TIN093-47	6	2
			Question no. / Uppgift nr 2

2.1

$$R(k, x, y, z) = R(k-1, x-\omega_k, y-\omega_k, z) \lor R(k-1, x-\omega_k, y, z-\omega_k) \lor R(k-1, x, y-\omega_k, z-\omega_k)$$

6

If it is not possible to store ω_k on two disks there exist no solution and the algorithm will return 0.

The answer is given by $R(n, U, U, U)$.

2.2

The algorithm runs in $O(nU)$ time, pseudopolynomial. More on this in question 3.

0

?

can't answer a question in another questions answer.

PS: sounds like a newscaster :)

1

CHALMERS

Anonymous code / Anonym kod: TIN093-47	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare): 7	Consecutive page no. / Löpande sid nr: 3
		Question no. / Uppgift nr: 3

Disks: D_0, D_1, D_2
Capacity: U
One file exists w size: $U/2$ and all files have total size $\frac{3U}{2}$.

3.1

1 An algorithm belongs to NP if it is possible to verify a solution of the algorithm in polynomial time. If given a suggestion on how to store our files we can check if the suggestion is a valid storage configuration in polynomial time. We simply store the files as suggested and if they fit the solution is valid. ?

3.2 HalfHalf Subset Sum $\leq_p$ Redundant Storage (HS $\leq_p$ RS)

Assume we have an algorithm capable of solving RS. We want RS to suceed for some instance x iff there exist a subset with sum $w = \sum_{i=0}^{n} \frac{w_i}{2}$ in x.

4 Let $U = \sum_{i=1}^{n} w_i$, add to set of numbers an element of size $\frac{U}{2}$. Iff RS is able to fit the "files" into 3 disks of size U there exists a subset of size w.

This is true because the extra element of size $\frac{U}{2}$ forces all original elements to be written to D_0 and (D_1 xor D_2). Hence we must be able to fit half of the set into D_1 and half into D_2 for RS to work.

This takes $O(n)$ time. ✓

3.3

2 No! It is merely pseudopolinomial! U is exponential in the size of the input. It is polynomial only in the magnitude of input numbers.

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

1

CHALMERS

Anonymous code / Anonym kod: TIN093-47	Points for question / Poäng på uppgiften	Consecutive page no. / Löpande sid nr: 4
		Question no. / Uppgift nr: 4

4.1

The recurrence equation for the number T(n) of comparisons:

$$T(n) = 1 \cdot T(\tfrac{n}{3}) + cn^0$$

Here the 1 represents the number of subinstances we look at.
The 3 is the divisor. We split each instance into three. The instance has size n.
The zero comes from the fact that we do a constant number of operations each iteration.

In the master theorem we would have: $a=1$, $b=3$, $k=0$.

5

4.2

$T(n) = T(\frac{n}{3}) + O(1) \Rightarrow T(n) = O(\log n)$ since $a = 1 = b^k$ in accordance w the master theorem.

3

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	TIN093-47		5
			Question no. / Uppgift nr 5

k persons
$G=(V,E)$ with n nodes and m edges and a subset $T \subset V$ of k terminal nodes. All edges have length 1.

Problem: Find a node $v \in V$ w the same distance to all nodes $t \in T$, also minimize the solution.

If we use BFS it takes $O(m)$ time per run. BFS will return the distance between the start node and all the other nodes in the graph. If we run BFS from each terminal node it would take $O(km)$. We would save each node's "distance from terminal node t_i $(1 \leq i \leq k)$ while running the algorithm $(O(n))$ and then check these values to see if any node have the same distance to all terminal nodes. If several exist, we pick the smallest. $O(n)$.

8

slightly confusing writing but OK

CHALMERS	Anonymous code / Anonym kod: TIN093-47	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 6 — Question no. / Uppgift nr: 6

$G=(V,E)$ is an undirected graph with unit edge lengths.
Assume there is a clique $C \subseteq V$ w k nodes.

5.1
Since we have only changed the clique it's enough to look at the nodes and edges connected to any node in the clique.

The reason the modification works is because the distance between any two nodes in a clique is 1. Hence as soon as you reach a node $V_c \in C$ there is only 1 step to any other node in C.

If we want to reach a node "on the edge of C", that is at least one edge incident to V_c is not part of the clique, all is well if we are coming from outside C. If we are already inside C this is true as well since the distance between two nodes used to be 1 and is now $1/2 + 1/2 = 1$.

If we are outside of the clique and want to reach a node that is not on the edge of C we will make our way to C and then take the one step $= 1/2$ to the new node and then one step $= 1/2$ to any other node, hence the modification works for this case too.

5

5.2
When $k=4$ it makes a difference. A complete graph of 4 nodes has 6 edges but our modified version has only 4. An undirected clique contains $O(k^2/2)$ edges whereas our modified version contains $\Theta(k)$ edges.

Difference for general k?

2

5.3
Since Clique $\in$ NPC it seems like a bad idea. BFS can be used to get the shortest path from a node to all other nodes in $O(m)$ time. What we want to know is if it's a good idea to: look for a clique (exponential time) and swap the edges to a new edge (have to look at all edges in the clique). When this is done we will have looked at all the edges in the clique (and the graph) already, so we could have used BFS from the start.

4

5.4
Since k-Clique only works for a fixed k we cannot solve the general clique problem.

In order to find a k-Clique you could check if you get k nodes left after deleting all nodes with fewer or more than k-1 incident edges.

→ Does this work in every case? (Is the condition sufficient?)

And what is now your conclusion?

2

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan