一：(1)

$$
dp[i][j] = \begin{cases} v_n * \lfloor \frac{j}{w_n} \rfloor \ , & i = n, \\ dp[i+1][j] \ , & 1 \le j < w_i, \ 1 \le i \le n, \\ \max\left(dp[i+1][j], dp[i][j-w_i] + v_i\right) \ , & j \ge w_i, \ 1 \le i \le n, \end{cases}
$$

(2)

$$
\begin{bmatrix} 0 & 2 & 4 & 6 & 8 & 10 \\ 0 & 0 & 4 & 4 & 8 & 8 \\ 0 & 0 & 0 & 4 & 5 & 5 \\ 0 & 0 & 0 & 0 & 5 & 5 \end{bmatrix}
$$

(3)

(4)

Algorithm 1 max_value

Input: the weight array w, the value array v, the capacity c, the amount of items n

Output: the max value

$dp[1...n][0...c]$
for $j = 0$ to c **do**
  $dp[n][j] = v_n * (j//w_n)$
end for
for $i = n - 1$ to 1 **do**
  for $j = 0$ to c **do**
    if $j < w_i$ **then**
      $dp[i][j] = dp[i+1][j]$
    else
      $dp[i][j] = max(dp[i+1][j], dp[i][j-w_i] + v_i)$
    end if
  end for
end for
return $dp[1][c]$

将二维数组降维至一维数组，伪代码如下：

Algorithm 2 max_value2

Input: the weight array w, the value array v, the capacity c, the amount of items n

Output: the max value

```
dp[0...c]
for i = 1 to n do
    for j = w[i] to c do
        dp[j] = max(dp[j], dp[j - w[i]] + v[i])
    end for
end for
return dp[c]
```

二.

设 $E[i][j]$ 为 $\{k_i, \cdots, k_j\}$ 的优化解的期望搜索代价，

递推方程为

$$E[i][j] = \begin{cases} q_{i-1}, & j = i-1 \\ \min\limits_{i \le r \le j} (E[i][r-1] + E[r+1][j] + W[i][j]), & j \ge i \end{cases}$$

其中

$$\begin{aligned} W[i][j] = &\sum_{m=i}^{r-1} p_m + \sum_{m=i-1}^{r-1} q_m \\ &+ \sum_{m=i}^{j} p_m + \sum_{m=i-1}^{j} q_m + p_r \end{aligned}$$

计算可得最优二叉搜索树代价为 3.12，结构为：

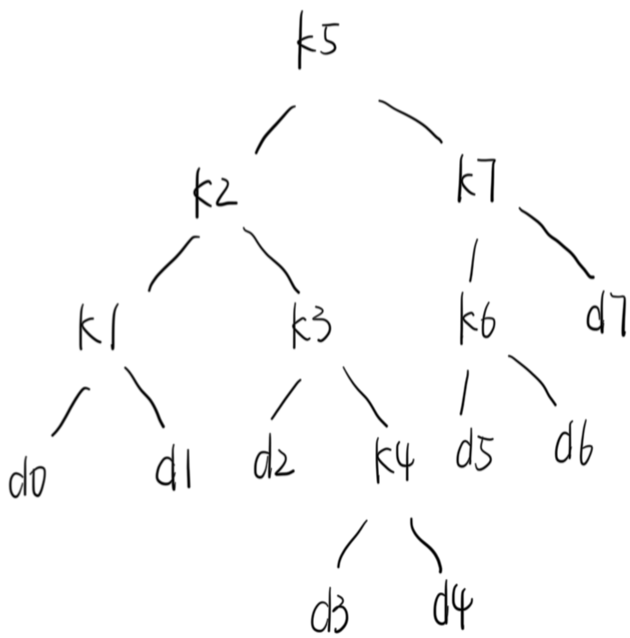

Figure 1: 2.png

三：

对于不首尾相连的情况，当前的第 i 个洞是否选取只取决于前面 $i-1$ 个洞是否选取，比较第 i 个洞选取的收益和第 $i-1$ 个洞选取的收益并选择其中最优的即可。设 $m[i]$ 为前 i 个洞的最优解，则

状态转移方程：

$$m[i] = \begin{cases} nums[0], & i = 0 \\ max(nums[0], nums[1]), & i = 1 \\ max(m[i-1], m[i-2] + nums[i]), & 2 \le i \le n \end{cases}$$

对于首尾相连的情况，将 n 个洞分为 0 $n-2$ 和 1 $n-1$ 两组，即分别去掉最后一个洞和第一个洞，最后在两种情况所求得的最优解中取最大值即可。