

Mini Project - Group 1

ET0702 Data Structures and Algorithms

1. Credentials:

Name	Adm No.
Brians Tjipto Meidianto	xxxxxxx
Yee Yi Feng	xxxxxxx
Lin Lihong Albert	xxxxxxx

Project Topic 1 - Warehouse rack robot

2. Contributions

Brians Tjipto Meidianto: File Reading, Algorithm, Display, Report

Yee Yi Feng: Algorithm, Report

Lin Lihong Albert: Display

3. Description of Design and Features of Program

```
ifstream myfile( s: "data.txt", mode: ios::in);

while (myfile >> tmp) {
    locations.push_back(tmp);
    count++;
}

for (int i = 0; i < locations.size(); i += 2) {
    numx = locations[i];
    locx.push_back(numx);
}

for (int x = 1; x < locations.size(); x += 2) {
    numy = locations[x];
    locy.push_back(numy);
}
```

We read the file and put it into an array before splitting the array with for loops to separate the respective X-coordinates and Y-coordinates into different vectors.

```

void permutation() {
    int n = locx.size();
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        a[i] = (i + 1);
    }
    do {
        set(a, n);
    } while (next_permutation( first: a.begin(), last: a.end()));
}

void set(vector<int> a, int n) {
    for (int i = 0; i < n; i++) {
        int x = a[i];
        permutation_vec.push_back(x);
    }
}

```

Next, we will get all the permutations by creating another vector with one-half of the size of the coordinate, using next_permutation.

```

const int column = locx.size();
int row = 1;
for (int i = 1; i <= column; ++i)
    row *= i;

vector<vector<int>> permutation( n: row, value: vector<int>( n: column, value: 0));
int k = 0;
for (int i = 0; i < row; i++) {
    for (int j = 0; j < column; j++) {
        permutation[i][j] = permutation_vec[k++];
    }
}

```

Following that, we found the factorial of the amount of location given, before converting the 1D permutation vector to a 2D permutation vector.

```

for (int i = 0; i < row; i++) {
    int sum = 0, count_seq = 0, sum_0 = 0;
    for (int x = 0; x < column - 1; x++) {
        int temp = permutation[i][x] - 1;
        int temp2 = permutation[i][x + 1] - 1;
        int sum_temp = abs( locx[temp] - locx[temp2]) + abs( locy[temp] - locy[temp2]);
        sum += sum_temp;
        if (count_seq == 0) {
            sum_0 = abs( locx[temp] - locx[temp2]) + abs( locy[temp] - locy[temp2]);
        }
        count_seq++;
    }
    sum += sum_0;
    vec_sum.push_back(sum);
}

```

We then get the distance of each permutation before placing it into another 1D vector called vec_sum.

Next, we have two int lines where we find the minimum value of the vec_sum with its corresponding index and distance.

```
int minDistanceIndex = (min_element( first: vec_sum.begin(), last: vec_sum.end()) - vec_sum.begin());
int minDistance = *min_element( first: vec_sum.begin(), last: vec_sum.end());
```

```
for (int i = 0; i < column; i++) {
    int tempseq = permutation[minDistanceIndex][i];
    seq.push_back(tempseq);
}

cout << "Sequence: ";
for (int i = 0; i < seq.size(); i++) {
    cout << seq[i];
    if (i < seq.size() - 1) {
        cout << ",";
    }
}
}
```

Following that, we will push the permutation into a sequence vector, before printing out the sequences and visuals

```
string NumToOrd(size_t num) {
    string suffix = "th";
    if (num % 100 < 11 || num % 100 > 13) {
        switch (num % 10) {
            case 1:
                suffix = "st";
                break;
            case 2:
                suffix = "nd";
                break;
            case 3:
                suffix = "rd";
                break;
        }
    }
    return to_string( val: num) + suffix;
}
```

