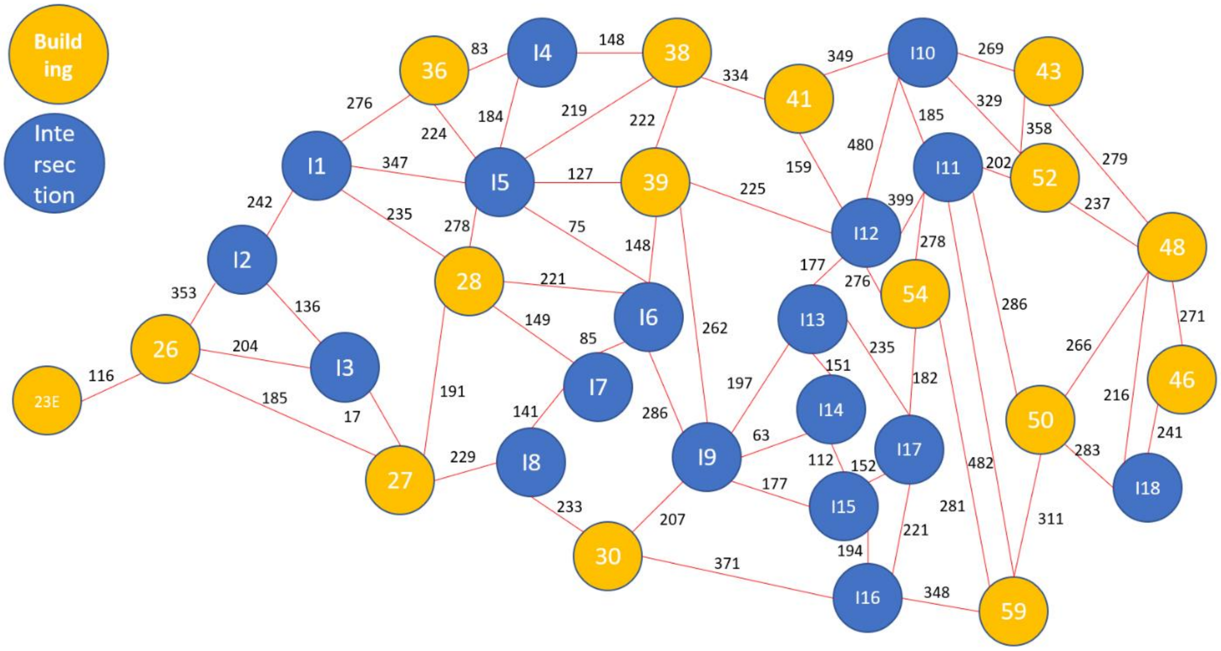
7205 HW5

Name: Xuebao Zhao   NUID: 002108354

Q1:

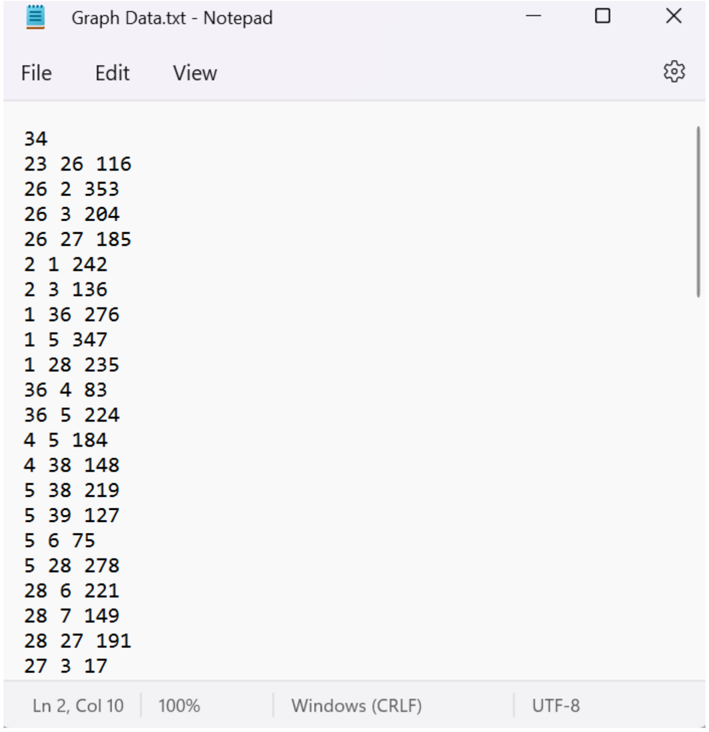
Undirected graph drawing

Select 16 buildings which makes the graph has four buildings from each one of the following numbering ranges:
20’s: 23, 26, 27, 28
30’s: 30, 36, 38, 39
40’s: 41, 43, 46, 48
50’s: 50, 52, 54, 59
Use yellow color to mark the buildings and blue color to mark the intersections. And the undirected graph is drawn as follows:



