Math 381 **HW#4** Xinming Zhang

Optimum trip:

I picked ten buildings on our campus to go for my optimum trip problem. The buildings I picked are Odegaard Undergraduate Library(OUG), Mary gates hall(MGH), Gould hall(GLD), Paul G. Allen Center(CSE), Bagley Hall(BAG), Fluke hall(FLK), Denny hall(DEN), Alaska airlines arena(EDP), Husky stadium(STD), and Hitchcock(HCK). I list them in order from location 1 to 10. Those are the ones either I will go to or the ones I have been there. Then, I use google maps to check the distance between them manually. The distance could be shown as following table, the units of the distance is in 100 meters:

City number	1	2	3	4	5	6	7	8	9	10
1 OUG	0	4.5	4.5	8	5	8	3	11	12	7
2 MGH	4.5	0	5	3	2	6	5	7	7	6
3 GLD	4.5	5	0	8	6	10	7	12	10	3
4 CSE	8	3	8	0	3.5	4	8	3.5	4.5	6
5 BAG	5	2	6	3.5	0	6	7	7	7	4.5
6 FLK	8	6	10	4	6	0	7	5	7	10
7 DEN	3	5	7	8	7	7	0	11	12	9
8 EDP	11	7	12	3.5	7	5	11	0	9	9
9 STD	12	7	10	4.5	7	7	12	9	0	7
10 HCk	7	6	3	6	4.5	10	9	9	7	0

Distance table (Unit: 100m)

We are trying to get the shortest path between some of the locations, so we are minimizing the total traveled distance through the locations. I will use solving a LP (Linear Program) to get the optimum trip between the locations.

First, we are going to define our variables.

Set n as the total number of locations, in this case n is equal to 10.

Set m as the number of locations we want to visit (number of locations on the shortest path).

Define $x_{i,j}$ as a binary variable (either be 0 or 1) for all $1 \leq i \leq m, 1 \leq j \leq n.$

$x_{i,j} = 1$ represents we are in location j at step i, when it equals 0, that means we are not in location j at step i.

Define $e_{a,b}$ as a binary variable (either be 0 or 1) for all $1 \leq a, b \leq n.$

$e_{a,b} = 1$ we traveled from location a to location b, when it equals 0, that means we are not traveling from location a to location b.

Define $d_{a,b}$ as a the distance between location a and b, for all $1 \leq a, b \leq n$.

We are trying to minimize the total traveled though m out of n locations. Using the variable we defined, we can get the objective function as:

$$\text{Minimize: } \sum_{1 \leq a,b \leq n} e_{a,b} d_{a,b}$$

Then, we need to set out constraints.
First one will be for our each step we are only visiting one location (one step one location), then it will be as:

$$\sum_{j=1}^{n} x_{i,j} = 1 \ , i = i,\ldots, m$$

Second one will be every location will be visited at most once, and the constraint will be as:

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \ , j = i,\ldots, n$$

Third one be the definition of variable $e_{a,b}$, when it equals to 1 if and only if we travel from location a to location b. Then, $x_{i,a} + x_{i+1,b}$ must be 2 if $e_{a,b}$ equals to 1. If either or both of $x_{i,a}$ and $x_{i+1,b}$ is 0, then $e_{a,b}$ is zero. Sum that up to get our third constraint as:

$$e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1 \ , i = 1,\ldots, m - 1$$

