Q2.

1- log log n. In recurrence tree, the cost of each level is 1 and the height of the tree is log log n.

2- (log n) (log log n). In recurrence tree, the cost of each level is log n and the height of the tree is log log n.

3- n (log^2 n). Master Theorem

Q3.

3.1 The function f(x) = x^2 is monotonically increasing, so one can just binary search. At every step, we maintain a contiguous interval of integer that contains the square root of n if it exists, and we split the interval into half based on comparing the square of the middle element with n.

3.2 The point to note is that the above idea for checking perfect squares works for checking if n is any b^th power, f or all b > 1. The crucial point is that we only need to check for values of b < log n, since for larger b, 2^b > n. So, co mpute the list of all the 'relevant' choice of b's and then for each run an algorithm similar to the above perfect square detection algorithm.

3.3 log n

3.4 The running time should be something of the form log^c n for some constant c.

Q4.https://en.wikipedia.org/wiki/Matrix_chain_multiplication or any standard text book on algorithms

Q5. https://en.wikipedia.org/wiki/3SUM