Create a text file

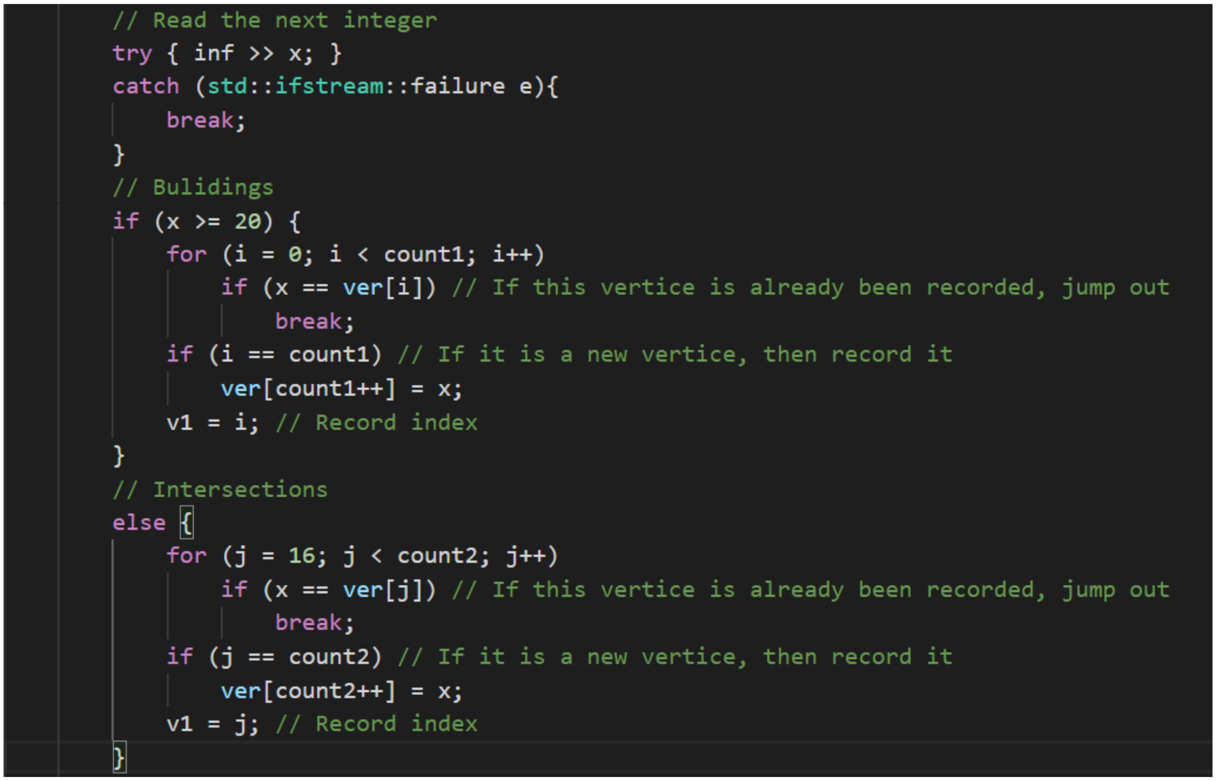
The text file includes the total number of vertices followed by the data of the graph edges. For each edge, provide its <vertex1> <vertex2> <distance>. Because the numbers of buildings are all greater than 20, I use the number under 20 to the intersections which making sure that they are different than the used building numbers. And the text file is as follows:



