Billy Saysavath

Duy Do

CPSC 335 – Section 3

Fall 2015

Assignment #3

# Exhaustive Optimization Algorithm:

* 1. Pseudocode:

#Code after the clock starts

Dist = farthest(n,P)

bestDist = n\*Dist

A [n]

for i in range(0, n-1):

A[i] = i;

def print\_perm( n, A[], sizeA, P, bestSet, bestDist ):

i, dist

if (n == 1):

dist = sqrt( pow( P[A[i]].x – P[A[sizeA-1]].x, 2 ) + pow( P[A[i]].y – P[A[sizeA-1]].y, 2 ))

if( disk < bestDist ):

bestDist = dist

else

for i in range (0, sizeA):

print\_perm(n – 1, A, sizeA, P, bestSet, bestDist)

if ( n % 2 == 0 ):

A[i], A[n-1] = A[n-1], A[i]

else:

A[0], A[n-1] = A[n-1], A[0]

print\_perm(n – 1, A, sizeA, P, bestSet, bestDist)

def farthest(n, P):

max\_dist=0;

i, j;

dist;

for i in range(0, n):

for j in range(0, n):

dist = (P[i].x - P[j].x)\*(P[i].x - P[j].x) + (P[i].y - P[j].y)\*(P[i].y - P[j].y)

if (max\_dist < dist):

max\_dist = dist

return sqrt(max\_dist)

* 1. Analyze:

Line 2 takes O(n2) steps. Line 3 takes n steps. Line 4 takes a constant step, so we say 1. Line 6 takes n steps. Calling print\_perm, line 10 takes 2 steps. Line 12 takes 1 step. Line 13 takes 1 + max(1, 0) step. Line 16, the for-loop will take sizeA steps, which equals to number of points m. Multiply the recursive call at line 17 which eventually becomes O(n!) since it is a permutation, and 1 + max(1, 1) for if-else statement from line 18 to 21. Lastly, we add another O(n!) for a recursive call. We have the following:

We have, where m is the number of points.