For last constraint, we will need to make sure both variable $e_{a,b}$, $x_{i,j}$ are binary variables (either 0 or 1):

$$e_{a,b}, x_{i,j} \in \{0, 1\} \quad 1 \leq i \leq m, \ 1 \leq a, b, j \leq n$$

Above all, we can get our LP in compact form as:

$$\text{Minimize: } \sum_{1 \leq a,b \leq n} e_{a,b} d_{a,b}$$

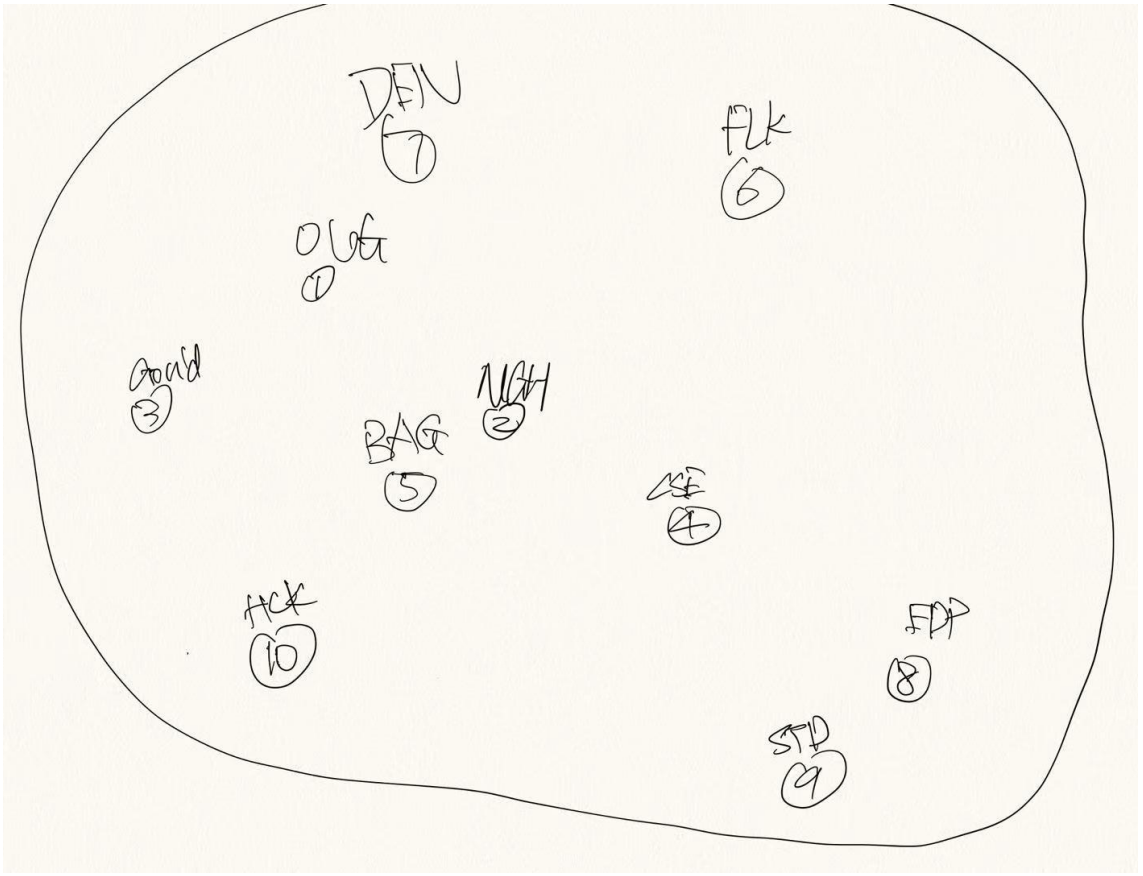
$$\text{Subject to: } \sum_{j=1}^{n} x_{i,j} = 1 \ , i = i,\ldots, m$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \ , j = i,\ldots, n$$

$$e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1 \ , i = 1,\ldots, m - 1$$

$$e_{a,b}, x_{i,j} \in \{0, 1\} \quad 1 \leq i \leq m, \ 1 \leq a, b, j \leq n$$

A hand-drawn map to visualize the locations (buildings):