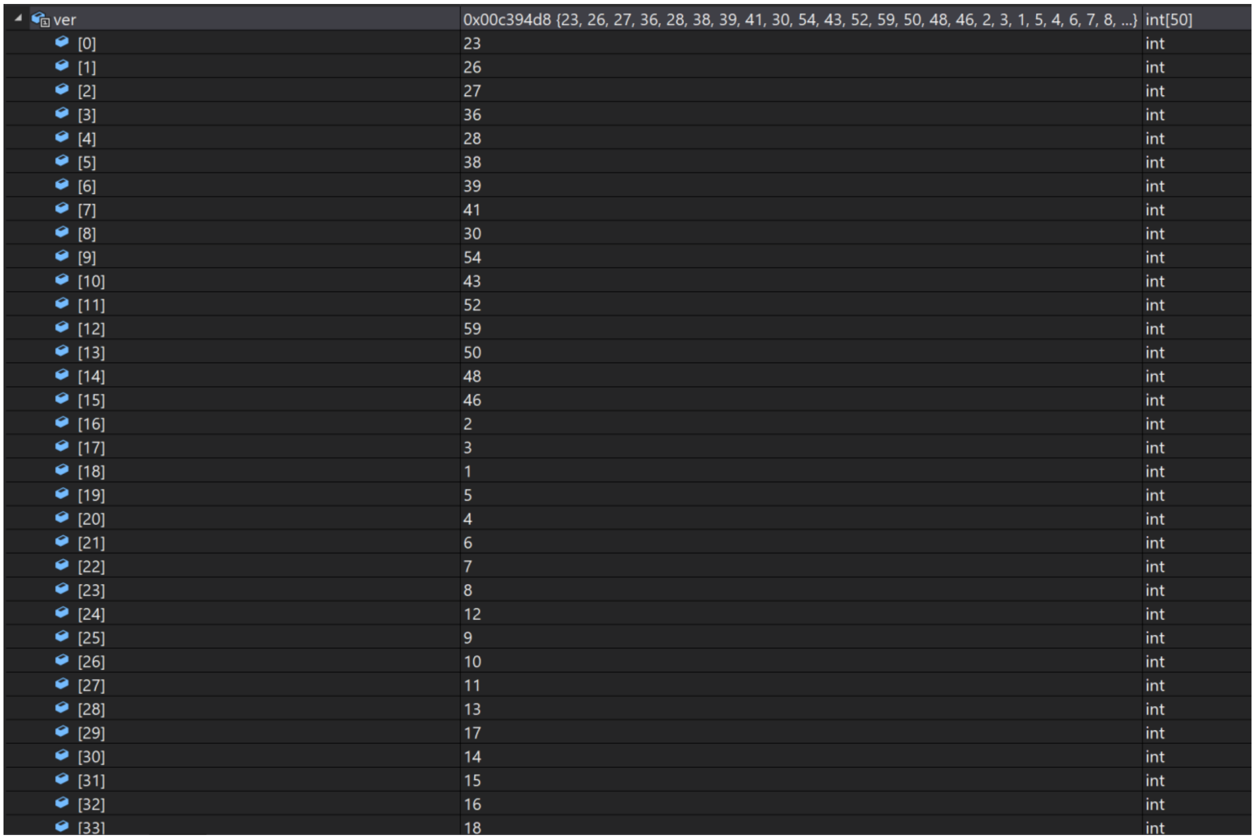
```
34
23 26 116
26 2 353
26 3 204
26 27 185
2 1 242
2 3 136
1 36 276
1 5 347
1 28 235
36 4 83
36 5 224
4 5 184
4 38 148
5 38 219
5 39 127
5 6 75
5 28 278
28 6 221
28 7 149
28 27 191
27 3 17
```

Store the vertices and edges

The program reads the '.txt' file and use an array to map the program indices to the user-friendly building and intersections numbers as stored in the text file. The first number in the file is the total number of the vertices. The remaining integers in groups of three represent an edge. When the program reads the first two integers in a group, first determines whether it is a building or an intersection. Then the program determines whether the vertex has been recorded in the array. If it is already in the array, record the index. Else store the numbers of buildings in the first fifteen positions of the array and store the numbers of intersections in the rest of the array. Then record the index.

