H2.Hamlet

- The assignment is due at Gradescope on Tuesday, January 26 at 12 noon.
- You can either type your homework using LaTex or scan your handwritten work. We will provide a LaTex template for each homework. If you writing by hand, please fill in the solutions in this template, inserting additional sheets as necessary. This will facilitate the grading.
- You are permitted to study and discuss the problems with 2 other students (per problem; any section). However, *you must write up your own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Similarly, please list any other source you have used for each problem, including other textbooks or websites.
- *Show your work.* Answers without justification will be given little credit.

PROBLEM 1 (25 POINTS) *Solve exercise 3 in Chapter 4 in the Kleinberg-Tardos textbook. (trucking)*

Solution: Collaborated with: Yael Sulkin, Bryan Lee

Algorithm The greedy algorithm this proof will be proving is the following:

Let all trucks have maximum weight, W
Let the number of trucks sent to Boston, $numTrucks$.
Let the truckweight at kth iteration be $truckWeight$.
Let i be the package that most recently arrived, woth weight of i be w_i

```
while packages arrive do
    if truckWeight + w_i < W then
        update truckweight = truckweight + w_i
    else
        increase numTrucks by 1
        reset truckWeight to 0
    end if
end while
```

Proof We will prove this algorithm inductively. For our greedy algorithm we want to show that it "stays ahead" or any other solution such that at the kth iteration our algorithm is at most the same as the optimal solution. Let A represent the number of trucks used by our greedy algorithm, O represent the number of trucks used by the optimal solution, and let A_k, O_k represent the respective quanity of trucks used at any given iteration k. Four our base case, we set $k = 1$. When our first package arrives to the station it is loaded on to the first truck in both algorithms. We know that for packagae i, $w_i \leq W$ which means the number of trucks used is at most 1. This solved our base case. Now, let $k > 1$ and assume our greedy algorithm stays ahead for $j < k$ iterations. We want to show that when $A_k \leq O_k$ at any given iteration then $A_{k+1} \leq O_{k+1}$. That is our inductive hypothesis and we can prove it using case analysis. We can break the cases down to $A_k = A_{k+1}$ and $A_k < A_{k+1}$. First if we know that $A_k \leq O_k \leq O_{k+1}$ and $A_k = A_{k+1}$ then we know $A_{k+1} \leq O_{k+1}$ which proves our inductive hypothesis for this case. Secondly, if $A_k < A_{k+1}$ ... not finished

Extra Space for your solution

PROBLEM 2 (25 POINTS) *Solve exercise 5 in Chapter 4 in the Kleinberg-Tardos textbook. (cell phone towers)*

Please note! Here and elsewhere, when the authors (or your instructors) say "give an algorithm" without further instructions, they mean "give an algorithm AND prove that it is correct and runs in polynomial time". This should be assumed henceforth.

Solution: Collaborated with: Yael Sulkin, Bryan Lee

Let h be in set H of all houses along the road.
Let x_h be the locatin of house h where x_h = 0 if at westernmost point and the easternmostpoint is at x_h max.
Let B represent the set of base towers placed
while $h \in H$ **do**
    if $x_h + 4 \geq$ easternmost point **then**
        place base on easternmost point
        add base to set B
        delete all houses in H s.t. $x_4 + 4 \geq$ easternmost point
    else
        place a base in $x_h + 4$
        add base to set B
        delete all houses in H s.t. $x'_h \leq x_h + 8miles$
    end if
end while

We know our algorithm runs on polynomial time because our loop iterates over the set of houses once, returning the set of bases after reaching the last house. Our algorithm runs on $O(n)$ time by definition.

Proof Let set A be the set of bases placed by our algorithm and set O be the set of bases placed by the optimal algorithm. If our set A is not optimal and we are looking for the minimal amount of bases then we know that if $|A| = n$ and $|O| = m$ then $m < n$. If $m < n$ then we know there exists some base, b in A such that it either covers 0 houses or there also exists some b' such that b' can be moved to cover the houses within b. We will continue this proof using case analysis and proof by contradiction.

In the case where b in A covers 0 houses, this, in fact, contradicts the algorithm for A which states "while there exists house $h \in H$". This case is impossible for our proposed greedy algorithm.

In the case where b' exists in A such that b' can be moved, let b be located at distance x and b' be located at distance x' such that $x \neq x'$ and $x, x' \geq 0$. According to our case statement, there exists h in $x - 4, x + 4$ such that b' can be moved to cover $x - 4, x + 4 \cup x' - 4, x' + 4$ or $x - 4, x + 4, x' - 4, x' + 4$. b' will have to at most cover $|(x - 4) - (x' + 4)| = |x - x' - 8|$. If the distance b' has to cover is greater than 8 this contradicts our givens. If $|x - x' - 8| = 0$ this means base b and base b' are exactly 8 meters apart. Let b be west of b'. This implies there exists some h such that x_h is exactly halfway between b and b'. However, this is impossible because according to our algorithm b' was placed 4 miles east of house $h \in H$ yet h was not in H at that iteration because x_h was covered by base b and therefore deleted from available input.