Then we generate the code as following:
Coding:

```
public class Math381_4_1 {

    public static void main(String[] args) {
        //The 2d array d stand for the distance between two cities, di,j stands for
        //the distance between city i and j
        double[][] d = new double[][]{{0, 4, 4.5, 8, 5, 8, 3, 11, 12, 7},
        {4.5, 0, 5, 3.5, 2, 6, 5, 7, 7, 6}, {4.5, 5, 0, 8, 6, 10, 7, 12, 10, 3},
        {8,3.5,8,0,3.5,4,8,3.5,4.5,6}, {5,2,6,3.5,0,6,7,7,7,4.5},
        {8,6,10,4,6,0,7,5,7,10},{3,5,7,8,7,7,0,11,12,9},{11,7,12,3.5,7,5,11,0,9,9},
        {12,7,10,4.5,7,7,12,7,0,7},{7,6,3,6,4.5,10,9,9,7,0}};

        //total number of cities, always 10
        int n = 10;
        //number of cities we want to connect in the shortest path
        int m = 5;

        //object function minimize the connected length
        System.out.print("min: ");
```

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        if(i != j){
        System.out.print("+ " + d[i - 1][j - 1] + "e_" + i + "_" + j);
        }
    }
}

System.out.println(";");


//e_a,b  iff x_i,a and 1+ x_i+1,b = 2
for (int i = 1; i <= m + 1; i++) {
    for (int a = 1; a <= n; a++) {
        for (int b = 1; b <= n; b++) {
            System.out.print("e_" + a + "_" + b);
            System.out.print(" >= ");
            int k = i + 1;
            System.out.println("x_" + i + "_" + a + "+ x_" + k + "_" + b + " -1;");
        }

    }
}


//Only one city each step constraint
for (int i = 1; i <= m; i++) {
    for (int j = 1; j <= n; j++) {
        if (j == n) {
            System.out.println(" + " + "x_" + i + "_" + j + " = 1;");
        } else {
            System.out.print(" + " + "x_" + i + "_" + j);
        }
    }
}

//Each city can not be visited twice (only once)
for (int j = 1; j <= n; j++) {
    for (int i = 1; i <= m; i++) {
        if (i == m) {
            System.out.println(" + " + "x_" + i + "_" + j + " <= 1;");
        } else {
            System.out.print(" + " + "x_" + i + "_" + j);
        }
    }
}
```

```
        //solutions are binary numbers (0 or 1)
        System.out.print("bin");
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                System.out.print(" x_" + i + "_" + j + ",");
            }
        }
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (j == n && i==n) {
                    System.out.println(" e_" + i + "_" + j + ";");
                } else {
                    System.out.print(" e_" + i + "_" + j + ",");
                }
            }
        }
    }

}
```

Running the code to get the input file as:

```
min: + 4.0e_1_2+ 4.5e_1_3 + … + 9.0e_10_7+ 9.0e_10_8+ 7.0e_10_9;
e_1_1 >= x_1_1+ x_2_1 -1;
e_1_2 >= x_1_1+ x_2_2 -1;
…….
e_10_6 >= x_6_10+ x_7_6 -1;
e_10_7 >= x_6_10+ x_7_7 -1;
e_10_8 >= x_6_10+ x_7_8 -1;
e_10_9 >= x_6_10+ x_7_9 -1;
e_10_10 >= x_6_10+ x_7_10 -1;
 + x_1_1 + x_1_2 + x_1_3 + x_1_4 + … + x_1_9 + x_1_10 = 1;
...
 + x_5_1 + x_5_2 + x_5_3 + x_5_4 + ... + x_5_9 + x_5_10 = 1;
 + x_1_1 + x_2_1 + x_3_1 + x_4_1 + x_5_1 <= 1;
 + x_1_2 + x_2_2 + x_3_2 + x_4_2 + x_5_2 <= 1;
 ...
 + x_1_9 + x_2_9 + x_3_9 + x_4_9 + x_5_9 <= 1;
 + x_1_10 + x_2_10 + x_3_10 + x_4_10 + x_5_10 <= 1;
bin x_1_1, x_1_2, x_1_3, x_1_4, ... , e_10_9, e_10_10;
```

Using the input file above to get the solution for m = 5 (shortest path visiting 5 locations):
Value of objective function: 12.50000000

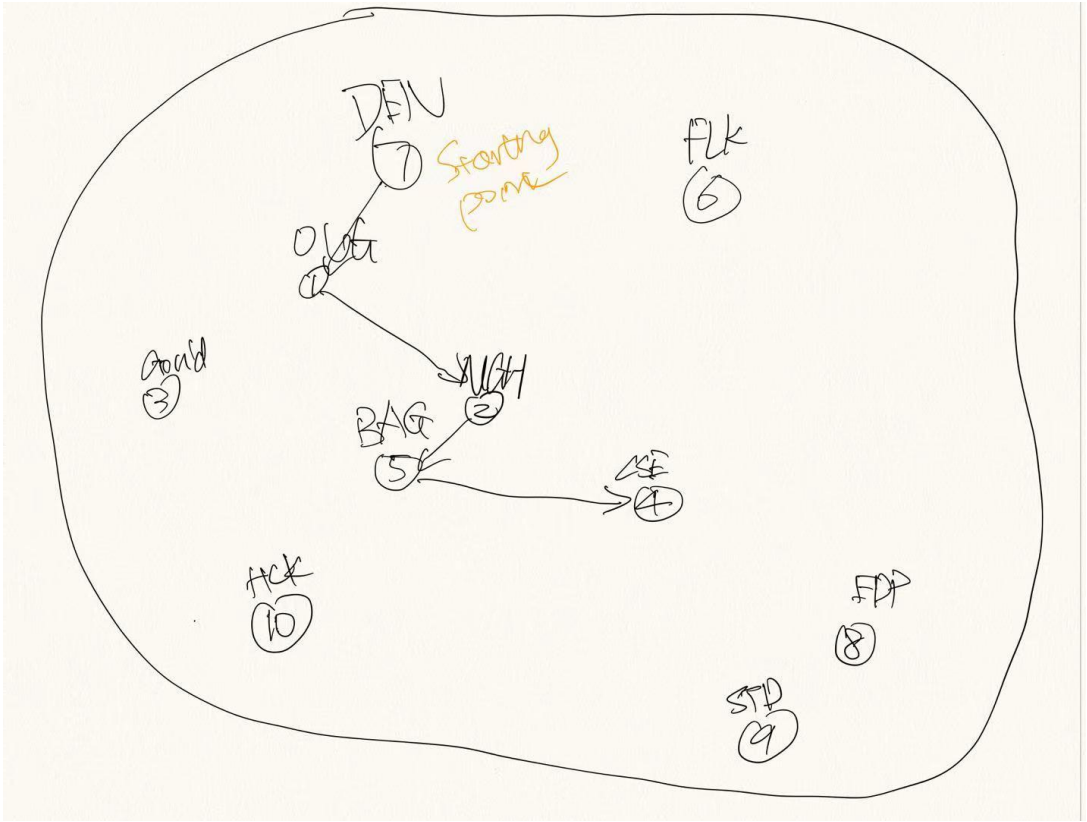
Actual values of the variables:

e_1_2	1
e_2_5	1
e_5_4	1
e_7_1	1
x_2_1	1
x_1_7	1
x_3_2	1
x_4_5	1
x_5_4	1

All other variables are zeros.
Used 1.137 seconds.

From the solution we can know, the length of the shortest path going through 5 locations out of 10 locations I picked will be 1250 meters. We will start at location 7 which is Denny hall(DEN). Then we will go through Odegaard Library(OUG), Mary Gates Hall (MGH), Bagley Hall (BAG), and Pual G. Allen Center (CSE) in order. We can express them in locations' numbers as 7-1-2-5-4, and the total traveled distance is 1250 meters.

Visualization (m=5):

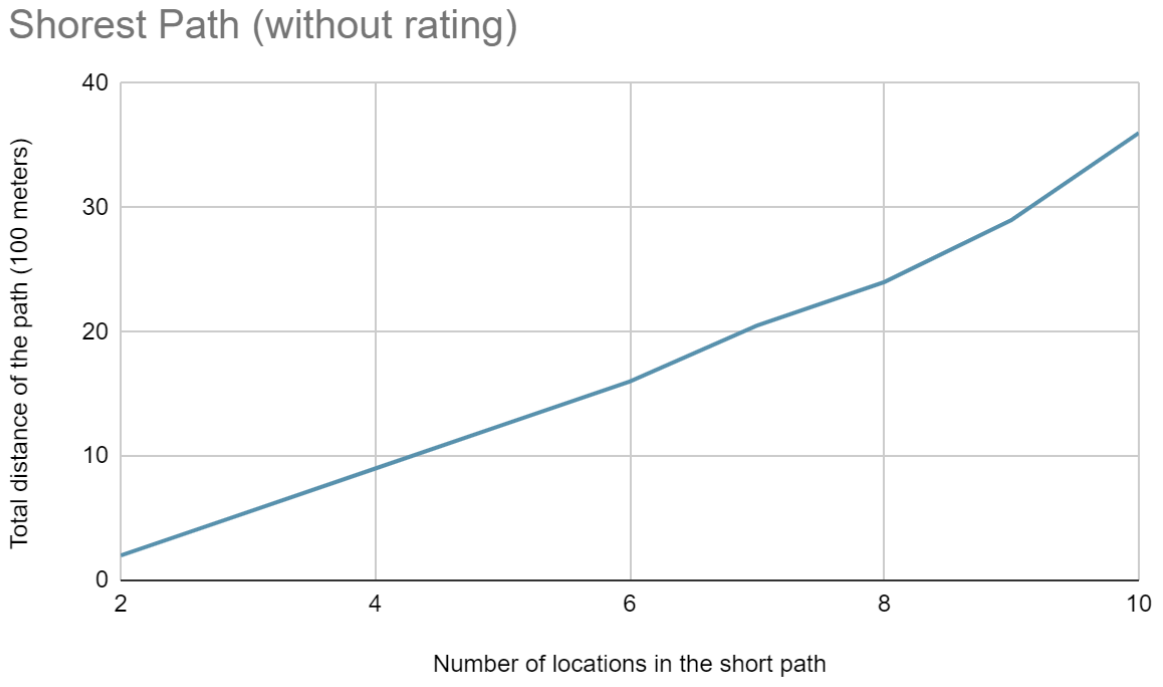


(Makes sense by just looking at the map)

We can generate the code and get the result for m = 2 to 10 (including 5) as the table next page:

Number of location traveled (variable m)	The length of shortest path(Units: 100 meters)	Locations' order in the path (in numbers form)	Runtime (Units: sec)
2	2	5-2	0.19
3	5.5	5-2-4	0.03
4	9	8-4-5-2	0.132
5	12.5	7-1-2-5-4	1.137
6	16	7-1-2-5-4-8	7.13
7	20.5	7-1-3-10-5-2-4	33.691
8	24	7-1-3-10-5-2-4-8	144.123
9	29	7-1-3-10-5-2-4-8-6	1436.023
10	36	9-2-8-6-2-5-10-3-1-7	4581

From the result table we can get the plot as :



There are some findings from the result table and the plot we got:

1. The runtime and the length of the short paths is increasing by increasing the variable m, which is the number of locations we want to travel. For m=10, the runtime is huge, and it is about 76 minutes.
