Integrated Exercise for Software I Final Review

team_tower
s1250125 Takahisa Watanabe
s1250183 Yuki Homma

1. Overview
2. Development Environment
3. Our Product
4. Demo
5. Contribution
6. Summary

1. Overview

-> Overview

~ About Our Team ~

Team Name: **team_tower** https://github.com/ie03-aizu-2019/ie03project-team_tower

Team Member: **s1250125 Takahisa Watanabe**

s1250183 Yuki Homma

-> Overview

~ System Overview ~

We developed a system to assist **road network construction**.

This system has following functions,

1. Detect all intersection points for given roads
2. Find the shortest distance and its path for given roads
3. Suggest the shortest road to be constructed to link given new locations
4. Detect all highways (in graph theorem, it is called bridge)

-> Overview

~ About Final Phase ~

We expand some outputs for making users understood it with ease.

2. Development Environment

-> Development Environment

~ Programming Language~

C language : We are familiar with C.

~ Editor~

Emacs : We are familiar with Emacs!

~ Version Management System ~

GitHub https://github.com/ie03-aizu-2019/ie03project-team_tower

3. Our Product

Task 1: Implemented
Task 2: Implemented
Tesk 3: Implemented
Tesk 4: Implemented
Task 5 & 6: Not
Task 7 & 8: Implemented(some Error)
Task 9: Implemented
Task 10, 11: Not

Task 1 and Task 2

-> Our Product

Data Structure

~ Task 1, Task 2~

We use the struct to construct points

```
3
4  typedef struct {
5      double x;
6      double y;
7      int roadA;
8      int roadB;
9      int id;
10 } point_t;
11
```

Good Idea for many road's crossing!

The roads which create this point when the roads are crossing.

ID for the point (1 ~ n)

-> Our Product

Algorithm

~ Task 1, Task 2~

Step 1.	If (5) = 0 Else	No intersection To Step 2
Step 2.	Calculate s and t from equation (6)	To Step 3
Step 3.	If 0≦s≦1 かつ 0≦t≦1 Else	Intersection found, To Step 4 No intersection
Step 4.	Calculate x,y coordinates. x is found by substituting s into (1) or (3) y is found by substituting t into (2) or (4)	

$$x = x_{P1} + (x_{Q1} - x_{P1})s \quad (1)$$

$$y = y_{P1} + (y_{Q1} - y_{P1})s \quad (2)$$

$$x = x_{P2} + (x_{Q2} - x_{P2})t \quad (3)$$

$$y = y_{P2} + (y_{Q2} - y_{P2})t \quad (4)$$

$$|A| = (x_{Q1} - x_{P1})(y_{P2} - y_{Q2}) + (x_{Q2} - x_{P2})(y_{Q1} - y_{P1}) \quad (5)$$

$$\begin{Bmatrix} s \\ t \end{Bmatrix} = \frac{1}{|A|} \begin{bmatrix} y_{P2} - y_{Q2} & x_{Q2} - x_{P2} \\ y_{P1} - y_{Q1} & x_{Q1} - x_{P1} \end{bmatrix} \begin{Bmatrix} x_{P2} - x_{P1} \\ y_{P2} - y_{P1} \end{Bmatrix} \quad (6)$$