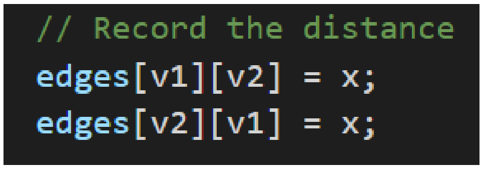


```cpp
// Read the next integer
try { inf >> x; }
catch (std::ifstream::failure e){
    break;
}
// Bulidings
if (x >= 20) {
    for (i = 0; i < count1; i++)
        if (x == ver[i]) // If this vertice is already been recorded, jump out
            break;
    if (i == count1) // If it is a new vertice, then record it
        ver[count1++] = x;
    v1 = i; // Record index
}
// Intersections
else {
    for (j = 16; j < count2; j++)
        if (x == ver[j]) // If this vertice is already been recorded, jump out
            break;
    if (j == count2) // If it is a new vertice, then record it
        ver[count2++] = x;
    v1 = j; // Record index
}
```



ver	0x00c394d8 {23, 26, 27, 36, 28, 38, 39, 41, 30, 54, 43, 52, 59, 50, 48, 46, 2, 3, 1, 5, 4, 6, 7, 8, ...}	int[50]
[0]	23	int
[1]	26	int
[2]	27	int
[3]	36	int
[4]	28	int
[5]	38	int
[6]	39	int
[7]	41	int
[8]	30	int
[9]	54	int
[10]	43	int
[11]	52	int
[12]	59	int
[13]	50	int
[14]	48	int
[15]	46	int
[16]	2	int
[17]	3	int
[18]	1	int
[19]	5	int
[20]	4	int
[21]	6	int
[22]	7	int
[23]	8	int
[24]	12	int
[25]	9	int
[26]	10	int
[27]	11	int
[28]	13	int
[29]	17	int
[30]	14	int
[31]	15	int
[32]	16	int
[33]	18	int

Store the edges in a two-dimensional array ‘edge’, such as ‘edge[i][j]’ represents distance from ‘vertice I’ to ‘vertice j’



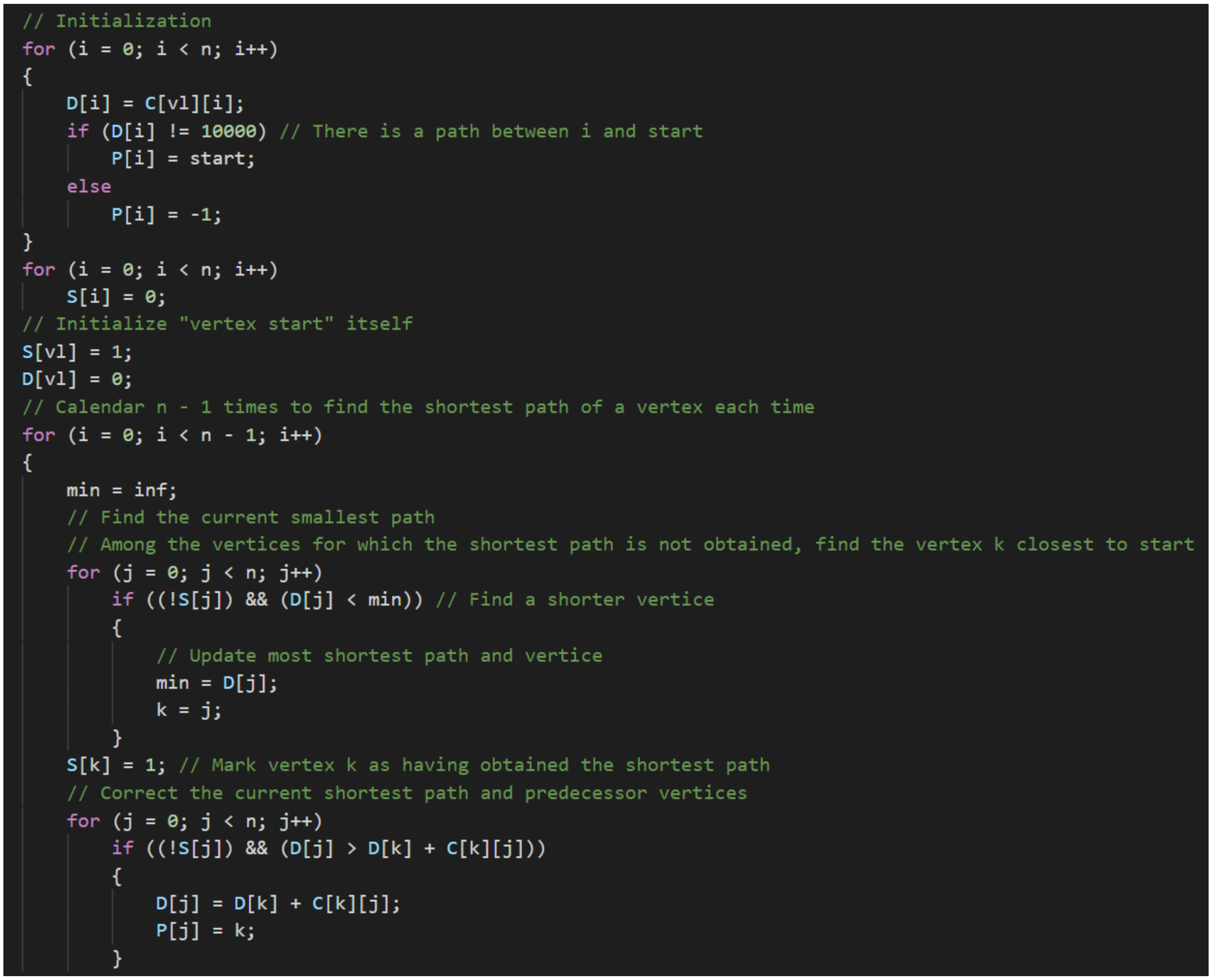
```
// Record the distance
edges[v1][v2] = x;
edges[v2][v1] = x;
```

