1

A balanced tree means the height of the left and right branches differ by no more than 1.

2

The search time of balanced binary tree is O(log(n)). When it compares the targeted value with the value of the current tree node in each level, it eliminates half of the remaining branches. The height of the tree is log(n) so in worst-case it takes O(log(n)) time to reach the lowest level.

3

A binary search tree may have an O(n) worst-case search time. When the value of the root is the greatest or the smallest of all nodes, the tree only has one branch. The height of the tree is equal to the number of nodes. Therefore, the worst-case search time is O(n).

4

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), a = 1, b = 2, f(n) = 1,$$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

5

$$T(n) = T\left(\frac{n}{2}\right) + c = \left(T\left(\frac{n}{4}\right) + c\right) + c$$

$$= T\left(\frac{n}{4}\right) + 2c = \left(T\left(\frac{n}{8}\right) + c\right) + 2c$$

$$= T\left(\frac{n}{8}\right) + 3c = \ldots\ldots\ldots = T\left(\frac{n}{2^k}\right) + kc$$

$$when\ n = 2^k, k = log(n), T\left(\frac{n}{2^k}\right) = T(1)\ is\ a\ constant.$$

$$T(n) = T(1) + clog(n) = clog(n) + c'$$

$$Therefore, \qquad T(n) = O(log(n))$$

$$T(n) = aT\left(\frac{n}{b}\right) + O\left(n^d\right),$$

$$T(n) = O\left(n^d\right) \qquad if\ d > log_b(a),$$

$$= O\left(n^d log(n)\right) \quad if\ d = log_b(a),$$

$$= O\left(n^{log_b(a)}\right) \quad if\ d < log_b(a)$$

$$a = 1, b = 2, d = 0, \qquad log_b(a) = 0 = d$$

$$Therefore, \qquad T(n) = O\left(n^d log(n)\right) = O(log(n))$$