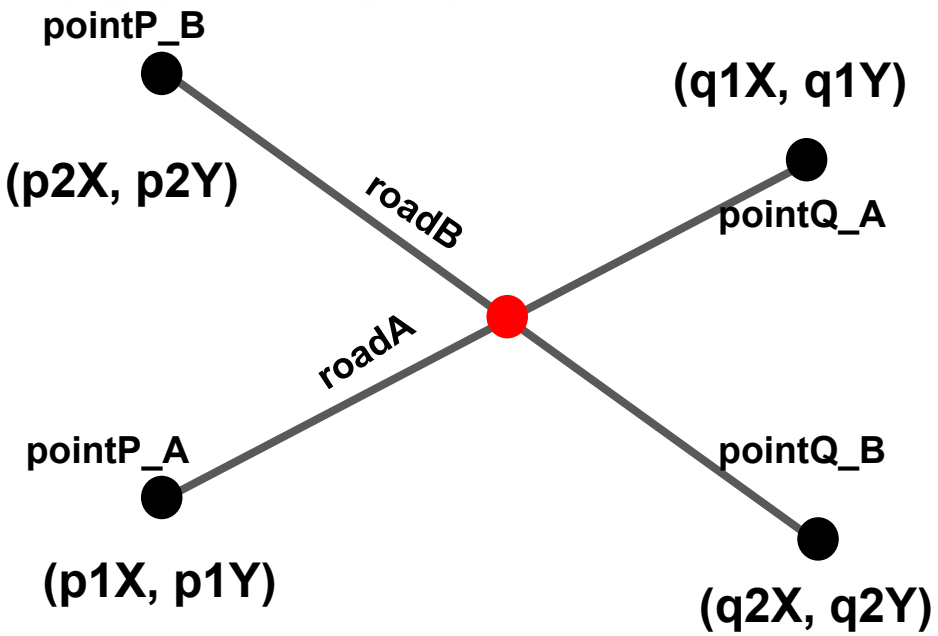
-> Our Product

Source Code

~ Task 1, Task 2~

```
// 交差地点を返す関数，ない場合はNotExist{-1, -1}を返す
point_t detectCrossing(point_t pointP_A, point_t pointQ_A, point_t pointP_B, point_t pointQ_B)
```

```
6   // 交差地点を返す関数，ない場合はNotExist{-1, -1}を返す
7   point_t detectCrossing(point_t pointP_A, point_t pointQ_A, point_t pointP_B, point_t pointQ_B) {
8     double p1X, p1Y, q1X, q1Y, p2X, p2Y, q2X, q2Y;
9     double s, t;
10    double x, y;
11    double determinant;
12    point_t crossing;
13    point_t notExist = {-1, -1};
14    int i, index;
15
16    p1X = pointP_A.x;
17    p1Y = pointP_A.y;
18    q1X = pointQ_A.x;
19    q1Y = pointQ_A.y;
20    p2X = pointP_B.x;
21    p2Y = pointP_B.y;
22    q2X = pointQ_B.x;
23    q2Y = pointQ_B.y;
24
```

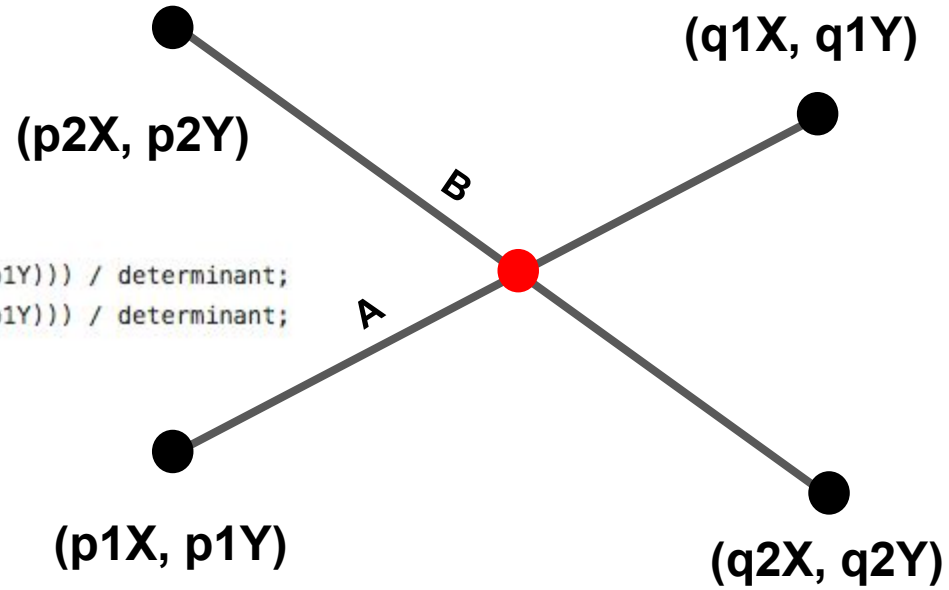


-> Our Product

Source Code

~ Task 1, Task 2~

```
25    // 行列式を求める
26    determinant = fabs(( (q1X - p1X)*(p2Y - q2Y) + (q2X - p2X)*(q1Y - p1Y) ));
27
28    // Step1
29    if( (determinant <= EPS) && (determinant >= EPS) ) {
30      return notExist;
31    }
32
33    // Step2
34    // パラメータを求める
35    s = fabs(((q2Y - p2Y)*(p2X - p1X) - (q2X - p2X)*(p2Y - p1Y))) / determinant;
36    t = fabs(((q1Y - p1Y)*(p2X - p1X) - (q1X - p1X)*(p2Y - p1Y))) / determinant;
37
```



