Informatics II, Spring 2023, Solution 11

Publication of exercise: May 14, 2023
Publication of solution: May 21, 2023
Exercise classes: May 22 - 26, 2023

Task 1

a)

1	1	2	3
2	3	4	4
1	1	2	5
2	2	1	1
3	3	4	5

Solution helper table

b)

$$S[i][j] = \begin{cases} max(1, S[i-1][j]+1, S[i][j-1]+1), & \text{if } abs(M[i][j]-M[i-1][j]) <= 1 \\ & \wedge abs(M[i][j]-M[i][j-1]) <= 1 \wedge i, j >= 1 \\ max(1, S[i][j-1]+1), & \text{else if } abs(M[i][j]-M[i][j-1]) <= 1 \wedge j >= 1 \\ max(1, S[i-1][j]+1), & \text{else if } abs(M[i][j]-M[i-1][j]) <= 1 \wedge i >= 1 \\ 1, & \text{else} \end{cases}$$

c)

```
int longestPath(int x, int y, int M[x][y]) {
   int S[x][y];
   for (int i = 0; i < x; i++) {
      for (int j = 0; j < y; j++) {
         S[i][j] = -100;
      }
   }
   int max_value = 0;
   int new_entry = 0;
   for (int i = 0; i < x; i++) {
      for (int j = 0; j < y; j++) {
         if (j - 1 >= 0 && i - 1 >= 0 && abs(M[i][j] - M[i][j - 1]) &&
```

```
            abs(M[i][j] - M[i - 1][j]) <= 1) {
          new_entry = max3(1, S[i][j - 1] + 1, S[i - 1][j] + 1);
          S[i][j] = new_entry;
          if (new_entry > max_value) {
            max_value = new_entry;
          }
        } else if (j - 1 >= 0 && abs(M[i][j] - M[i][j - 1]) <= 1) {
          new_entry = max(1, S[i][j - 1] + 1);
          S[i][j] = new_entry;
          if (new_entry > max_value) {
            max_value = new_entry;
          }
        } else if (i - 1 >= 0 && abs(M[i][j] - M[i - 1][j]) <= 1) {
          new_entry = max(1, S[i - 1][j] + 1);
          S[i][j] = new_entry;
          if (new_entry > max_value) {
            max_value = new_entry;
          }
        } else {
          S[i][j] = 1;
        }
      }
    }
    // If you want to see what the helper matrix S looks like uncomment
    // print_matrix(x, y, S);
    return max_value;
}
```

Task 2

a) 5
b) 0
c) 4
d)

```
int isPalindrome(char X[], int i, int j) {
    while (i <= j) {
        if (X[i] != X[j]) {
            return false;
        }
        i++;
        j--;
    }
    return true;
}
```

e):

```
int findMinCutsRecusive(char X[], int i, int j) {
    int count = 9999;
    int minimum = 9999;
    if (i == j || isPalindrome(X, i, j) == true) {
        return 0;
    }

    for (int k = i; k < j; k++) {
        count =
            1 + findMinCutsRecusive(X, i, k) + findMinCutsRecusive(X, k + 1, j);
        if (count < minimum) {
            minimum = count;
```

```
      }
   }

   return minimum;
}
```

f):

```
int findMinCuts(char X[], int n) {
   int helper_matrix[n][n];
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         helper_matrix[i][j] = false;
      }
   }
   for (int i = n - 1; i >= 0; i--) {
      for (int j = i; j < n; j++) {
         if (i == j) {
            helper_matrix[i][j] = true;
         } else if (X[i] == X[j]) {
            if (j - i == 1) {
               helper_matrix[i][j] = true;
            } else {
               helper_matrix[i][j] = helper_matrix[i + 1][j - 1];
            }
         } else {
            helper_matrix[i][j] = false;
         }
      }
   }
   // If you want to see what the helper matrix looks like uncomment
   // print_matrix(n,n,helper_matrix);

   int helper_array[n];
   for (int i = 0; i < n; i++) {
      helper_array[i] = 9999;
   }

   for (int i = n - 1; i >= 0; i--) {
      if (helper_matrix[i][n - 1] == true) {
         helper_array[i] = 0;
      } else {
         for (int j = n - 2; j > i - 1; j--) {
            if (helper_matrix[i][j] == true) {
               helper_array[i] = min(helper_array[i], 1 + helper_array[j + 1]);
            }
         }
      }
   }
   return helper_array[0];
}
```

Task 3

a)

0	0	0	5	0	0
0	3	0	4	0	0
3	2	1	3	1	2
2	1	0	2	0	1
1	0	0	1	1	0

Solution helper table bottom

0	0	0	1	0	0
0	1	0	1	0	0
1	2	3	4	5	6
1	2	0	1	0	1
1	0	0	1	2	0

Solution helper table left

0	0	0	1	0	0
0	1	0	1	0	0
6	5	4	3	2	1
2	1	0	1	0	1
1	0	0	2	1	0

Solution helper table right

b)

```
void makeHelper(int x, int y, int M[x][y], int top[x][y], int bottom[x][y],
                int left[x][y], int right[x][y]) {
   for (int i = 0; i < x; i++) {
      for (int j = 0; j < y; j++) {
         if (M[i][j] == 1) {
            if (i - 1 >= 0) {
               top[i][j] = top[i - 1][j] + 1;
            } else {
               top[i][j] = 1;
            }
            if (j - 1 >= 0) {
               left[i][j] = left[i][j - 1] + 1;
            } else {
               left[i][j] = 1;
            }
         } else {
            top[i][j] = 0;
            left[i][j] = 0;
         }
      }
   }

   for (int i = x - 1; i >= 0; i--) {
      for (int j = y - 1; j >= 0; j--) {
         if (M[i][j] == 1) {
            if (i + 1 < x) {
               bottom[i][j] = bottom[i + 1][j] + 1;
            } else {
               bottom[i][j] = 1;
            }
            if (j + 1 < y) {
               right[i][j] = right[i][j + 1] + 1;
            } else {
               right[i][j] = 1;
            }
         } else {
            bottom[i][j] = 0;
            right[i][j] = 0;
         }
      }
   }
}
```

c)

```
int biggest_plus(int x, int y, int M[x][y]) {
   int bottom[x][y];
   int top[x][y];
   int left[x][y];
   int right[x][y];
   makeHelper(x, y, M, top, bottom, left, right);
   int biggest_value = 0;
   int new_value;
   int S[x][y];

   // Uncomment to see helper matrices top, bottom, left and right
   // print_matrix(m, n, top);
   // print_matrix(m, n, bottom);
```

```
14    // print_matrix(m, n, left);
15    // print_matrix(m, n, right);
16
17    for (int i = 0; i < x; i++) {
18       for (int j = 0; j < y; j++) {
19          new_value = min(bottom[i][j], top[i][j], left[i][j], right[i][j]);
20          if (new_value < 1) {
21             new_value = 0;
22          } else {
23             new_value = (new_value - 1) * 4 + 1;
24          }
25          S[i][j] = new_value;
26          if (new_value > biggest_value) {
27             biggest_value = new_value;
28          }
29       }
30    }
31    // Uncomment to print helper matrix S
32    // print_matrix(x, y, S);
33    return biggest_value;
34 }
```