Dijkstra

When calculating the shortest path in the graph by Dijkstra, the starting point ‘start’ needs to be specified.

Furthermore, three arrays P, S, and D are introduced. The role of P is to record the vertices for which the shortest path has been found (and the corresponding shortest path length), while S is to record the vertices for which the shortest path has not been found. And D is to record the distance from the vertex to the starting point).

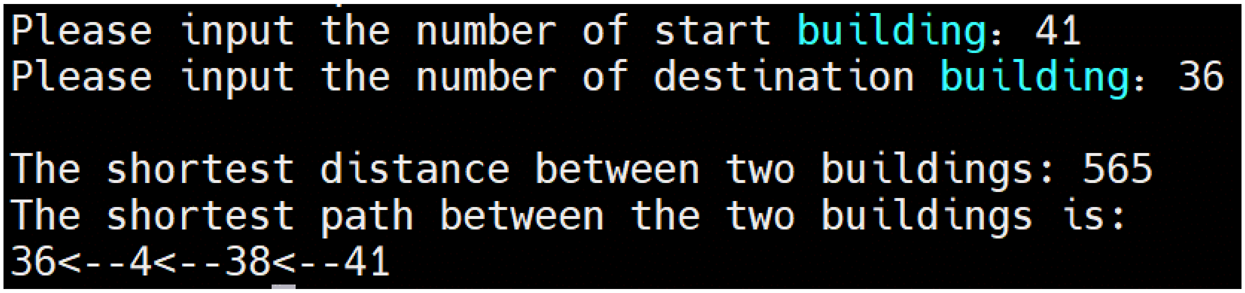
Initially, there is only the starting point ‘start’ in P; there are vertices other than ‘start’ in S, and the path of the vertices in D is "the path from the starting point s to this vertex". Then, find the vertex with the shortest path from S and add it to P; then, update the vertex in S and the path corresponding to the vertex. Then, find the vertex with the shortest path from S and add it to P; then, update the vertex in S and the path corresponding to the vertex. ... repeat this operation until all vertices have been traversed.



```
// Initialization
for (i = 0; i < n; i++)
{
    D[i] = C[vl][i];
    if (D[i] != 10000) // There is a path between i and start
        P[i] = start;
    else
        P[i] = -1;
}
for (i = 0; i < n; i++)
    S[i] = 0;
// Initialize "vertex start" itself
S[vl] = 1;
D[vl] = 0;
// Calendar n - 1 times to find the shortest path of a vertex each time
for (i = 0; i < n - 1; i++)
{
    min = inf;
    // Find the current smallest path
    // Among the vertices for which the shortest path is not obtained, find the vertex k closest to start
    for (j = 0; j < n; j++)
        if ((!S[j]) && (D[j] < min)) // Find a shorter vertice
        {
            // Update most shortest path and vertice
            min = D[j];
            k = j;
        }
    S[k] = 1; // Mark vertex k as having obtained the shortest path
    // Correct the current shortest path and predecessor vertices
    for (j = 0; j < n; j++)
        if ((!S[j]) && (D[j] > D[k] + C[k][j]))
        {
            D[j] = D[k] + C[k][j];
            P[j] = k;
        }
```

