Nearest Neighbor

```
// allocate space for the Visited array of Boolean values
Visited = new bool[n];
// set it all to False
for (i = 0; i<n; i++)
        Visited[i] = false;

// calculate the starting vertex A
A = farthest_point(n, P);
// add it to the path
i = 0;
M[i] = A;

// set it as visited
Visited[A] = true;

for (i = 1; i<n; i++) {
        // calculate the nearest unvisited neighbor from node A
        B = nearest(n, P, A, Visited);
        // node B becomes the new node A
        A = B;
        // add it to the path
        M[i] = A;
        Visited[A] = true;
}

// calculate the length of the Hamiltonian cycle
dist = 0;
for (i = 0; i < n - 1; i++)
        dist += sqrt((P[M[i]].x - P[M[i + 1]].x)*(P[M[i]].x - P[M[i + 1]].x) +
        (P[M[i]].y - P[M[i + 1]].y)*(P[M[i]].y - P[M[i + 1]].y));
dist += sqrt((P[M[0]].x - P[M[n - 1]].x)*(P[M[0]].x - P[M[n - 1]].x) +
        (P[M[0]].y - P[M[n - 1]].y)*(P[M[0]].y - P[M[n - 1]].y));
```

n2 ; n ; 1 ; 1+1 ; 1 ; n+5 ; n ; 1 ; 1+1 ; 1+1 ; 1 ; $\sum_{i=0}^{n-1} 28 = 28n$; 28

```
//-----------------------------------------
int farthest_point(int n, point2D *P)
// function to calculate the furthest distance between any two 2D points
{
        float max_dist = 0, dist = 1;
        int p1, p2;
        for (int i = 0; i < n-1; i++) {
                for (int j = i + 1; j < n; j++) {
                        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);
                        dist = sqrt(dist);
                        if (dist > max_dist) {
                                max_dist = dist;
                                p1 = i; p2 = j;
                        }
                }
        }
        return p2;
}
```

Rt = $O(n^2)$; 1+1 ; 1+1 ; 6+1+6+1+6+1+6 ; $\sum_{j=i}^{n} 33 = (n-i+1)\,33$; 1+1 ; 1+max(3,0)

$\sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 33 =$

$= n^2 - \frac{(n-1)(n)}{2} + 33n = n^2 - \frac{n^2}{2} - \frac{n}{2} + 33n$

$= \frac{n^2}{2} + \frac{65}{2}n \in O(n^2)$ ✓

```
int nearest(int n, point2D *P, int A, bool *Visited)
// function to calculate the nearest unvisited neighboring point
{
        float min_dist = 9999999, dist = 0;
        int nearest;
        for (int i = 0; i < n; i++) {
                //if not visited then check distance
                if (Visited[i] == false) {
                        dist = (P[A].x - P[i].x)*(P[A].x - P[i].x) + (P[A].y - P[i].y)*(P[A].y - P[i].y);
                        dist = sqrt(dist);
                }
                if (dist < min_dist) {
                        min_dist = dist;
                        nearest = i;
                }
        }

        return nearest;

}
```

