1. In order to expand his brand, Sadat decides to launch a board game, Shaiks and Ladders. Despite its name, the game is more like Monopoly. The game board consists of a straight line of tiles, with each tile paying out a money bonus or causing the player to lose money. The player must choose a starting tile, and can then move along as many tiles as they want in consecutive ascending order, winning or losing money based on the tiles passed.

   Ben thinks Sadat is a better businessman than game designer, and wants to create an algorithm that will find the sequence of tiles to pass over that will maximize the amount of money he can make.

   (a) Show that it is possible to find the best sequence of tiles to pass given that the player must start from the first tile in $O(n)$ time.

   (b) Give an algorithm for finding the overall best sequence of tiles to pass in time $O(n^2)$.

   (c) Give a divide and conquer $O(n \log n)$ algorithm for this problem.
2. Sadat’s board game becomes an instant hit among computer scientists, with release day lines stretching blocks upon blocks. You and the others waiting excitedly in line really would like a good view of the store, but there’s someone taller than you blocking your sight. So, each of the $n$ people in line wants to pin the blame on the first person in front of them who is taller than they are. If no such person exists, then you each still want to be angry at someone, so you arbitrarily decide to be angry at the first person in line.

In order to find the position in line of who each person should blame, we can try the naive $\Theta(n^2)$ solution of:

For each person $p$ in line, step forward from their location in line until we either encounter someone taller than them or we reach the first person in line. Denote them as person $k$. Determine that $p$ blames $k$.

Design a more efficient algorithm such that everyone in line will know who to pin the blame on in $\Theta(n)$ time. Give a brief argument to prove that your algorithm is correct and derive its running time.

3. After letting Shaiks and Ladders destroy their friendship, Joseph, David and Carolina decide they no longer want to live together. However, they still want to live close by, so they decide to find houses in the same neighborhood. In fact, they are committed to living as close together as possible within separate houses. They want to develop an efficient algorithm that, given a set of houses in a neighborhood, $N = \{h_1, h_2, \ldots, h_n\}$, $n \geq 3$, with each house $h_i$ having location $(x_i, y_i)$, finds three houses $\{h_q, h_p, h_r\} \subseteq N$ such that

\[ d(h_p, h_q) + d(h_q, h_r) + d(h_p, h_r) \]

is minimized,

where, for any two houses $h_a$ and $h_b$, $d(h_a, h_b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$.

4. Many Indians, including me, are obsessed with food. When Indians living in the U.S. travel, especially to India, they want to maximize the amount of food that they can bring back while meeting the airline weight restrictions (sometimes, even violating that). In my case, I like to pack as many khakhra\(^1\) as possible when returning to the U.S. from India. During my last trip, after packing all my luggage, most of it with khakhra packets, as we were driving to the airport, we passed by a shop and my friend, who was driving me, asked me if I had tried a particular kind of khakhra, called dosa khakhra, from that shop (the shop supposedly is famous for this variety of khakhras). I had not tried them and being the greedy person that I am, especially, when it comes to khakhras, I decided to buy a few packets of dosa khakhras from that shop. I knew that I had to be quick as we were already late, and I wanted to get as many packets, while making sure that I don’t violate the weight constraint. As it is with many items in India, the packets did not specify the weight. I could easily have taken a packet to sample and left, but greed took over and I was carrying several packets. Let $k$ be a positive integer that denotes the weight of the khakhra packets in kilograms. My luggage was packed almost to the capacity, so I knew that there could be issue with the weight, so I wanted to know the exact value of $k$. The

\[^1\]https://en.wikipedia.org/wiki/Khakhra
shopkeeper was puzzled as to why I was so fussy about finding the weight of the khakhra. It was a busy time at his shop, so all he could help me with is give me a balance beam (a scale that has two pans, on one you put the item to be weighed and in the other pan you put the weights) and seemingly unlimited supply of one kilogram masses. Being in a hurry, I wanted to find $k$ by using the balance only a few times. My Theory skills came in handy and I was able to determine $k$ by using the scale at most $O(\log k)$ times. How did I do it? Justify your answer.

5. Consider the hat-check problem in which $n$ customers give a hat to a hat-check person, called JJ, at a restaurant. JJ being new at his job does not keep the hats in an orderly fashion and when the customers come to collect their hats he does not know which hat belongs to which customer. It is also not possible for him to tell the relative sizes of heads of two customers or the relative sizes of the hats, which means that he cannot sort the hats nor can he sort the customers based on the sizes of their heads. The only test that JJ has is to have the customers try on a hat, and tell him whether the hat is too big, too small, or fits perfectly. All heads are of different sizes, so all hats are different in size. You may assume that each test takes unit time. In this problem, we want to design an efficient algorithm that JJ can use to return hats to the customers.

(a) Consider the following algorithm to give the hats back to the customers. Let $C$ denote the group of customers and $H$ be the set of hats.

Hat-Check($C$, $H$)
1. if $|H| = |C| = 1$
2.   give the hat to the customer
3. else
4.   Pick a hat $h$ at random from $H$
5.   for each customer $c$ in $C$
6.     Customer $c$ tries on hat $h$
7.     if it fits then
8.       Give hat $h$ to customer $c$
9.       $C' = C - \{c\}$
10.      $H' = H - \{h\}$
11.     break out of the for loop
12.    Hat-Check($C'$, $H'$)

What is the expected running time of this algorithm? What is its worst case running time? Justify your answer.

(b) Design an algorithm with a better bound than the one in part (a). Be sure to explain your algorithm well, analyze its expected running time, and analyze its worst case running time.

(c) Suppose the following: each time JJ gives a hat to a customer, the customer either says that the hat fit or it didn’t, without giving any information on whether it is too small or too big. Does your algorithm still work? Does the algorithm in part (a) still work?
6. To salvage his friendship with David, Sadat wishes to gift David with one of his many properties. He asks David to value each of his properties, to gauge his preferences. Let the true values of Sadat’s \( n \) properties be \( x_1, x_2, \ldots, x_n \), and let David’s valuations of the corresponding properties be \( w_1, w_2, \ldots, w_n \) such that \( \sum_{i=1}^{n} w_i = W \). Sadat wishes to give David the property \( x_k \) satisfying

\[
\sum_{x_i < x_k} w_i < \frac{W}{2} \quad \text{and} \quad \sum_{x_i > x_k} w_i \leq \frac{W}{2}
\]

For example, if the true values of Sadat’s properties are $30,000, $5,000, $80,000, $170,000, $110,000, $200,000, $140,000, and David believes they are worth $5,000,000, $1,500,000, $1,500,000, $1,200,000, $1,600,000, $2,100,000, $500,000, respectively, then Sadat selects the property worth $80,000 to give to David.

a. Give an algorithm to compute the given property of Sadat’s \( n \) properties in \( O(n \log n) \) worst-case time.

b. Give an algorithm to compute the given property in \( O(n) \) worst-case time.