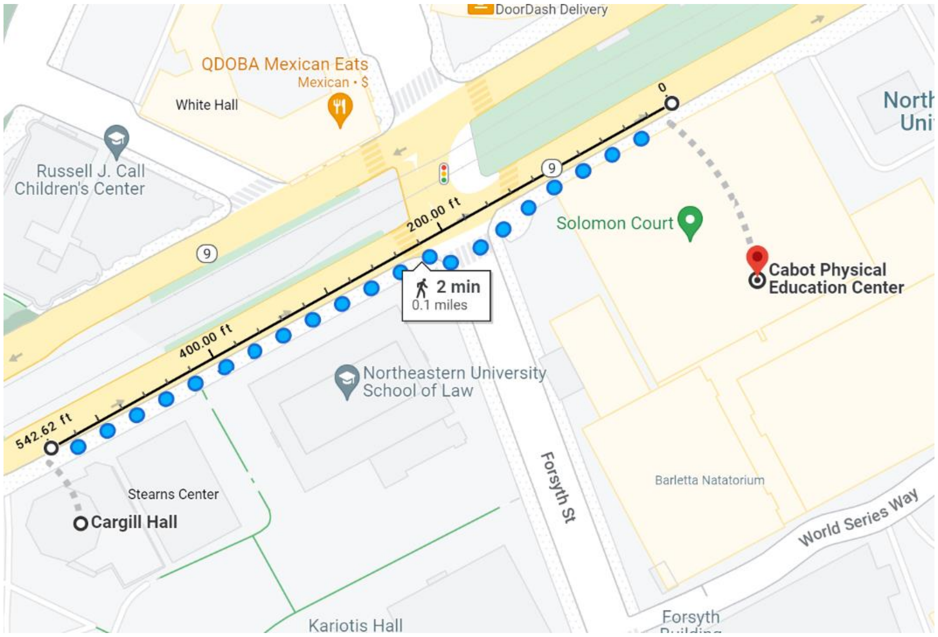
Result

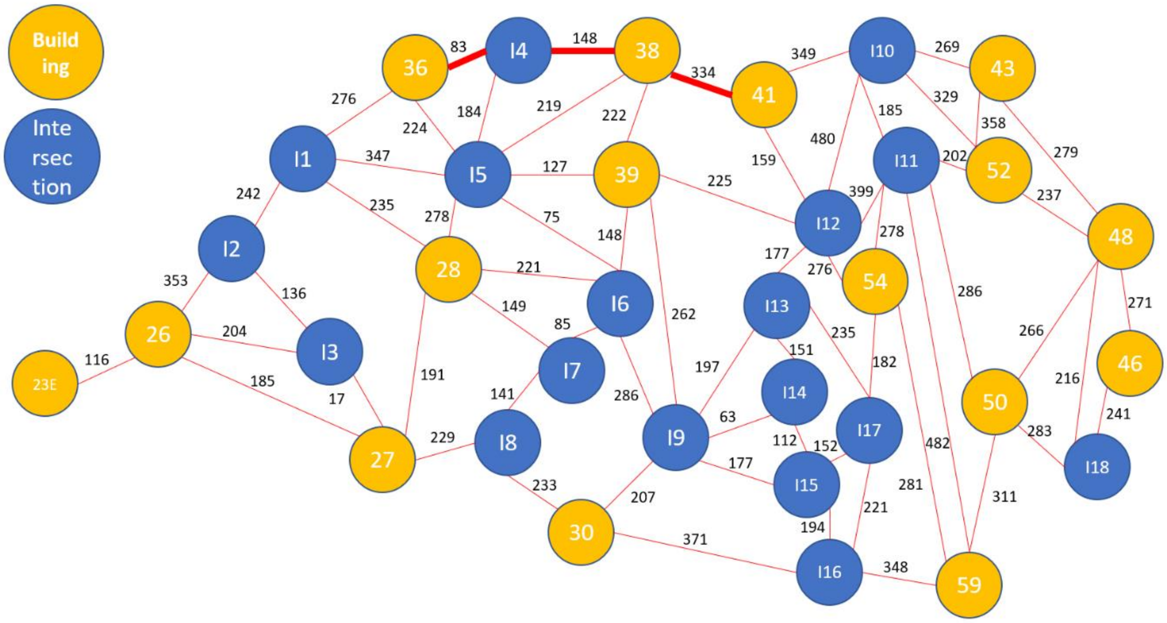
Case1: 41 -> 36



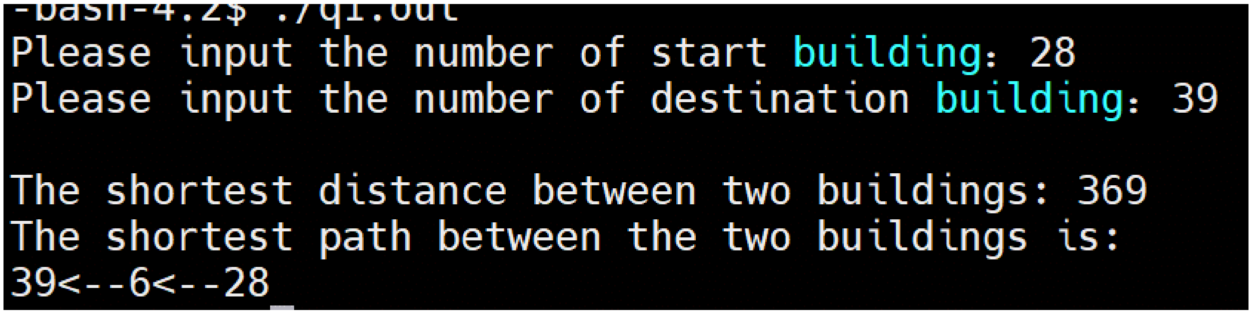
```
Please input the number of start building: 41
Please input the number of destination building: 36

The shortest distance between two buildings: 565
The shortest path between the two buildings is:
36<--4<--38<--41
```


DoorDash Delivery
QDOBA Mexican Eats
Mexican • $
White Hall
Russell J. Call
Children's Center
9
0
200.00 ft
400.00 ft
542.62 ft
2 min
0.1 miles
Solomon Court
Cabot Physical
Education Center
North
Uni
Northeastern University
School of Law
Stearns Center
Cargill Hall
Forsyth St
Barletta Natatorium
World Series Way
Kariotis Hall
Forsyth
Building