$Rt = O(n)$

$1+1$

1

$6+1+6+1+6+1+6$ } $\max(29,0) = 29$

$1+1$

$\sum_{i=0}^{n} 31 = 31n+31$

1 } $\max(2,0) = 2$

1

$3+31n+31+1 = 31n+35 \in O(n)$

Running time Nearest Neighbor

$n+n^2+4+n+5+28n+29$

$n^2+30n+38 \in O(n^2)$

$n^2+30n+38 \leq Cn^2 \quad \forall n \geq n_0$

Let $C = 70$

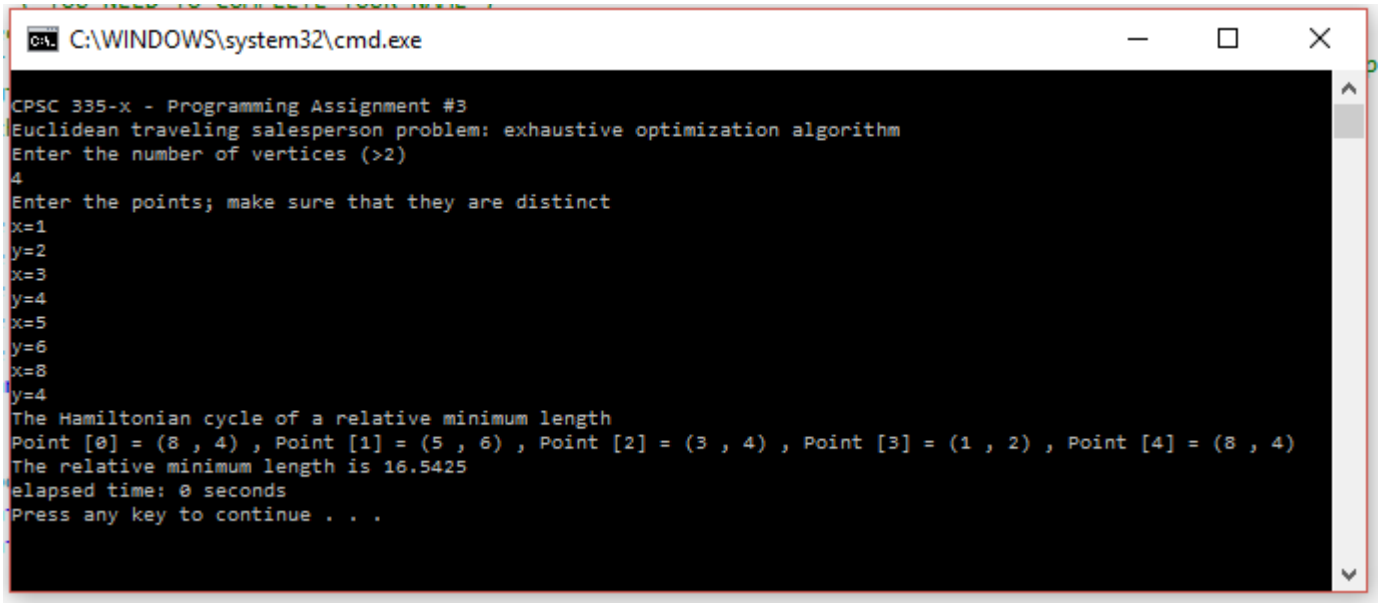
$n^2+30n+38 \leq 70n^2 \quad \forall n \geq n_0$

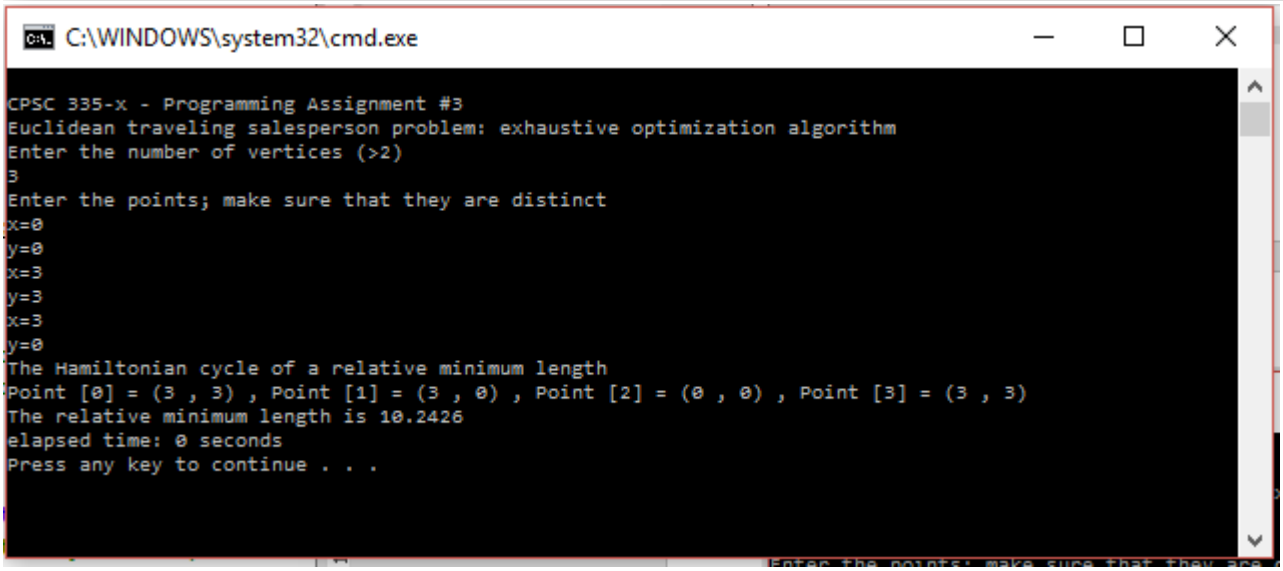
$30n+38 \leq 69n^2 \quad \forall n \geq n_0$