To print out visuals, we have a NumToOrd function to convert the numbers to their ordinal form (1 becomes 1st, 2 becomes 2nd, and so on) for a dynamic number sequencing then, we will print out the coordinates the robot arm has moved to with the help of the sequence vector.

```

void display(vector<vector<int>> v) {
    for (vector<int> visual10 : v) {
        for (int x : visual10) {
            cout << (char)x << " ";
        }
        cout << endl;
    }
}

void visuals() {
    vector<vector<int>> visual( n: 6, value: vector<int>( n: 7, value: '0' ));

    visual[5][0] = 'H';

    for (int i = 0; i < locx.size(); i++) {
        char x = 'X' + 1;
        visual[5 - locy[i]][0 + locx[i]] = 'X';
    }
    display( v: visual);
    cout << "(" << NumToOrd( num: 1) << " display)" << endl;
    cout << "H - home (0,0)\n\n";

    for (int i = 0; i < locx.size(); i++) {
        int y = seq[i] - 1;
        visual[5 - locy[y]][0 + locx[y]] = (y + 49);
        display( v: visual);
        cout << "(" << NumToOrd( num: i + 2) << " display)" << endl;
        cout << "(" << locx[y] << ", " << locy[y] << ") retrieved\n\n";
    }
}

```

To print out the table, we used a 2D vector. Set up the “home” for the robot arm. Set up the points according to the txt file. Before using the NumToOrd function to print 1st before printing the table with the home and points. Then, display the rest of the points with the corresponding points.

```

visuals();
cout << "Total distance moved: " << minDistance << endl;

```

And finally, we printed the distance traveled.

```

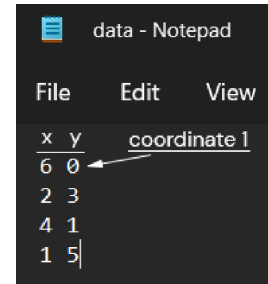
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <algorithm>
#include <cmath>

```

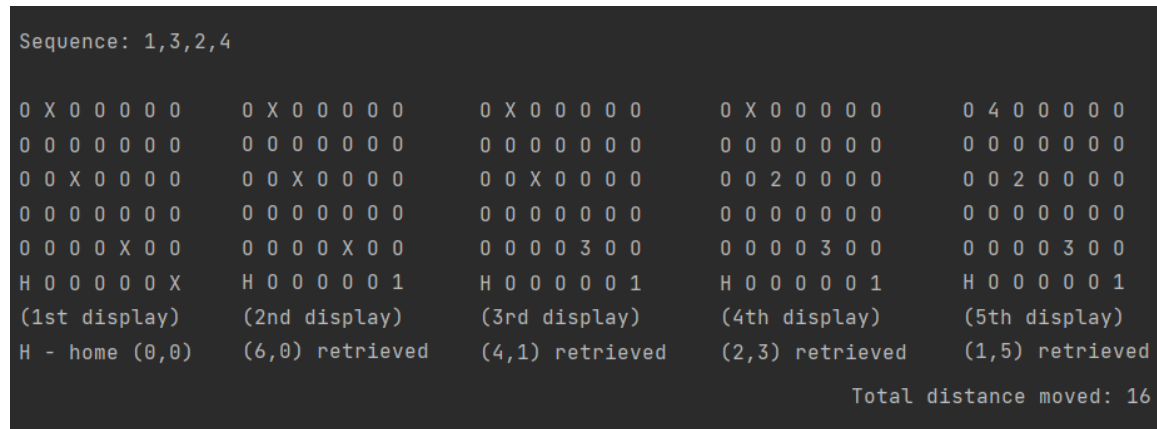
Here is all the imported libraries that we use.