2. From the path column in the result table, we can see that the short path 2-5 or 5-2 is always included in every path. That means no matter which buildings we are going to, the path from Mary Gates Hall to Bagley Hall is always the shortest path. That makes sense, because from the map we can see they are the closest ones. The paths 1-7 or 7-1 start to be included from m = 5. Location 1 is Odegaard Library, and Location 7 is Denny Hall. Looking at the map only by eyes, they could be the second shortest pair. That is why they are always on the path.
3. From the plot (graph), the relationship between the number of locations traveled and total distance traveled are in a positive linear relationship. The plot is close to a linear line with a positive slope. Especially for the first five points, the line is almost straight. From m = 6, the line is not that straight, but still a positive relationship.

Then we move on to the problem with a rating for each location.
First, here is my rating for each location:

Location number	Rating
1	7
2	6
3	7
4	5
5	8
6	5
7	9
8	8
9	10
10	5

We will change the object function a little bit, we will maximize the total rating of the locations we visited as:

$$\text{Maximize: } \sum_{1 \le i,j \le n} R_j x_{i,j}$$

Where R_j represent the rating the location j.

In this case, we will not set a number of locations we want to travel, that means we are using variable m anymore. Then, the second constraint for the previous part will change to:

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , j = 1,\dots, n$$

On the other hand, I am adding two new constraints to the LP.
The first one will be the upper bound of the total distance traveled.
We can modify it as:

$$\sum_{1 \leq i,j \leq n} e_{i,j} d_{i,j} \leq B$$

Where B is the upper bound (traveled distance) we want to input.
The second one will be added is we cannot skip steps, which means we must take (i-1)th step first to take ith step (i is between 2 to n-1 to make (i-1) not smaller than 1):

$$\sum_{j=1}^{n} x_{i,j} \leq \sum_{j=1}^{n} x_{i-1,j} \quad , i = 2,\dots, n-1$$

Above all, we can get our new LP in compact form as:

$$\text{Maximize: } \sum_{1 \leq i,j \leq n} R_j x_{i,j}$$

$$\text{Subject to: } \sum_{1 \leq i,j \leq n} e_{i,j} d_{i,j} \leq B$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \quad , i = 1,\dots, n$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1 \quad , j = 1,\dots, n$$

$$e_{a,b} \geq x_{i,a} + x_{i+1,b} - 1 \quad , i = i,\dots, n-1$$

$$\sum_{j=1}^{n} x_{i,j} \leq \sum_{j=1}^{n} x_{i-1,j} \quad , i = 2,\dots, n-1$$

$$e_{a,b}, x_{i,a} \in \{0, 1\} \quad , 1 \leq a, b, i, j \leq n$$

We have to change the code a little bit according to the LP:
We will change every variable m appeared in the code to variable n.
Then changing the objective function and adding the rating list as:

```
//rating array for each location
int[] rating = new int[]{7, 6, 7, 5, 8, 5, 9, 8, 10, 5};
//n equals to the number of locations
int n = rating.length;
//New objective function with rating (maxing total rate)
```

```
        System.out.print("max: ");
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                System.out.print("+ " + rating[i - 1] + "x_" + i + "_" + j);
            }
        }
        System.out.println(";");
```

We can use the objective function code from the last part to get our first new constraint.

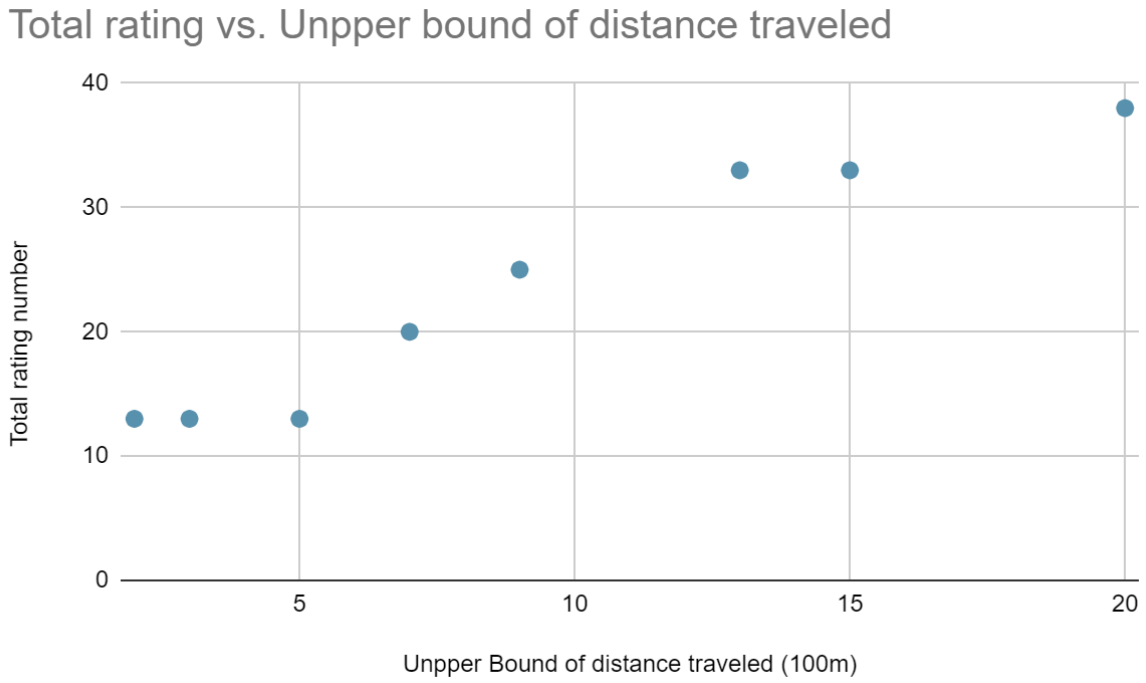
For second new one, we can write it as:

