International Institute of Information Technology, Bangalore
CSE 511 Algorithms: Practice Problems 1
18, August 2025.

1. 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 $F(n)$.

2. 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 $F(n)$.

3. 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 $F(n)$.

4. 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 $F(n)$.

5. If $T(n) = \Theta(1)$, for $n < 5$, write the solutions to the following recursions, by Masters Theorem.
   (a) $T(n) = 4T(n/2) + n^2$,
   (b) $T(n) = 16T(n/2) + n$,
   (c) $T(n) = 3T(n/3) + n \log n$
   (d) $T(n) = 2T(n/4) + \log n$
   (e) $T(n) = 4T(n/2) + n/\log n$
   (f) $T(n) = 9T(n/3) + n$
   (g) $T(n) = 3T(n/3) + n^2$
   (h) $T(n) = 2T(n/4) + n^{2/3}$
   (i) $T(n) = 3T(n/9) + n^{3/4}$
   (j) $T(n) = 8T(n/3) + n^2$
   (k) $T(n) = 3T(n/4) + n \log n$
   (l) $T(n) = 6T(n/3) + n^2 \log n$

6. What is the complexity of the following algorithms?

(a) $while(n > 0)\{$
$for(i = 1; i < n; i = i * 2)c++;$

$n = n/2;\}$

(b) $while(n > 0)\{$
$for(i = 1; i < n; i++)c++;$

$n = n/2;\}$

(c) $j = 1;$
$while(j < n)\{$
$for(i = 1; i < n; ++i)c++;$

$j = 2 * j;\}$

(d) $while(n > 0)\{$
$for(i = 1; i < n; i = i * 3)c++;$

$n = n/3;\}$

(e) $while(n > 0)\{$
$for(i = 1; i < n; i++)c++;$

$n = n/3;\}$

(f) $j = 1;$
$while(j < n)\{$
$for(i = 1; i < n; ++i)c++;$

$j = 3 * j;\}$

7. Solution to which of the following recursion is linear ?

(a) $T(n) = 3T(n/5) + T(n/4) + n$

(b) $T(n) = 3T(n/9) + 8T(n/11) + n$

(c) $T(n) = 3T(n/10) + 8T(n/8) + n$

(d) $T(n) = 3T(n/7) + 4T(n/8) + n$
(e) $T(n) = 2T(n/5) + 4T(n/7) + n$
(f) $T(n) = 3T(n/3) + 2T(n/4) + n$
(g) If $n = 3m, T(n) = n + 5/n \sum_{k=0}^{k=m-1} T(3k)$
(h) $T(n) = n + 49/n \sum_{k=0}^{k=n/5} T(k)$
(i) $T(n) = n + 15/n \sum_{k=0}^{k=n/3} T(k)$

8. 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.

9. 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.

10. 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.

11. 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].

12. 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.

13. 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.

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 at least k days after you bought 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 at most k days after you bought it.

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

17. 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.

18. 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.

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 at most k.

20. Given an array of sorted integers and an integer $X > 0$, design a linear time algorithm to count the number of pair elements in the array such that $A[j] - A[i] > X$.

21. Given an array of integers , design an efficient algorithm to decide if there is i, j, k, l such that $A[i] - 2A[j] = A[k] - 3A[l]$.

22. Given n, radius of a circle with $(0, 0)$ as center, write a linear time algorithm to compute the number of lattice (integer) points inside the circle.

23. Given a stream of n (about 10^9) numbers, design an $O(n)$ time and $O(k)$ space algorithm to find an element of rank k.

24. Given a sequence of n numbers and an integer $k < n$, design a linear time algorithm to find k numbers, closest to the median.

25. Given two sorted arrays of size m and n respectively and an integer k, design an $O(\log k)$ algorithm to find an element of rank k in the merged array.