CS29003 ALGORITHMS LABORATORY
Dynamic Programming
Last Date of Submission: 13 – September – 2018

General Instruction

1. Please do not use any global variable unless you are explicitly instructed so.
2. Please use proper indentation in your code.
3. Please name your file as <roll_no>_<assignment_no>. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c or 14CS10001_3.cpp` as applicable.
4. Please write your name and roll number at the beginning of your program.

A truck manufacturing farm wishes to find the capacity of their trucks. Suppose the capacity of every truck is $\mathcal{C}$ tonnes for some positive integer $\mathcal{C}$. The only way to get an idea of the capacity of the trucks is to load any truck with some goods of weight $\mathcal{W}$. The truck breaks down if and only if $\mathcal{W} > \mathcal{C}$. A truck, once breaks down, becomes useless. The farm knows from their engineers that the capacity of their truck is at most $\mathcal{T}$. The farm wishes to find $\mathcal{C}$. However, the firm does not want to lose too many of their trucks. So the firm agrees to give you $\mathcal{N}$ number of their trucks which you can use for your experimentation. Since loading tonnes of goods take a lot of time and effort, you can perform one test per day; a test is loading some tonnes of goods in the truck and observe whether the truck breaks down or not. Your job is to find $\mathcal{C}$ as early as possible in the worst case. Convince yourself that if $\mathcal{N} = 1$, then any strategy which is guaranteed to find $\mathcal{C}$ needs at least $\mathcal{T}$ days in the worst case and there is obviously a strategy which finds $\mathcal{C}$ in at most $\mathcal{T}$ days. Let us define the function $g(\mathcal{N}, \mathcal{T})$ which takes as input the number $\mathcal{N}$ of trucks available for experimentation and the upper bound $\mathcal{T}$ on the capacity $\mathcal{C}$ of the trucks and maps it to the minimum number of days one needs in the worst case to find $\mathcal{C}$. For example, $g(1, \mathcal{T}) = \mathcal{T}$.

In this exercise, we will devise dynamic programming based algorithms to find the number of days $g(\mathcal{N}, \mathcal{T})$.

Part I: A $\mathcal{O}(\mathcal{N}\mathcal{T}^2)$ Algorithm

Let $\mathcal{A}(n, \ell)$ = the minimum number of days one needs in the worst case to find $\mathcal{C}$ using at most n trucks assuming $\mathcal{C} \leqslant \ell$. Then we have $\mathcal{A}(n, 0) = 0, \mathcal{A}(1, \ell) = \ell$ for every positive integer n and ℓ. Convince yourself that the following recurrence holds:

$$\mathcal{A}(n, \ell) = 1 + \min\{\max\{\mathcal{A}(n-1, x-1), \mathcal{A}(n, \ell - x)\} : 1 \leqslant x \leqslant \ell\}$$

Implement the above dynamic programming using the following prototype.

int findMinimumDays(int $\mathcal{N}$, int $\mathcal{T}$)

The function findMinimumDays takes the number of trucks $\mathcal{N}$ and the upper bound $\mathcal{T}$ on $\mathcal{C}$ as input and returns $g(\mathcal{N}, \mathcal{T})$. Please assume that $\mathcal{N} \geqslant 1$. You can use one 2-dimensional array of size $O(\mathcal{N}\mathcal{T})$ to implement the above dynamic programming based algorithm (although two one dimensional array of size $\mathcal{O}(\mathcal{T})$ is enough). Please dynamically allocate such an array and make sure you free it once you do not need it anymore.

Part II: A $\mathcal{O}(\mathcal{N}\mathcal{T}\log\mathcal{T})$ Algorithm

Design a dynamic programming based algorithm for finding $g(\mathcal{N}, \mathcal{T})$ in time $\mathcal{O}(\mathcal{N}\mathcal{T}\log\mathcal{T})$. You can use one 2-dimensional array of size $O(\mathcal{N}\mathcal{T})$ to implement your dynamic programming based algorithm. Implement your algorithm using the following prototype.

int findMinimumDaysFaster(int $\mathcal{N}$, int $\mathcal{T}$)

The function findMinimumDaysFaster takes the number of trucks $\mathcal{N}$ and the upper bound $\mathcal{T}$ on $\mathcal{C}$ as input and returns $g(\mathcal{N}, \mathcal{T})$. Please assume that $\mathcal{N} \geqslant 1$. You can use one 2-dimensional array of size $O(\mathcal{N}\mathcal{T})$ to implement the above dynamic programming based algorithm (although two one dimensional array of size $\mathcal{O}(\mathcal{T})$ is enough in this case too). Please dynamically allocate such an array and make sure you free it once you do not need it anymore.

main()

1. Take $\mathcal{N}$ and $\mathcal{T}$ as input from the user.
2. Find $g(\mathcal{N}, \mathcal{T})$ using findMinimumDays and output it.
3. Take $\mathcal{N}$ and $\mathcal{T}$ as input from the user again.
4. Find $g(\mathcal{N}, \mathcal{T})$ using findMinimumDaysFaster and output it.

Sample Output 1:

Write n: 3
Write t: 100
g(3, 100) = 9
Write n: 2
Write t: 100
g(2, 100) = 14

Sample Output 2:

Write n: 3
Write t: 200
g(3, 200) = 11
Write n: 20
Write t: 1000
g(20, 1000) = 10