1:

recursive:

```
def Fib(arg) :
  if (arg == 0 or arg == 1) :
    return arg                              -constant time(1)
  else :
    return Fib(arg - 1) + Fib(arg - 2)      -O(2^(n-1)) + O(2^(n-2))
```

complexity: $O(2^{n-1}) + O(2^{n-2}) + O(1) = O(2^n)$

memosation:

```
memo = {}
def Fib(arg) :
    if (arg == 0 or arg == 1) :
        return arg                                  -constant time(1)
    if arg in memo :
        return memo[arg]                            -constant time(1)
    else :
        memo[arg] = Fib(arg - 1) + Fib(arg - 2)     -O(n)
        return memo[arg]
```

complexity: $O(n) + O(1) + O(1) = O(n)$

Because the recursive method call the function two times for each argument, but the memosation method only call the function once because the previous result is stored in the memory.

2: The storing time will change. Still O(n).

3.1: a: M, N, J, K, L

b: A

c: A

d: F, G, H

e: A, B

f: I, M, N

g: F, G, H

h: left: J, right: K

i: 1

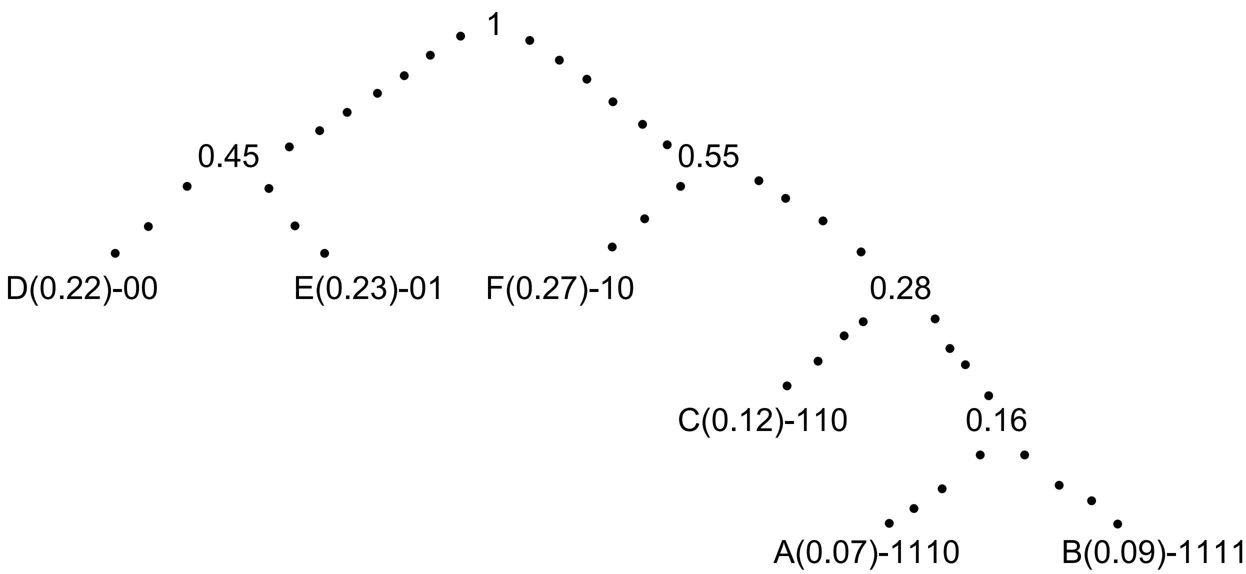
j: 2

3.2: ABEI, ACGJ, ACGK, ACHL, BEIM, BEIN. Therefore, 5 paths.

3.6:

	preorder(n) < preorder(m)	inorder(n) < inorder(m)	postorder(n) < postorder(m)
n is to the left of m	✓	✓	✓
n is to the right of m	✓		✓
n is a proper ancestor of m	✓	✓	
n is a proper descendant of m		✓	✓

3.20:



So: 0.07 * 4 + 0.09 * 4 + 0.12 * 3 + 0.22 *2 +0.23 *2 + 0.27 *2 = 2.44

3.21: depth(a) > depth(b),

So length(a) > length(b) and probability(b) = probability(a) + probability(others).

Therefore, probability(a) ≤ probability(b).