4. Program User Guide

How the user will use the program is, that they will key in the coordinates into the txt file (data.txt) with the format of cord 1 "x y" \n cord 2 "x y" till they have listed the all the coordinates. After they have run the program, what they will see first is the sequence in which the robot arm will go first, cord 1 being point 1, etc. next, the user will see a more detailed variant with a move counter, origin to destination, origin point, endpoint, and distance moved. After that, the user will see the layout and where the arm has been retrieved with each display.



```
data - Notepad
File Edit View
x y coordinate 1
6 0
2 3
4 1
1 5
```



```
Sequence: 1,3,2,4

0 X 0 0 0 0 0 0 X 0 0 0 0 0 0 X 0 0 0 0 0 0 X 0 4 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 X 0 0 0 0 0 0 X 0 0 0 0 0 0 X 0 2 0 0 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 X 0 0 0 0 0 0 X 0 0 0 0 0 3 0 0 0 0 0 3 0 0 0 0 0
H 0 0 0 0 0 X H 0 0 0 0 0 1 H 0 0 0 0 0 1 H 0 0 0 0 0 1 H 0 0 0 0 0 1
(1st display) (2nd display) (3rd display) (4th display) (5th display)
H - home (0,0) (6,0) retrieved (4,1) retrieved (2,3) retrieved (1,5) retrieved
Total distance moved: 16
```

5. Research and References

Cplusplus.com. 2022. next_permutation - C++ Reference. [online] Available at: https://cplusplus.com/reference/algorithm/next_permutation/ [Accessed 15 July 2022].
GeeksforGeeks. 2022. Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming) - GeeksforGeeks. [online] Available at: <https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/> [Accessed 12 July 2022].

6. Reflection

Brians:

This Project has taught me a lot about C++, I have learned ways to tackle all the roadblocks that I've faced during the coding process, from file reading not reading properly to the various algorithms that we have to use and learn. Yi Feng started looking at Dijkstra's shortest path algorithm but it is not very optimal as we needed to add tracking for points visited, and we didn't understand the code well. Then I came up with using the Pythagoras theorem to find the shortest distance from point one to point two but we found out that it will not work with every scenario. Towards the end, we landed on the traveling salesman problem, we learned and understood the algorithm to use it as a

reference for our algorithm, Yi Feng and I worked off of each other really well even though we couldn't explain our ideas to the fullest, but I think in the end it worked out. And finally, the displaying part was also a challenge as we had to make it dynamic, and show the sequence unfolding one by one.

Yi Feng:

The DSA project itself wasn't too complicated, to begin with. It was just that there were too many ways to do it. For instance, brains thought of Pythagoras' theorem to calculate the shortest path between each point, even though it didn't work because the shortest path between each point does not equate to the shortest overall path. So I came over Dijkstra's Algorithm and there was even a code online that we could refer to. However, I didn't understand the code and the code wasn't dynamic as required in the question. Hence, Brains came to me with the idea of permutation. Which we came about when researching the traveling salesman problem I thought it would've been too computation intensive but it wasn't true. We then developed it together. We cliqued pretty well, able to work off each other's ideas and solutions, able to understand what each other was saying despite being unable to fully explain our ideas. So in general, the code itself had its hiccups where we had the issue of the code checking itself with the Pythagoras theorem however, it was generally manageable.

Albert:

This project was relatively manageable, even if there were hiccups at the beginning. I was tasked with the coding of the display of the vector and their respective positions for each step the robot takes to retrieve the items in the given sequence from their respective coordinates. I have also added the code to output the final vector to a .txt file called RetrievalComplete. After which, I then defined three empty rack vectors called rack1, rack2, and rack3, and used a switch-case statement to select which rack to retrieve the items from their respective coordinates. While most of the code is dynamic and allows flexible input (the display), what requires improvement is the dynamic allocation of the multiple racks (i.e. Racks 1, 2, and 3 etc...), such that it is not limited to only 3 racks, and reduces the clutter in the code. Also, I can be more independent when it comes to implementing my code in this project. The code can also be further decluttered by using functions and passing variables through them, instead of using chunks of code in the main() section.