International Institute of Information Technology, Bangalore
CS 511 Algorithms: Practice Problems 1
2, August 2023.

1. A binary string is called *dense* if the number of 1's in the string is more than the number of 0's. For example 1, 101,110101 are *dense*, but 10, 1001,100001 are not *dense*.

   Given a binary string of length n, design an $O(n \log n)$ time algorithm to compute the number of *dense* sub-strings of the given string. For example, given 10101, the answer is 6.

2. Given a binary string of length n, design a linear time algorithm to compute the length of the largest *dense* sub-string of the given string.

3. Given a binary string of length n, design a linear time algorithm to compute the length of the largest sub-string which contains equal number of 0's and 1's.

4. Given a binary string S of length n, design a linear time algorithm to compute k, such that the number of 0's in S[0..k] is equal to number of 1's in S[k+1..n-1].

5. Write a linear time iterative algorithm to reverse a linked list.

6. Write a linear time algorithm to decide if a linked list contains a cycle or not.

7. Given a linked list containing a cycle, write a linear time algorithm to delete the cycle.

8. Design a linear time algorithm to decide if a given sequence of numbers is a stack sequence.

9. You are given an array of integers, there is a sliding window of size at most k which is moving from left to right. You can only see at most k numbers in the window. Each time the sliding window moves right by one position. Design an linear time algorithm to compute the maximum in each window.

10. Given a array A of numbers , write a linear time algorithm to compute array B, such that $B[i] = j, j$ is the smallest $j > i$ such that $A[j] < A[i]. B[i] = n$ if all the numbers to the right of $A[i]$ are greater than or equal to $A[i]$.

11. Given a array A of numbers , write a linear time algorithm to compute array B, such that $B[i] = j, j$ is the largest $j < i$ such that $A[j] > A[i]. B[i] = -1$ if all the numbers to the left of $A[i]$ are less than or equal to $A[i]$.

12. Given a array A of numbers , write an $O(n \log n)$ time algorithm to compute array B, such that $B[i] = j, j$ is the smallest j such that $A[j] < A[i]. B[i] = -1$ if all the numbers to the left of $A[i]$ are greater than or equal to $A[i]$.

13. Given a array A of numbers , write an $O(n \log n)$ linear time algorithm to compute array B, such that $B[i] = j, j$ is the largest j such that $A[j] > A[i].B[i] = n$ if all the numbers to the right of $A[i]$ are less than or equal to $A[i]$.

14. Given a sequence of n numbers, representing the stock prices of a stock on different days, design a linear time algorithm to compute the maximum profit that you can make by buying and selling a stock exactly once, you can sell a stock only after you buy it.

15. Given a sequence of n numbers, representing the stock prices of a stock on different days, design a linear time algorithm to compute the maximum profit that you can make by buying and selling a stock exactly once, you can sell a stock exactly k days after you bought it.

16. Given a sequence of n numbers, representing the stock prices of a stock on different days, design a linear time algorithm to compute the maximum profit that you can make by buying and selling a stock exactly once, you can sell a stock at least k days after you bought it.

17. Given a sequence of n numbers, representing the stock prices of a stock on different days, design a linear time algorithm to compute the maximum profit that you can make by buying and selling a stock exactly once, you can sell a stock at most k days after you bought it.

18. Given a sequence of n numbers design a linear time algorithm to compute the length of the maximum sum sub array.

19. Given a sequence of n numbers and an integer k, design a linear time algorithm to compute the length of the maximum sum sub array , whos length is exactly k.

20. Given a sequence of n numbers and an integer k, design a linear time algorithm to compute the length of the maximum sum sub array , whos length is at least k.

21. Given a sequence of n numbers and an integer k, design a linear time algorithm to compute the length of the maximum sum sub array , whos length is at most k.

22. Let $F(0) = 0, F(1) = 1$ and $F(n) = (F(n-1) + F(n-2))\%m$. If $n < 10^{18}$ and $m < 10^5$, write an efficient algorithm to compute $F(n)$.

23. Let $F(0) = 0, F(1) = 1$ and $F(n) = (F(n-1) + F(n-2))\%m$. If $n < 10^{10000}$ and $m < 10^5$, write an efficient algorithm to compute $F(n)$.

24. Let $F(0) = 0, F(1) = 1, F(2) = 2$ and $F(n) = (F(n-1)+F(n-2)+F(n-3)+1)\%m$. If $n < 10^{10000}$ and $m < 10^5$, write an efficient algorithm to compute $F(n)$.

25. Let $F(0) = 0, F(1) = 1, F(2) = 2$ and $F(n) = (2F(n-1) - 3F(n-3))\%m$. If $n < 10^{10000}$ and $m < 10^5$, write an efficient algorithm to compute $F(n)$.