Extra Space for your solution

PROBLEM 3 (25 POINTS) *The Running Sums problem is defined as follows:*

Input: *a sequence $(a_1, \ldots, a_n)$, where each a_i is either 1 or -1.*

Desired output: *a sequence $(b_1, \ldots, b_n)$, where each b_i is either 0 or 1.*
*Your goal is to **minimize** $\sum_{1 \le i \le n} b_i$, subject to the **constraint** that*

$$\sum_{1 \le i \le j} (a_i + b_i) \ \geq \ 0 \ , \qquad \text{for all} \quad j \in \{1, 2, \ldots, n\} \ .$$

Give an algorithm for this problem that, for full credit, should run in $O(n)$ steps.

Solution: Collaborated with: Yael Sulkin and Bryan Lee

Let b start off as 0
Let J represent the running sum of a_i and b_i
Let B represent the running sum of b_i
while a_i in $a_1 \ldots a_n$ **do**
  if $J > 0$ **then**
    b_i remains 0
    Update J and add a_i and b_i
    Keep B the same
  else
    b_i can be derived by the equation $b_i = \frac{a_i}{2} + \frac{1}{2}$
    Update J by add a_i and b_i
    Update B by adding b_i
  end if
end while

This algorithm runs on O(n) time as it loops through sequence a once while building towards b at every step.

Proof An optimal algorithm increases b minimally at each step and we know b can either be 0 or 1 so our greedy algorithm's goal is to maximize the number of 0's in b and limit the number of 1's in b. As such, we want more cases where b should be 0 and less otherwise. Thus, we will prove our greedy algorithm with contradiction. Let A represent the sum of b's of my algorithm and O represent the sum of b' of the optimal solution. We want to show $A \leq O$ at any given iteration k. Let A_k, O_k represent the respective sum of b at kth iteration. We will treat each k iteration using case analysis, particularly looking at why b would ever want to be 1. The only scenario would be if $a_i = -1$ and the current running sum at kth iteration of $a_i + b_i = 0$. In this scenario, b can not be 0 or our running sums constraint would be broken. However, let $O \leq A$ then we know there exists at least one b in which at some kth iteration, given some a_k, b should be 0 and not 1, such that $O_{k+1} increasesby 0$. Well if $a_i + b_i = 0$ and $O_{k+1} = 0$ then we know $a_i was 1$ which for our algorithm always produces a value of 0. -not finished

Extra Space for your solution

PROBLEM 4 (25 POINTS) *In the Hopping Game, there is a sequence of n spaces. You begin at space 0 and at each step, you can hop 1,2,3, or 4 spaces forward. However, some of the spaces have obstacles and if you land on an obstacle, you lose.*

Give a greedy algorithm which, given an array $A[1, \ldots, n-1]$ of Boolean values with $A[i]$ indicating the presence/absence of obstacle at position $i \in [1, n-1]$, find the minimum number of hops needed to reach space n without losing, if it is possible to do so. (We assume spaces 0 and n are obstacle-free, and are not part of the input.) Prove that your algorithm is correct. For full credit, your algorithm should run in time $O(n)$.

Solution: Collaborated with: Yael Sulkin

```
while for every input a_i in A do
    if a_{i+4} is 0 then
        add to hops counter
        change location to a_{i+4}
    else
        if a_{i+3} is 0 then
            add to hops counter
            change location to a_{i+3}
        else
            if a_{i+2} is 0 then
                add to hops counter
                change location to a_{i+2}
            else
                if a_{i+1} is 0 then
                    add to hops counter
                    change location to a_{i+1}
                else
                    Return hops counter, Terminate loop. Reaching N is not possible
                end if
            end if
        end if
    end if
end while
```

This algorithm has runtime O(n) since we work our way through array A from left to right once and never loop over any square multiple times. Our algorithm's runtime is proportional to the size of A which by definition is O(n).

Proof Let the number of hops made by our algorithm be denoted by H and by the most optimal algorithm be denoted by O. We will prove our algorithm provides at most the same amount of hops as the optimal solution using proof by induction. We will denote the number of hops at iteration, k, made by our greedy algorithm as H_k and those by the optimal algorithm as O_k. Our base case is when $k = 1$. When $k = 1$, both algorithms are starting from space 0. The optimal algorithm will jump once if there is indeed a path to space n, thus, $O_1 = 1$. Similarly, our greedy algorithm will jump at least some distance if there is an avaialable square within the next 4 tiles thus $H_1 = 1$. Our base case is proved. We must show that for the $k + 1$st case our greedy algorithm stays ahead and at each iteration $H \leq O$. Well, for each hop in the optimal algorithm, the player gets closest to n such that the distance between the player and space n is minimal after each hop. We want to show our greedy algorithm minimizes the distance to space n with each hop. For the optimal algorithm we know 1 hop = n - (hop distance + a_i) where $n - (hopdistance + a_i)$ is minimized. Since, n is a constant and a_i is also a constant representing current location, we know that our greedy algorithm must maximize hop distance at each iteration. Our greedy algorithm does, in fact, by assuming maximum hop distance and working our way to smaller distances only if greater ones aren't available.

Extra Space for your solution