Assignment 3: The Rectilinear Traveling Salesperson Problem

Pseudocode: Exhaustive Optimization Algorithm

void print_cycle(int n, point2D *P, int *seq)

```
For i to n
        cout<<"(" << P[seq[i]].x << "," << P[seq[i]].y << ")";
End For
cout<<"(" << P[seq[0]].x << "," << P[seq[0]].y << ") "<<endl;
```

$$\sum_{i=0}^{n} 1 = (\frac{n-0}{1} + 1)1 = n + 1$$

T.C : n + 1
Big-Oh: O(n)

---

void print_perm(int n, int *A, int sizeA, point2D *P, int *&bestSet, float &bestDist)

```
(1) int i;
(1) float dist;

if (n == 1) {
        (1) dist = 0;
        For i to sizeA - 1
                dist += sqrt((pow((P[A[i]].y - P[A[i + 1]].y), 2))+(pow((P[A[i]].x - P[A[i + 1]].x), 2)));
        End For
```

$$\sum_{i=0}^{n-1} 8 = (\frac{n-1-0}{1} + 1)8 = 8n$$

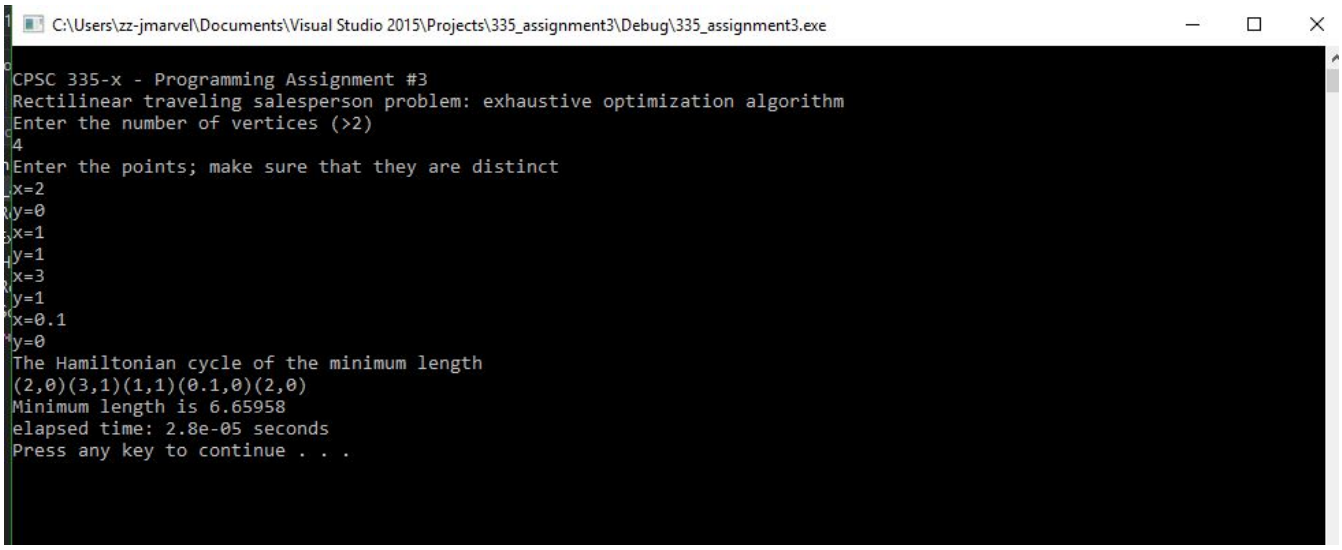
```
        (8)     dist += sqrt((pow((P[A[sizeA - 1]].y - P[A[0]].y), 2))+(pow((P[A[sizeA - 1]].x -
P[A[0]].x), 2)));

        if (dist < bestDist)
                bestDist = dist;
                for i to sizeA - 1
                        bestSet[i] = A[i];
                End For
        End If
```

$$\sum_{i=0}^{n-1} 1 = (\frac{n-1-0}{1} + 1)1 = n$$ ---------- > n+1

```
else {                                                    (n!  [recursion])
        for i to n - 1
                print_perm(n - 1, A, sizeA, P, bestSet, bestDist);
                if (n % 2 == 0) {
                        int temp = A[i];
                        A[i] = A[n - 1];
                        A[n - 1] = temp;
                End If
                else
                {
                        int temp = A[0];
                        A[0] = A[n - 1];
                        A[n - 1] = temp;
                End Else
        End For
        End Else
                print_perm(n - 1, A, sizeA, P, bestSet, bestDist);
        End If
                                        1 + max(n+1, n!) = 1 + n!
```

T.C: (8n + 10) * (n!)
Big-Oh: O(n * n!)



Pseudocode: Improved Nearest Neighbor Algorithm

```
int farthest_point(int n, point2D *P)
        (2) int i, j = 0, index = 0;
        (2) float dist = 0, max_dist = 0;

        for i to n-1
                dist = 0;
                for (j = 0; j < n - 1; 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); (9)
                        if (max_dist < dist)                                    1 + max(3,0) = 4
                                max_dist = dist;
                                index = i;
                                dist = 0;
                        End If
                End For
        End For
        (1) return index;
```

$$\sum_{i=0}^{n-1}\sum_{i=0}^{n-1} 13 = (\frac{n-1-0}{1}+1)13 = 13n = \sum_{i=0}^{n-1} 13n = (\frac{n-1-0}{1}+1)13n = 13n(n)$$

T.C: 13n(n) + 4 = 13 n^2 + 4

Big-Oh: O(n^2)

```
int nearest(int n, point2D *P, int A, bool *Visited)
        (2) int i, index = 0;
        (2) float dist = 0, min_dist = 0;

        for i to n
                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);

                if (!Visited[i])                        1 + (1 + max(3,0)) = 1+ (1+3) = 5
                        if (dist < min_dist)
                                min_dist = dist;
                                index = i;
                                dist = 0;

                End If
        End For
        (1) return index;
```

$$\sum_{i=0}^{n} 14 = (\frac{n-0}{1}+1)14 = 14n+14$$

T.C: 14n + 14

Big-Oh: O(n)

---

void print_cycle(int n, point2D *P, int *seq)

```
For i to n
        cout << "(" << P[seq[i]].x << "," << P[seq[i]].y << ")";
End For
```

$$\sum_{i=0}^{n} 1 = (\frac{n-0}{1} + 1)1 = n + 1$$

```
cout << "(" << P[seq[0]].x << "," << P[seq[0]].y << ") " << endl;
```

$$(n+1)+1 = n + 2$$

T.C: n + 2
Big-Oh: O(n)