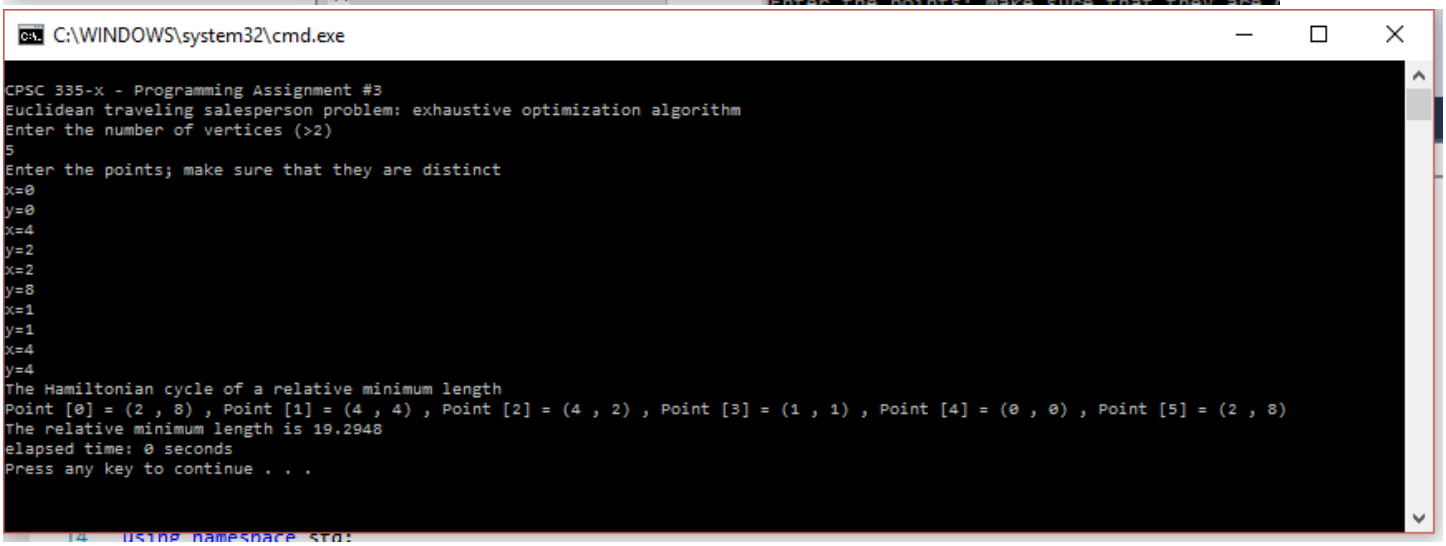
if $n=1$ $\quad 68 \leq 69$ ✓ therefore $n^2+30n+38 \leq Cn^2 \forall n \geq 1$

therefore

Nearest Neighbor $O(n^2)$ ✓


C:\WINDOWS\system32\cmd.exe
CPSC 335-x - Programming Assignment #3
Euclidean traveling salesperson problem: exhaustive optimization algorithm
Enter the number of vertices (>2)
4
Enter the points; make sure that they are distinct
x=1
y=2
x=3
y=4
x=5
y=6
x=8
y=4
The Hamiltonian cycle of a relative minimum length
Point [0] = (8 , 4) , Point [1] = (5 , 6) , Point [2] = (3 , 4) , Point [3] = (1 , 2) , Point [4] = (8 , 4)
The relative minimum length is 16.5425
elapsed time: 0 seconds
Press any key to continue . . .


C:\WINDOWS\system32\cmd.exe
CPSC 335-x - Programming Assignment #3
Euclidean traveling salesperson problem: exhaustive optimization algorithm
Enter the number of vertices (>2)
3
Enter the points; make sure that they are distinct
x=0
y=0
x=3
y=3
x=3
y=0
The Hamiltonian cycle of a relative minimum length
Point [0] = (3 , 3) , Point [1] = (3 , 0) , Point [2] = (0 , 0) , Point [3] = (3 , 3)
The relative minimum length is 10.2426
elapsed time: 0 seconds
Press any key to continue . . .


C:\WINDOWS\system32\cmd.exe
CPSC 335-x - Programming Assignment #3
Euclidean traveling salesperson problem: exhaustive optimization algorithm
Enter the number of vertices (>2)
5
Enter the points; make sure that they are distinct
x=0
y=0
x=4
y=2
x=2
y=8
x=1
y=1
x=4
y=4
The Hamiltonian cycle of a relative minimum length
Point [0] = (2 , 8) , Point [1] = (4 , 4) , Point [2] = (4 , 2) , Point [3] = (1 , 1) , Point [4] = (0 , 0) , Point [5] = (2 , 8)
The relative minimum length is 19.2948
elapsed time: 0 seconds
Press any key to continue . . .