-> Our Product

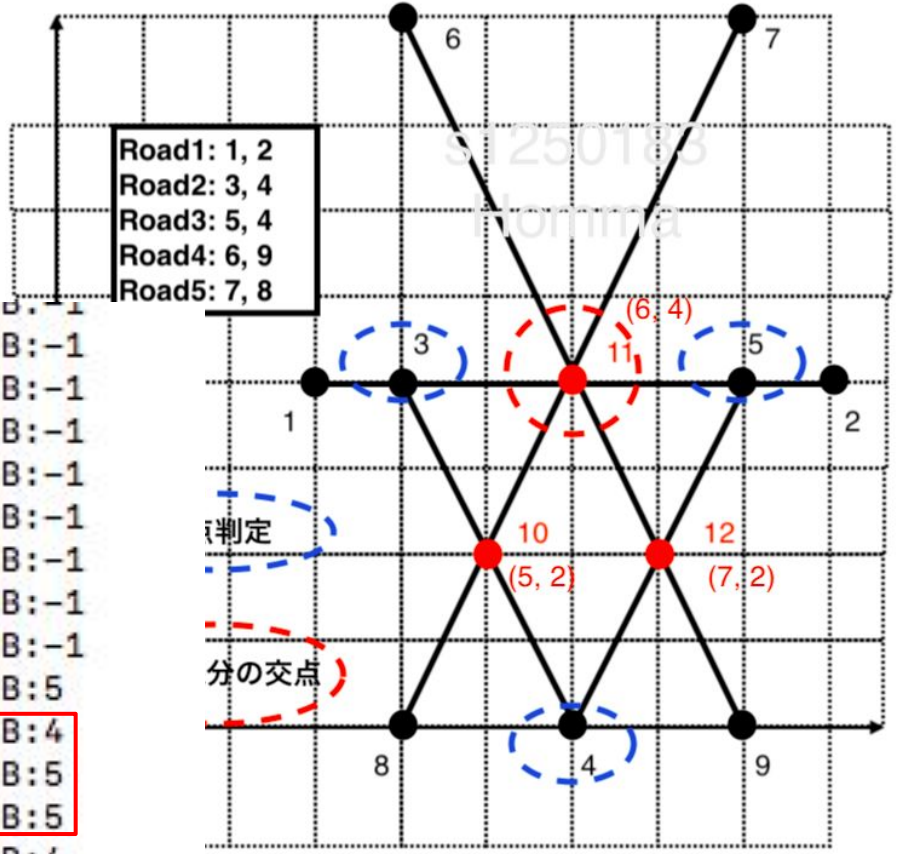
Hard Point

~ Task 1, Task 2~

It's hard to detect the crossing point where many roads are crossing.

Solved

```
point:2 x:9.000000, y:4.0000                    roadB:-1
point:3 x:4.000000, y:4.0000    roadA:-1, roadB:-1
point:4 x:6.000000, y:0.0000    roadA:-1, roadB:-1
point:5 x:8.000000, y:4.0000    roadA:-1, roadB:-1
point:6 x:4.000000, y:8.0000    roadA:-1, roadB:-1
point:7 x:8.000000, y:8.0000    roadA:-1, roadB:-1
point:8 x:4.000000, y:0         roadA:-1, roadB:-1
point:9 x:8.000000, y:          roadA:-1, roadB:-1
point:10 x:5.000000, y:2.000000, roadA:2, roadB:5
point:11 x:6.000000, y:4.000000, roadA:1, roadB:4
point:11 x:6.000000, y:4.000000, roadA:1, roadB:5
point:11 x:6.000000, y:4.000000, roadA:4, roadB:5
point:12 x:7.000000, y:2.000000, roadA:3, roadB:4
```



Task 3 and Task 4

-> Our Product

Algorithm

~ Task 3, Task 4~

Dijkstra's algorithm

For finding the shortest path between nodes

グラフ $G = (V, E)$ 、スタートとなる頂点 s 、および u から v への辺の長さ $\mathrm{length}(u, v)$ を入力として受け取る

// 初期化
各頂点 $v \in V$ に対し、$d(v) \leftarrow (v = s$ ならば 0 、それ以外は $\infty)$
各頂点 $v \in V$ に対し、$\mathrm{prev}(v) \leftarrow$ 「無し」
Q に V の頂点を全て追加

// 本計算
while (Q が空ではない)
  $u \leftarrow Q$ から $d(u)$ が最小である頂点を取り除く
  for each (u からの辺がある各頂点 $v \in V$)
    if ($d(v) > d(u) + \mathrm{length}(u, v)$)
      $d(v) \leftarrow d(u) + \mathrm{length}(u, v)$
      $\mathrm{prev}(v) \leftarrow u$

We do not use priority queue
O(n^2)

Reference from Wikipedia,
https://ja.wikipedia.org/wiki/%E3%83%80%E3%82%A4%E3%82%AF%E3%82%B9%E3%83%88%E3%83%A9%E6%B3%95

