Problem 1. Solution:Input: P[1 … n], Q [1…n]Output:The minimum length of the leash connect two frogs l.Algorithm:(1)Create a n by n matrix d where d[i, j] = |P[i], Q[j]| (the distance between P[i] and Q[j]). Both iand j range from 1 to n.(2)Create a n by n matrix L to save the minimum length of the leash between P[i] and Q[j](3)L[1, 1] = d[1, 1](4)for i⇓1 to nfor j⇓1 to nL[i, j] = min (L[i-1, j], L[i-1, j-1], L[i, j-1])L[i, j] = max (L[i, j], d[i, j])(5)l = L[n, n]Analysis:With using dynamic programming, the matrix L is filled from left to right, and from up to bottom. Thealgorithm make the calculation of L[i, j] relies on its left, up, and left-up elements which are all filled byprevious steps. So the run time of the algorithm is n^2.The algorithm is based on the following logic:The minimum length of the leash is decided by the minimum length of the leash of frogs at previousposition and the length between frog at current position. The leash must long enough to cover theprevious minimum leash and the current distance between two frogs, so theL[i, j] = max (L[i, j], d[i, j]).From previous position of frogs to current positions, there are only three choices. From these threechoices, we always want the least length leash. So theL[i, j] = min (L[i-1, j], L[i-1, j-1], L[i, j-1])With the base case L[1, 1] = d[1, 1], we can calculate the whole L matrix from left-up corner to right-bottom corner row by row. Then the value of L[n, n] is just the minimum length of leash need to connecttwo frogs at the position P[n] and Q[n].