Algorithm Design and Analysis

吴长亮 201828013229136

Assignment 2

1 Question 1

1.1 Algorithm Description

$$OPT(i) = max \begin{cases} OPT(i-1) \\ OPT(i-2) + V_i \quad \text{Select No.i} \end{cases}$$

1.2 Pseudo-code

Algorithm 1 Money robbing case 1

```
function DP(Array, n)
    dp[0] = Array[0]
    dp[1] = max(Array[0], Array[1])
    for i= 2 to n do
        dp[i] = max(dp[i − 2] + Array[i], dp[i − 1])
    end for
    return result
end function
```

Algorithm 2 Money robbing case 2 (Loop)

```
function DP(Array, n)
    //Select No.1
    dp[0] = Array[0]
    dp[1] = Array[0]
    for i = 2 to n do
        dp[i] = max(dp[i − 2] + Array[i], dp[i − 1])
    end for
    MaxValue = dp[size − 2]
```

```
    //Do not select No.1
    dp[0] = 0
    dp[1] = Array[1]
    for i = 2 to n do
        dp[i] = max(dp[i − 2] + Array[i], dp[i − 1])
    end for
    MaxValue = max(MaxValue, dp[size − 1])
    return result
end function
```

1.3 Complexity

case 1:

$$O(n)$$

case 2:

$$O(n)$$

2 Question 2

2.1 Algorithm Description

Use DFS search the tree

i-the present node

k-the pre node

[0]-do not select the node

[1]-select the node

$$OPT[i][0] = OPT[i][0] + max(OPT[k][0], OPT[k][1])$$

$$OPT[i][1] = OPT[i][1] + OPT[k][0]$$

2.2 Pseudo-code

Algorithm 3 Node selection

```
function DFS(x, pre)
    for node linked to x except pre do
        DFS(t, x)
        dp[x][0]+ = max(dp[t][1], dp[t][0])
        dp[x][1]+ = dp[t][0]
    end for
    return 0
end function
function DP
```

```
    DFS(1, 0)
    return max(dp[1][0], dp[1][1])
end function
```

2.3 Complexity

$$O(n)$$

3 Question 3

3.1 Algorithm Description

$$OPT(i) = \begin{cases} OPT(i-2) & \text{only double-digit} \\ OPT(i-1) & \text{only single-digit} \\ OPT(i-1) + OPT(i-2) & \text{single-digit or double-digit} \end{cases}$$

3.2 Pseudo-code

Algorithm 4 Decoding

```
function DP
    dp[0] = 1
    dp[1] = 1(single − digit)/2(double − digit)
    for i = 2 to n do
        if V_i = 0 then
            dp[i] = dp[i − 2]
        else if 10 < V_i and V_i < 27 then
            dp[i] = dp[i − 1]
        else
            dp[i] = dp[i − 1] + dp[i − 2]
        end if
    end for
    return dp[n − 1]
end function
```

3.3 Complexity

$$O(n)$$