-> Our Product

Data Structure

~ Task 3, Task 4~

```
3   // 座標の構造体
4   typedef struct {
5     double x;
6     double y;
7     int roadA;
8     int roadB;
9     double cost;     // コスト
10    int done;     // 訪問済みのフラグ
11    int preNode; // 最短経路として選択された自身の前のノード
12    int id;
13  } point_t;
```

We use the struct of points as a node in graph

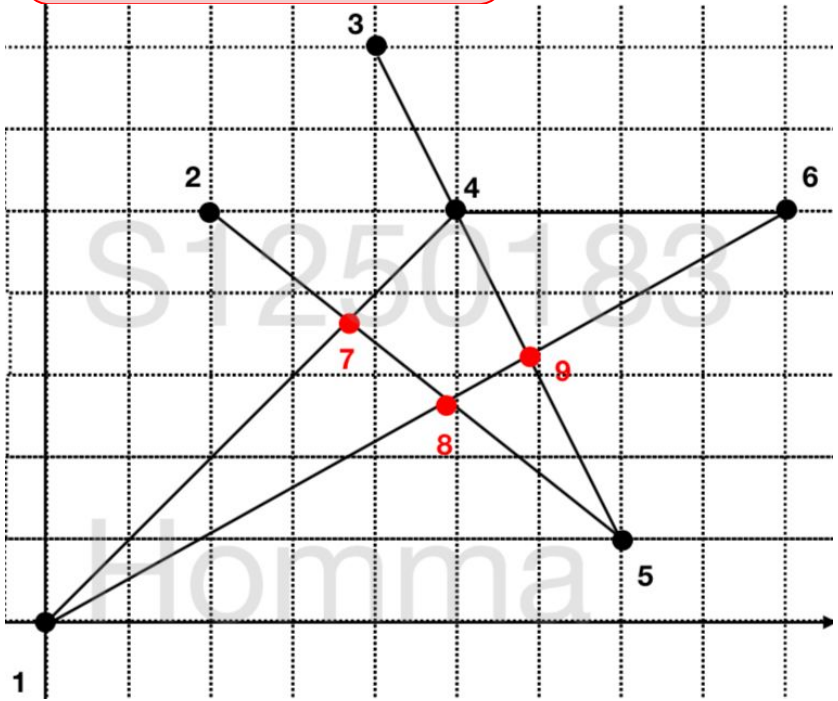
-> Our Product

Data Structure

~ Task 3, Task 4~

Adjacency Matrix

```
double edge[NMAX][NMAX]
```



	1	2	3	4	5	6	7	8	9
1	-1	-1	-1	-1	-1	-1	w	w	-1
2	-1	-1	-1	-1	-1	-1	w	-1	-1
3	-1	-1	-1	w	-1	-1	-1	-1	-1
4	-1	-1	w	-1	-1	w	w	-1	w
5	-1	-1	-1	-1	-1	-1	-1	w	w
6	-1	-1	-1	w	-1	-1	-1	-1	w
7	w	w	-1	w	-1	-1	-1	w	-1
8	w	-1	-1	-1	w	-1	w	-1	w
9	-1	-1	-1	w	w	w	-1	w	-1

-> Our Product

Hard Point

~ Task 3, Task 4 ~

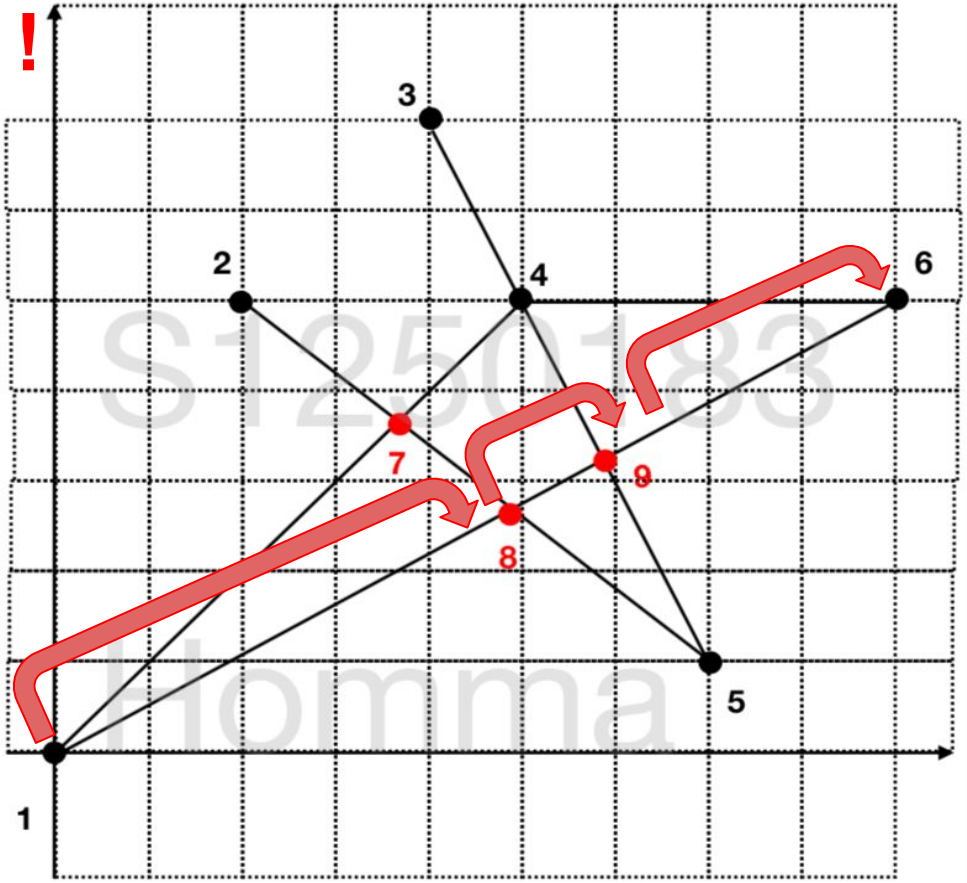
It’s hard to create edge !

