Assignment 3 — AD

Schmidt, Victor Alexander, `rqc908`
Ibsen, Helga Rykov, `mcv462`
Larsen, Christian Vadstrup, `ljq919`

Monday 08:00, March 6th

Task 1

Algorithm 1 Is_$\bar{b}$_beers_possible(p, b, $\bar{b}$)

```
for i = 1 to n − 1 do
    toll = 2(p_{i+1} − p_i)
    if (b_i < b̄) or (b_i > b̄ and toll < b_i − b̄) then
        beers to send = b̄ − b_i
        b_{i+1} = b_{i+1} − (beers to send + toll)
        b_i = b_i + beers to send
    end if
end for
if b_n ≥ b̄ then
    return True
end if
return False
```

Task 2

The greedy choice property in **Algorithm 1** is b_i. We make sure that the current bar has $\bar{b}$ beers, by greedily importing $\bar{b} - b_i$ beers from the next bar over, no matter how many beers the next bar has. If we have a surplus of beers at bar i that exceeds the *toll* (the amount lost in translation); $toll < b_i - \bar{b}$, we send the surplus to the next bar over instead.

This way, bar i always have $b_i \geq \bar{b}$ and bar $i+1$ has the optimal substructure, while making the greedy choice for bar i. The final problem then is if bar n has $b_n \geq \bar{b}$ beers, since we greedily solved for all other bars in order, then there exists an optimal solution.

Loop invariant proof

To prove the correctness of **Algorithm 1**, we need to show that the loop invariant holds true for each iteration of the for-loop. The for-loop maintains the following invariant:

At the start of each iteration of the for-loop,
all the elements of b before the i-th index
have a value greater than or equal to $\bar{b}$.

We use the loop invariant as follows:

Initialization: At the start of the loop, we assume that the loop invariant holds true. Before the first iteration, we have $i = 1$. Therefore, the loop invariant is that for all $j \in [1, i)$, $b_j \geq \bar{b}$.

Maintenance: We need to show that if the loop invariant holds true for i, then it also holds true for $i+1$. In each iteration, we calculate the toll as $toll = 2(p_{i+1} - p_i)$ and check if the current value of b_i is less than $\bar{b}$ or greater than $\bar{b}$ with the difference between them greater than the toll. If the condition is true, we calculate the number of beers to send as beers to send $= \bar{b} - b_i$ and update the values of b_{i+1} and b_i accordingly. Now, we need to show that the loop invariant still holds true for $i+1$.

If $b_i \geq \bar{b}$, then we do not update the values of b_i and b_{i+1}. Therefore, the loop invariant still holds true for $i+1$.

If $b_i < \bar{b}$ or $b_i > \bar{b}$ and $toll < b_i - \bar{b}$, then we update the values of b_i and b_{i+1} such that $b_{i+1} = b_{i+1} - (\text{beers to send} + toll)$ and $b_i = b_i + \text{beers to send}$. Since beers to send $= \bar{b} - b_i$ and $toll = 2(p_{i+1} - p_i)$, we have $b_{i+1} \geq \bar{b}$ and $b_j \geq \bar{b}$ for all $j \in [1, i+1)$. Therefore, the loop invariant holds true for $i+1$.

Termination: The loop terminates when $i = n - 1$. At this point, we need to check if $b_n \geq \bar{b}$. If this condition is true, then we return True. Otherwise, we return False. Since the loop invariant holds true for all $j \in [1, n-1)$, we know that $b_n \geq \bar{b}$ if and only if $b_j \geq \bar{b}$ for all $j \in [1, n)$. Therefore, the termination condition also satisfies the loop invariant.

In conclusion, the loop invariant holds true for the entire for-loop and **Algorithm 1** is correct. ∎

Task 3

Algorithm 2 Find_max_$\hat{b}$(p, b)

$\hat{b} = 0$
$L = 1$
$R = \max\{b\}$
while $L \leq R$ **do**
  $m = \lfloor \frac{L+R}{2} \rfloor$
  if Is_$\bar{b}$_beers_possible(p, b, m) **then**
    $L = \hat{b} + 1$
  else
    $R = \hat{b} - 1$
  end if
  $\hat{b} = L$
end while
return $\hat{b}$

This algorithm runs in $O(n \log B)$ time, where $B = \max\{b\}$ (the maximum amount of beer in any bar). The algorithm draws heavy inspiration from the binary search algorithm, as finding $\hat{b}$ is like exactly the same as finding a *key* in a sorted array. The sorted array that we simulate binary seach on would be $\{1, 2, ..., B-1, B\}$. We know binary search takes $O(\log n)$ time, which in our case is $O(\log B)$ time.

As we already know the running time of "Is_$\bar{b}$_beers_possible(p, b, $\bar{b}$)" has been proven in **Task 2** to run in $O(n)$ time, and since we know the while-loop runs this function $\log B$ times, we have the following runtime:

$$O(n) \cdot O(\log B) = O(n \log B)$$