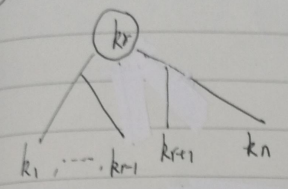
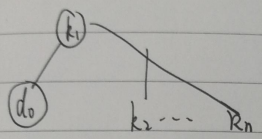
Exercise 1.

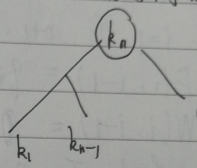
(a) K = {k1, k2, ... kn} 的优化解的根必为 K 中某个 kr



① 若 r=1，左子树仅包含 d0.



② 若 r=n，右子树仅包含 dn.



如果优化二叉搜索树 T 具有包含关键字集合 {ki, ..., kj} 的子树 T'，则 T' 是关于关键字集合 {ki, ki+1, ... kj} 的子问题的优化解.

故对其子问题：

Kij = {ki, ..., kj} 的优分解 Tij 的根必为 K 中某个 kr'，结构同上课.

只要对于每个 kr ∈ K，确定 {ki, ..., kr-1} 和 {kr+1, ..., kj} 的优化解，我们就可以求出 K 的优化解.

(b) 由上述结构分析可知

E(i,j) = pr + E(i, r-1) + W(i, r-1) + E(r+1, j) + W(r+1, j)

(其中 W 为合并子树时深度加 1 所造成的代价)

且 W(ij) = pr + W(i, r-1) + W(r+1, j)

故有递归方程

W(i, i-1) = qi-1

W(i,j) = W(i,r-1) + pr + W(r+1, j) = W(i,j-1) + pj + qj

$E(i, i-1) = q_{i-1}$.

$E(i,j) = E(i, r-1) + E(r+1, j) + W(i,j)$

(c1) 数据结构：

数组 E：存储优化前的搜索代价

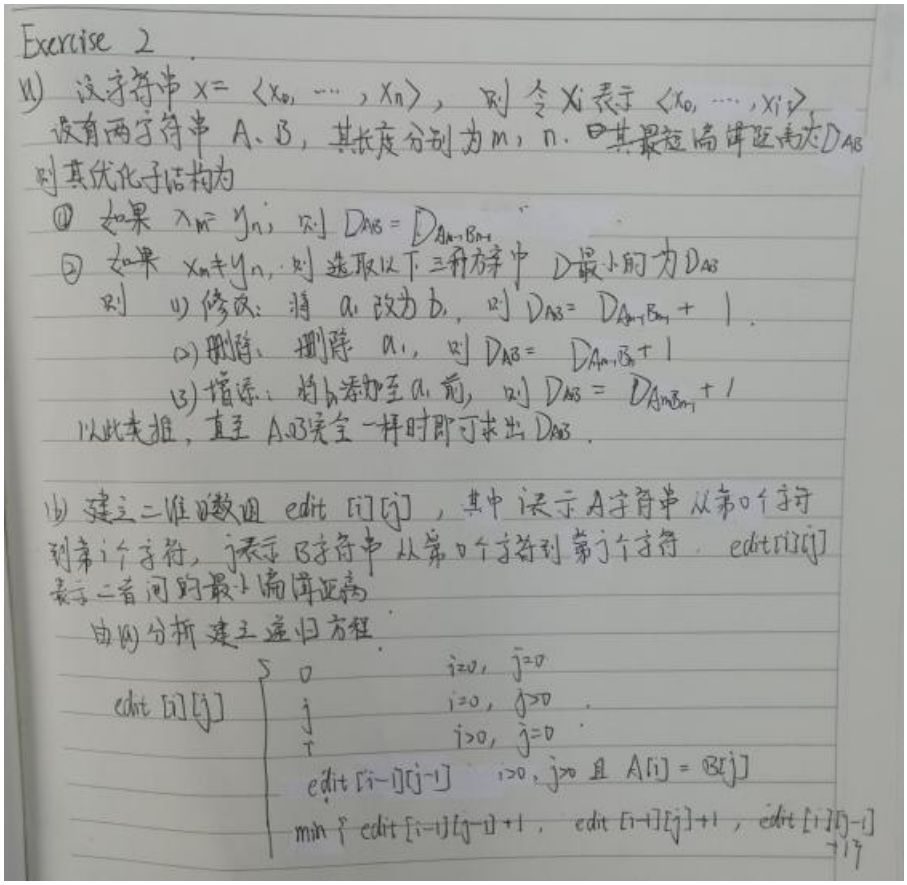
数组 W：存储代价增量

数组 Root：记录子问题优化前的根

```
Optimal-BST (p, q, n)
For i=1 To n+1 Do
    E(i, i-1) = q_{i-1};
    W(i, i-1) = q_{i-1};
For l=1 To n Do
    For i=1 To n-l+1 Do
        j = i+l-1;
        E(i, j) = ∞;
        W(i, j) = W(i, j-1) + q_j + p_j;
        For r=i To j Do
            t = E(i, r-1) + E(r+1, j) + W(i, j)
            If t < E(i, j)
            Then E(i, j) = t;
                 Root(i, j) = r;
Return E and Root.
```

