Lab 10

# Problem 1

S = {S0, S1, S2, S3, S4}, w = {1, 4, 6, 2, 5}, v = {4, 6, 3, 1, 4}, W = 12

* b0 = 4, b1 = 1.5, b2 = 0.5, b3 = 0.5, b4 = 0.8
* S’ = {S0, S1, S4, S2, S3} , w’ = {1, 4, 5, 6, 2}, v’ = {4, 6, 4, 3, 1}

w = 1 + 4 + 5 = 10 < 12

Solution {S0, S1, S4} is not optimal.

# Problem 2

BinarySearch is non-overlapping problem, Recursive Fibonacci is overlap problem

The sub problem of BinarySearch is not overlap because BinarySearch do the self-call in 2 distinct ranges, the first one from bottom element to middle element, the second one from middle plus 1 element to top element.

The Recursive Fibonacci is overlap because it does safe-call for same input several times.

E.g:

f(n) = f(n - 1) + f(n - 2)

f(n - 1) = f(n - 2) + f(n - 3) // f(n - 2) is already called above

# Problem 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A[i,j] | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | {} | NULL | NULL | NULL | {4} | NULL |

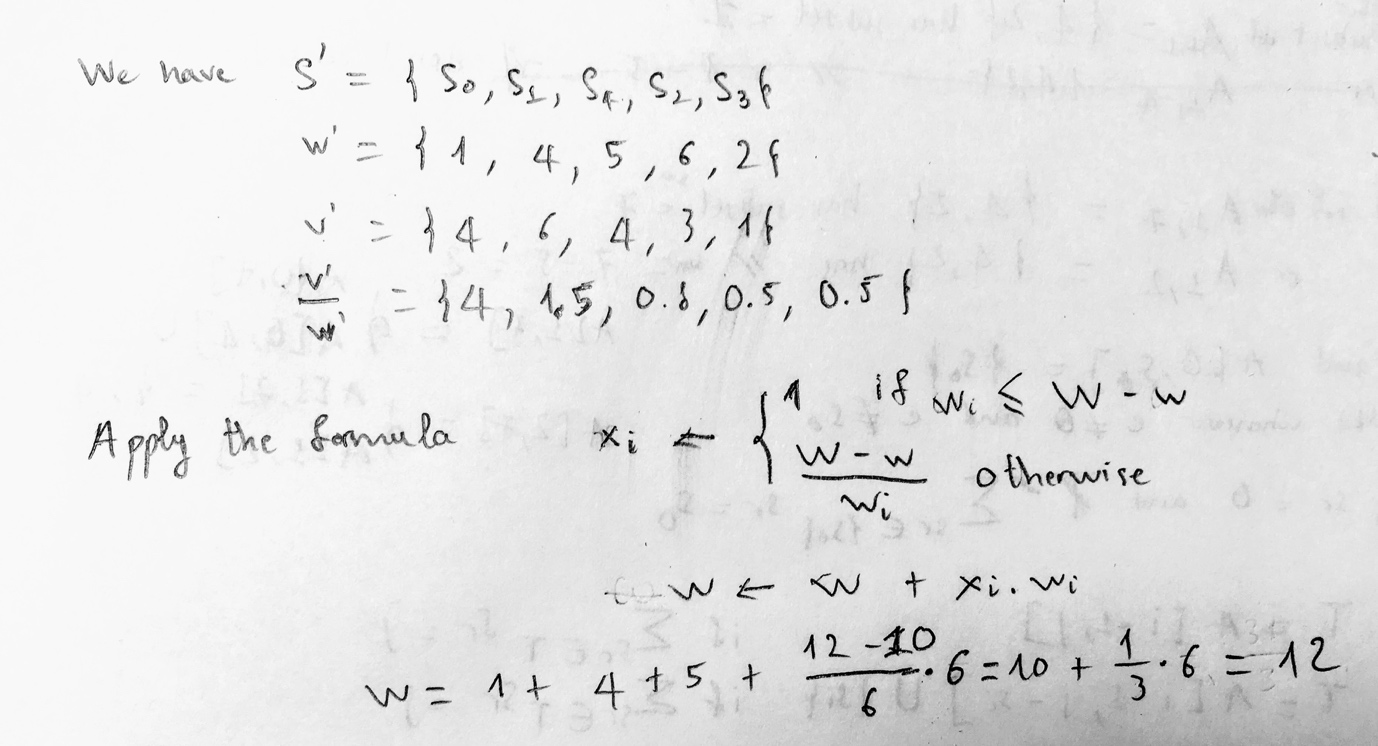
# Problem 4

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A[i,j] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0 | {} | NULL | NULL | NULL | {4} | NULL | NULL | NULL | NULL |
| 1 | {} | NULL | NULL | {3} | {4} | NULL | NULL | {4, 3} |  |
| 2 |  |  |  |  |  |  |  | {4, 3} |  |
| 3 |  |  |  |  |  |  |  |  |  |

# Problem 5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A[i,j] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | {} | {} | {} | {s0} | {s0} | {s0} | {s0} | {s0} |
| 1 | {} | {s1} | {s1} | {s0} | {s0,s1} | {s0,s1} | {s0,s1} | {s0,s1} |
| 2 | {} | {s1} | {s1} | {s0} | {s0,s1} | {s0,s1} | {s0,s2} | {s0,s1,s2} |
| 3 |  |  |  |  |  |  |  | {s0,s1,s2} |

# Problem 6



# Problem 7

package lab10;  
  
*/\*\*  
 \* Created by Bi on 8/8/17.  
 \*/*public class EditDistance {  
  
 public static int editDistance(String A, String B) {  
 int n = A.length();  
 int m = B.length();  
  
 int[][] table = new int[n + 1][m + 1];  
 table[0][0] = 0;  
  
 for (int i = 1; i <= n; i++) {  
 table[i][0] = i;  
 }  
  
 for (int j = 1; j <= m; j++) {  
 table[0][j] = j;  
 }  
  
 for (int i = 1; i <= n ; i++) {  
 for (int j = 1; j <= m ; j++) {  
 if (A.charAt(i - 1) == B.charAt(j - 1)) {  
 table[i][j] = table[i - 1][j - 1];  
 } else {  
 table[i][j] = *minOfTripple*(table[i - 1][j], table[i - 1][j - 1], table[i][j - 1]) + 1;  
 }  
 }  
 }  
  
 return table[n][m];  
 }  
  
 public static int minOfTripple(int a, int b, int c) {  
 int min = a;  
 if (b < min) {  
 min = b;  
 }  
 if (c < min) {  
 min = c;  
 }  
 return min;  
 }  
  
 public static void main(String[] args) {  
 System.*out*.println(*editDistance*("DUCK", "TUG"));  
 System.*out*.println(*editDistance*("QUY", "GUYC"));  
 }  
  
}

# Problem 8