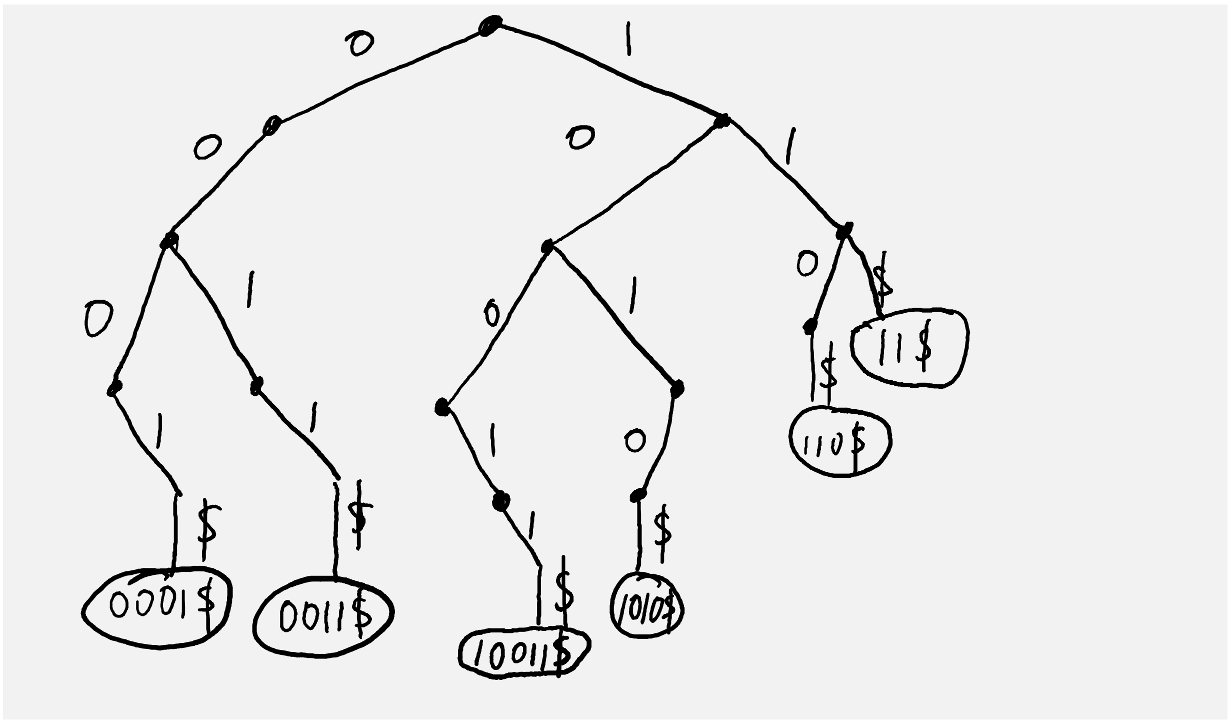
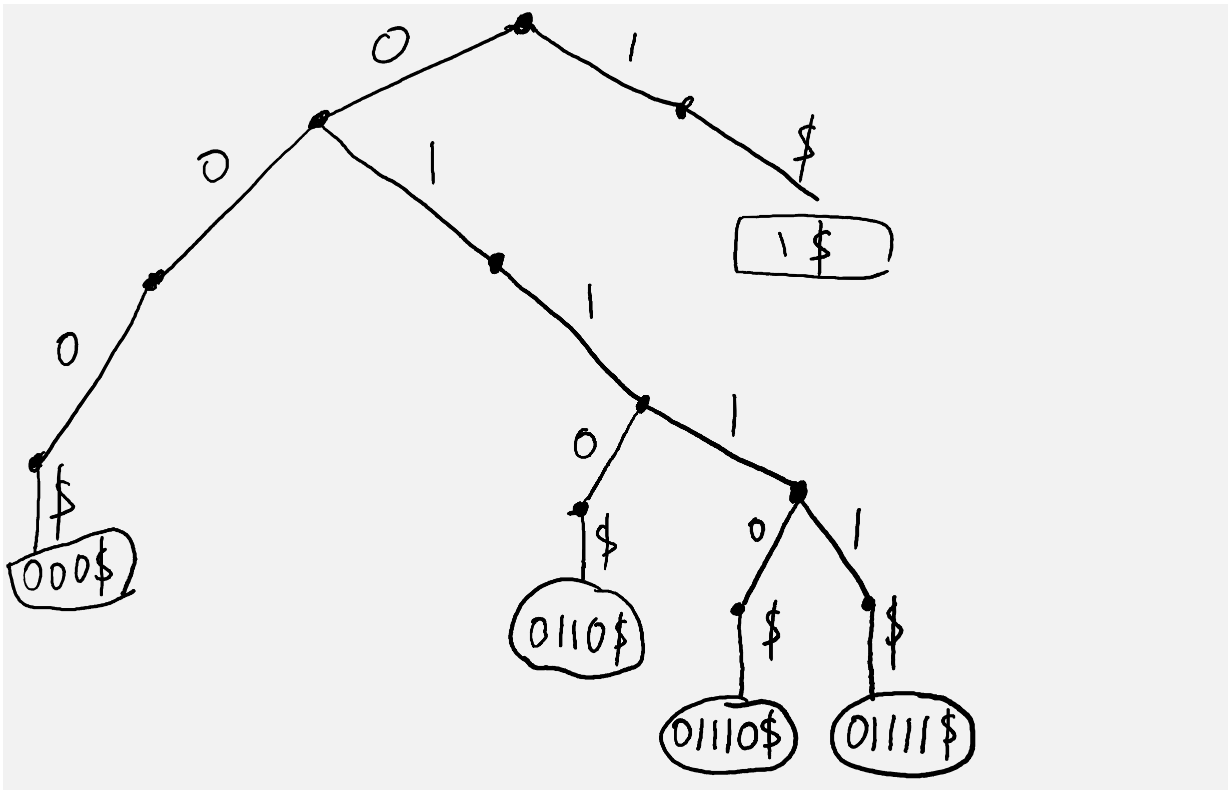
Q6:

a)


0
1
0
0
1
0
1
0
1
0
$
1
1
1
0
$
$
$
1
$
0001$
0011$
10011$
1010$
110$
11$

b)


0
1
0
1
$
1$
0
1
1
$
0
000$
0
1
$
0110$
$
$
01110$
01111$

c)

The algorithm first search the position of x, then recursively add strings, which are (l – x.length) or less levels below that position, to a list. Lastly, return the list.

Searching x takes time $O(m)$ where m is the length of x;

Add string to lists takes time $O(2^{l-m+1})$ where m is the length of x, l is the given length.

```
//returns a pointer to the node containing x(without $)
Node* search(T, x){
      p : pointer to a node
      p = T.root;
      for (auto& one_char: x)
            if (one_char == 0){
                  p = p.left;
            } else if (one_char == 1){
                  p = p.right;
            }
      }
      return p;
}

// add strings starting from pointer p with length less than or equal
// to "length" to my_list
void add_strings(old_str, my_list, p, length){
old_str: string containing previous result
my_list: list of strings as result
p : pointer to a node
length: string length

      if (length >= 0){
            if (p.is_end()){
                  my_list.add(old_str + p.value);
            }
            add_strings(old_str + p.value, my_list, p.left, length - 1);
            add_strings(old_str + p.value, my_list, p.right, length - 1);
      }
}

List<string> Look(T, x, l){
T: Triee
x: given string
l: given maximum length

      p : pointer to a node
      p = search(T, x);
      length = l - x.length();
      my_list : list of strings, initially empty
      add_strings(x, my_list, p, length);
      return my_list;
}
```