(c2) 时间复杂度：三层循环，时间复杂度为 $O(n^3)$

空间复杂度：二个 $(n+1)\times(n+1)$ 数组，一个 $n\times n$ 数组，空间复杂度为 $O(n^2)$

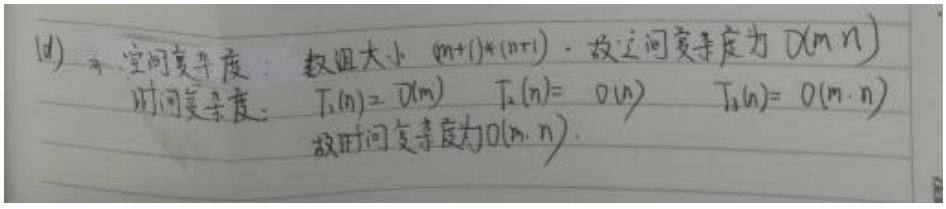


（c）代码实现：

```
import java.util.Scanner;
public class edit {
public static void main(String[] args) {
    int i , j;
    Scanner sc = new Scanner(System.in);
    System.out.println("Input:");
    String A = sc.next();
    String B = sc.next();
    int[][] edit = new int[A.length()+1][B.length()+1];
    for(j = 0 ; j <= B.length() ; j++)
    {
        edit[0][j] = j;
    }
    for(i = 1; i <= A.length() ; i++)
    {
        edit[i][0] = i;
    }
    for( i = 0; i < A.length() ; i++)
    {
```

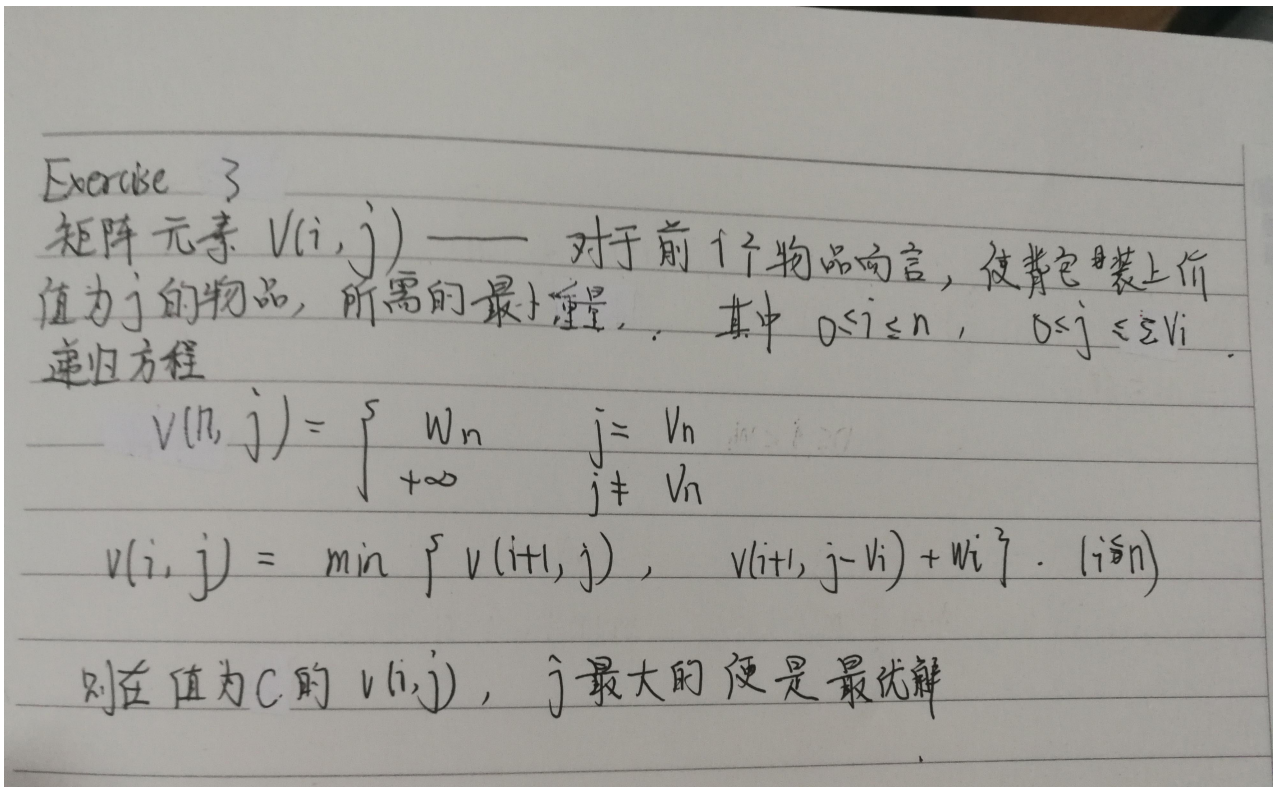
```
            int m = i + 1;
            for( j = 0; j < B.length(); j++)
            {

                int n = j + 1;
                if(A.charAt(i) == B.charAt(j))
                {
                    edit[m][n] = edit[m-1][n-1];
                }
                else
                {
                    edit[m][n] = edit[m-1][n-1] + 1;
                    if((edit[m][n-1]+1) < edit[m][n])
                    {
                        edit[m][n] = edit[m][n-1] + 1;
                    }
                    if((edit[m-1][n]+1) < edit[m][n])
                    {
                        edit[m][n] = edit[m-1][n] + 1;
                    }
                }
            }
        }
        System.out.println("Output:");
        System.out.println(edit[A.length()][B.length()]);
}
}
```