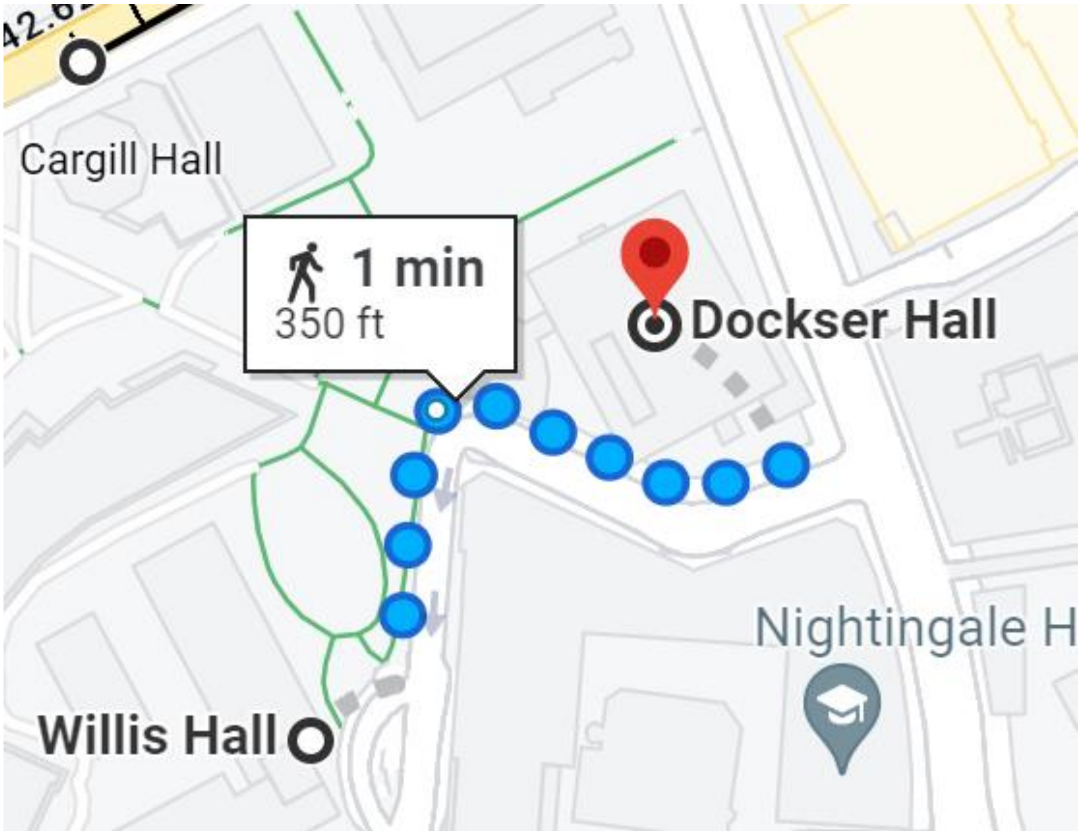

Build
ing
Inte
rsec
tion
36
I4
38
41
I10
43
I1
I5
39
I11
52
48
I2
28
I12
54
I6
I13
26
23E
I3
I7
46
I14
50
I17
I18
27
I8
I9
I15
30
I16
59
83
148
334
349
269
276
184
219
222
224
329
358
185
480
347
127
159
202
279
242
235
278
75
225
399
237
148
278
177
276
353
136
221
149
286
271
204
85
262
235
266
116
185
17
191
197
151
182
216
141
286
63
241
229
112
152
283
482
233
177
281
311
207
221
194
371
348

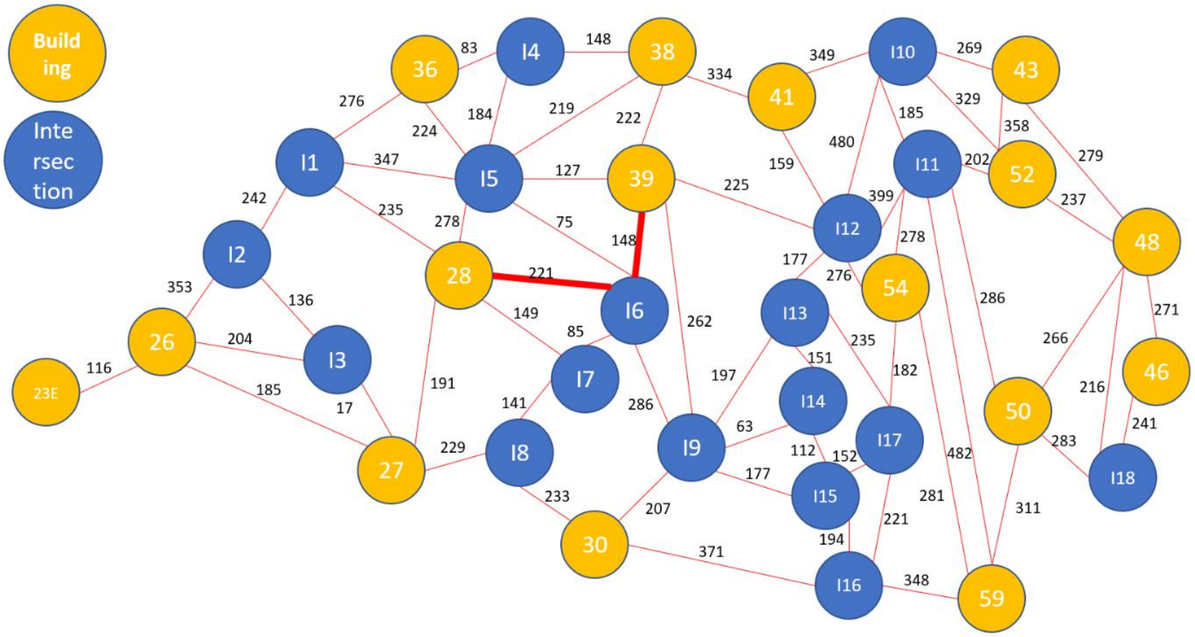
Case2: 28 -> 39



```
-bash-4.2$ ./q1.out
Please input the number of start building: 28
Please input the number of destination building: 39

The shortest distance between two buildings: 369
The shortest path between the two buildings is:
39<--6<--28
```




Building
Intersection
36
83
I4
148
38
334
41
349
I10
269
43
276
224
184
219
222
329
358
I1
347
I5
127
39
225
159
480
185
I11
202
52
279
242
235
278
75
148
399
I12
237
48
I2
28
221
I6
177
276
54
278
286
353
136
149
262
I13
271
26
204
I3
85
I7
235
151
182
266
46
23E
116
185
17
191
141
197
I14
50
216
286
63
I17
241
27
229
I8
I9
112
152
482
283
I18
177
I15
281
311
233
207
221
30
371
194
I16
348
59