Ugeopgave 2

Mathias Bramming Philip Meulengracht
Rasmus Borgsmidt

Task 1

Optimal Substructure

The problem exhibits optimal substructure because any optimal solution to a problem of size $n > 0$ contains within it optimal solutions to subproblems. If for example we consider a sequence of n heights $\{h_1, \ldots, h_n\}$, an optimal solution s, can be obtained by adding to optimal solutions $s' = \{s'_1, \ldots, s'_p\}$ and $s'' = \{s''_1, \ldots, s''_q\}$, $p, q \leq n-1$,for the smaller sequence $\{h_1, \ldots, h_{n-1}\}$:

$$s = \begin{cases} \{s'_1, \ldots, s'_p, h_n\} & s'_p < h_n \text{ and } s'_{p-1} > s'_p \\ \{s''_1, \ldots, s''_p, h_n\} & s''_p > h_n \text{ and } s''_{p-1} < s''_p \\ s' & \text{otherwise} \end{cases}$$

In the event that both s' and s'' exist to satisfy the two first clauses above, either result will be an optimal solution. If $n = 0$, the empty subsequence is the only optimal solution.

Overlapping Subproblems

The problem exhibits overlapping subproblems because for each iteration, we need to consider multiple optimal solutions for each subproblem. Therefore unless information for these is retained, the algorithm will need to solve the same subproblem more than once.

Task 2

The length of a longest zig-zag subsequence of $\{h_1, \ldots, h_n\}$ with largest element $m = \max\limits_{1 \leq k \leq n}(h_k)$ can be described by the following recursive definition:

$$l_n = \max(u_{[n,1]}, d_{[n,m]})$$

$$u_{[n,j]} = \begin{cases} 0 & n = 0 \\ \max\left(u_{[n-1,j]}, 1 + d_{[n-1,h_n]}, \max\limits_{j<k\leq m}(u_{[n,k]})\right) & n > 0 \text{ and } j < h_n \\ u_{[n-1,j]} & \text{otherwise} \end{cases}$$

$$d_{[n,j]} = \begin{cases} 0 & n = 0 \\ \max\left(d_{[n-1,j]}, 1 + u_{[n-1,h_n]}, \max\limits_{1\leq k<j}(d_{[n,k]})\right) & n > 0 \text{ and } j > h_n \\ d_{[n-1,j]} & \text{otherwise} \end{cases}$$

In this definition, each $u_{[n,j]}$ denotes the length of a longest subsequence 'zigging' up to a value $v > j$, and each $d_{[n,j]}$ denotes the length of a longest subsequence 'zagging' down to a value $v < j$.

Task 3

We prove the correctness of the algorithm using mathematical induction over n.

The base case is $n = 0$, the empty sequence of bottles. For this sequence, any subsequence is trivially empty and therefore identical to the full sequence. This makes the empty sequence an optimal solution to the problem for $n = 0$.

Turning to the inductive step, we assume that the definitions presented in the answer to Task 2 of $u_{[n-1,j]}$ and $d_{[n-1,j]}$ hold for $n > 0$ and $1 \leq j \leq m$.

Looking at the definition of $u_{[n,j]}$, we recognize two possible cases: If $j < h_n$, we have $u_{[n,j]} = \max(u_{[n-1,j]}, 1 + d_{[n-1,h_n]}, \max_{j<k\leq m}(u_{[n,k]}))$. This is correct since $u_{[n,j]}$ denotes the length of a longest subsequence zigging up to a value $v > j$; either this subsequence already zigged up to v earlier in the sequence than n, or it zagged down to a value $v' < j$, in which case adding h_n to the subsequence increases its length by 1, or it is produced by one of the $u_{[n,j']}$, where $j' > j$. If $j \geq h_n$, h_n cannot usefully contribute to the subsequence and we get $u_{[n,j]} = u_{[n-1,j]}$.

Similarly looking at the definition of $d_{[n,j]}$, we recognize two possible cases: If $j > h_n$, we have $d_{[n,j]} = \max(d_{[n-1,j]}, 1 + u_{[n-1,h_n]}, \max_{1\leq k<j}(d_{[n,k]})))$. Again this is correct since $d_{[n,j]}$ denotes the length of a longest subsequence 'zagging' down to a value $v < j$; either this subsequence already zagged down to v earlier in the sequence than n, or it zigged up to a value $v' > j$, in which case adding h_n to the subsequence increases its length by 1, or it is produced by one of the $d_{[n,j']}$, where $j' < j$. If $j \leq h_n$, h_n cannot contribute and we get $d_{[n,j]} = d_{[n-1,j]}$.

This concludes the inductive step, and by the principles of mathematical induction we have shown the correctness of $u_{[n,j]}$ and $d_{[n,j]}$ for $n \geq 0$ and $1 \leq j \leq m$. Given that $u_{[n,1]}$ denotes the length of a longest subsequence zigging up to a value $v > 1$ and that $d_{[n,m]}$ denotes the length of a longest subsequence zagging down to a value $v' < m$, an optimal solution has to be found by one or both of them. Therefore l_n as defined in the answer to Task 2 is also correct.

Task 4

Running Time

The max operation in definition l_n itself takes $\Theta(1)$ time. And if $n = 0$, both $u_{[n,j]}$ and $d_{[n,j]}$ are also $\Theta(1)$. So in this case the running time $T(n) = \Theta(1)$.

Now let $T_u(n)$ be the running time of $u_{[n,j]}$ and $T_d(n)$ be that of $d_{[n,j]}$. When $n > 0$, $u_{[n,j]}$ takes $\Theta(1)$ time for the outer max operation, $T_u(n-1)$ time for the first term, and $1 + T_d(n-1)$ time for the second term. Looking at the two first terms, we see that for each recursion n is decreased by 1. Therefore the running time for each of these is $\Theta(n)$. The running time of the last term is $(m-j)T_u(n)$, which means we get a total of $m - j + 1$ times the running costs embedded in $u_{[n,j]}$, that is $T_u(n) = (m - j + 1)(\Theta(1) + \Theta(n) + \Theta(n)) = \Theta(mn)$.

By the same argument, we get $T_d(n) = \Theta(mn)$. And putting it all together:

$$T(n) = \begin{cases} \Theta(1) & n = 0 \\ \Theta(1) + \Theta(mn) + \Theta(mn) = \Theta(mn) & n > 0 \end{cases}$$

Memory Usage

A simple implementation of this algorithm could store the calculated values for $u_{[n,j]}$ and $d_{[n,j]}$, respectively, in two $(n \times m)$-tables and use these for lookups. This would make the memory usage $\Theta(mn)$.

Task 5

Still to do.