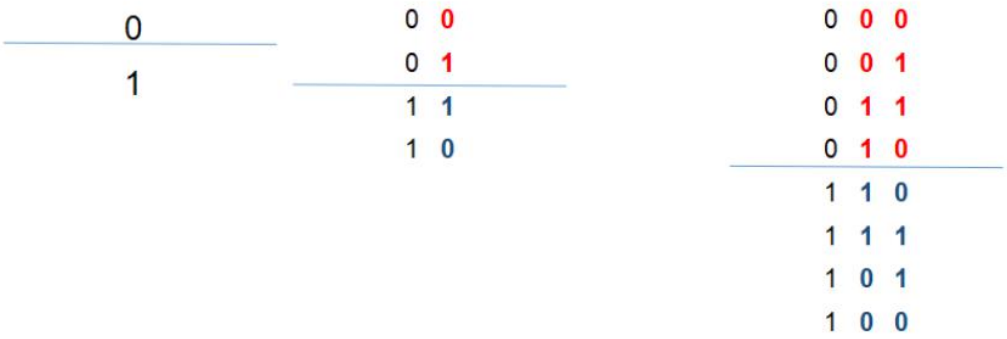


结果：

```
Input:
horse
ros
Output:
3
```



Exercise 4
思路：



如图，由格雷码的性质，格雷码中间切割后，取出第一位数字，前半部分与后半部分对称，而前半部分第一位数字为 0，后半部分第一位数字为 1。由此联想

到利用分治算法（不是代价类函数不好用动态规划算法？）表示格雷码。

代码实现：

```
import java.lang.Math;
import java.util.Scanner;


public class GrayCode {
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int i;
        System.out.println("Input:");
        int n = sc.nextInt();
        int[] a = divide(n);
        System.out.println("Output:");
        System.out.print("[ ");
        for(i = 0; i < a.length ; i++)
        {
            System.out.print(" " + a[i] + " ");
        }
        System.out.print(" ]");
    }


    public static int[] divide(int n)
    {
        int i , j;
        if(n == 1)
        {
            return new int[] {0,1};
        }
        else
        {
            int[] temp = divide(n-1);
            int[] ret = new int[temp.length * 2];
            for(i = 0 ; i < temp.length ; i++)
            {
                ret[i] = temp[i];
            }
            for(i = temp.length , j = temp.length - 1 ; i < ret.length && j >=
0 ; i++ , j--)
            {
                ret[i] = temp[j] + (int)Math.pow(2, n-1);
            }
```

```
            return ret;
        }
    }
}
```

结果：

```
Input:
2
Output:
[  0  1  3  2  ]

Input:
3
Output:
[  0  1  3  2  6  7  5  4  ]
```