```
//Step in step, no skipping steps
for (int i = 2; i <= n; i++) {
            int k = i - 1;
            for (int j = 1; j <= n; j++) {
                System.out.print(" + " + "x_" + i + "_" + j);
            }
                System.out.print(" <= ");
            for (int j = 1; j <= n; j++) {
                System.out.print(" + " +"x_" + k + "_" + j);
            }
            System.out.println(" ;");
        }
```

Using lp_solve to solve the input file from running the code to get our results as the table:

Upper Bound B (Units: 100 m)	Number of Locations traveled	Result path represented in number form	Total ratings	Runtime (seconds)
2	2	2-5	13	0.171
3	2	3-10	13	0.343
5	2	3-1	13	0.809
7	3	3-2-5	20	1.538
9	4	5-2-4-8	25	4.836
13	5	3-10-5-2-4	33	66.632

15	5	3-1-2-5-4	33	199.089
20	6	3-1-2-4-5-10	38	1685.615



Finding: I was trying to increase the Upper Bound of the traveled distance (B) one at each time to get a better idea about the changing of the locations. However, from B = 12, the runtime starts to be pretty long. For B = 12, the runtime is getting longer than a minute. When B = 15, the runtime is more than three minutes, so I started to increase B by 5 each time. The next and the last try is B = 20, and it took a long time to get the solution (28 minutes). I did not go for the next try, because ideally it would have an even longer runtime than B = 20. From the plot, we can see the total rating might not be increasing by the increase of B. The table told us, even if the total rate might not be changed the path will be different. My ratings on each place are in integer, and some of them are the same. That could be a factor, and another one is the distance between locations (buildings). Some distances are the same as well, and those two factors made the plot like that.

Combining those two parts together. The second problem has a relationship with the first part. The upper bound B from the second part must be affected by the shorts path as well. The pair 5-2 or 2-5 appeared again in the second part. I rated them medium to medium-high, and they are the closest buildings in those ten buildings. That makes sense. However, when B = 20, the pair 2-5 broke up. I think that is caused by the setting of B. The lowest number of B to get total rating of 38 should not be 20. Ideally if we are setting upper bound (B) lower to a lowest number to get total rating of 38, 5-2 path should be included again.