Solved

We created the edges one by one for each road.

Crossing points have the id of road which is on. (roadA, roadB)

That makes us possible to create edge with ease.



Task 5 and Task 6

-> Our Product

~ Task 5, Task 6~

Mr. Watanabe will explain.

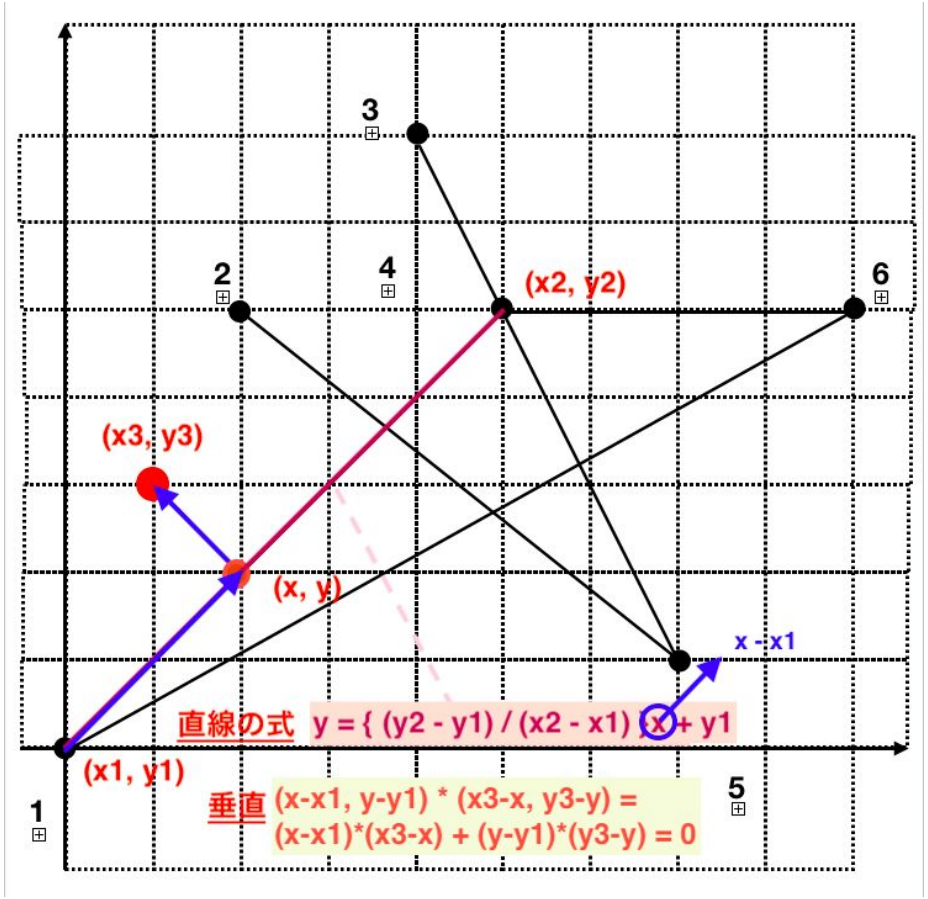
Task 7

-> Our Product

Algorithm

~ Task 7 ~

We use the idea of inner product in vectors



-> Our Product

Algorithm

~ Task 7 ~

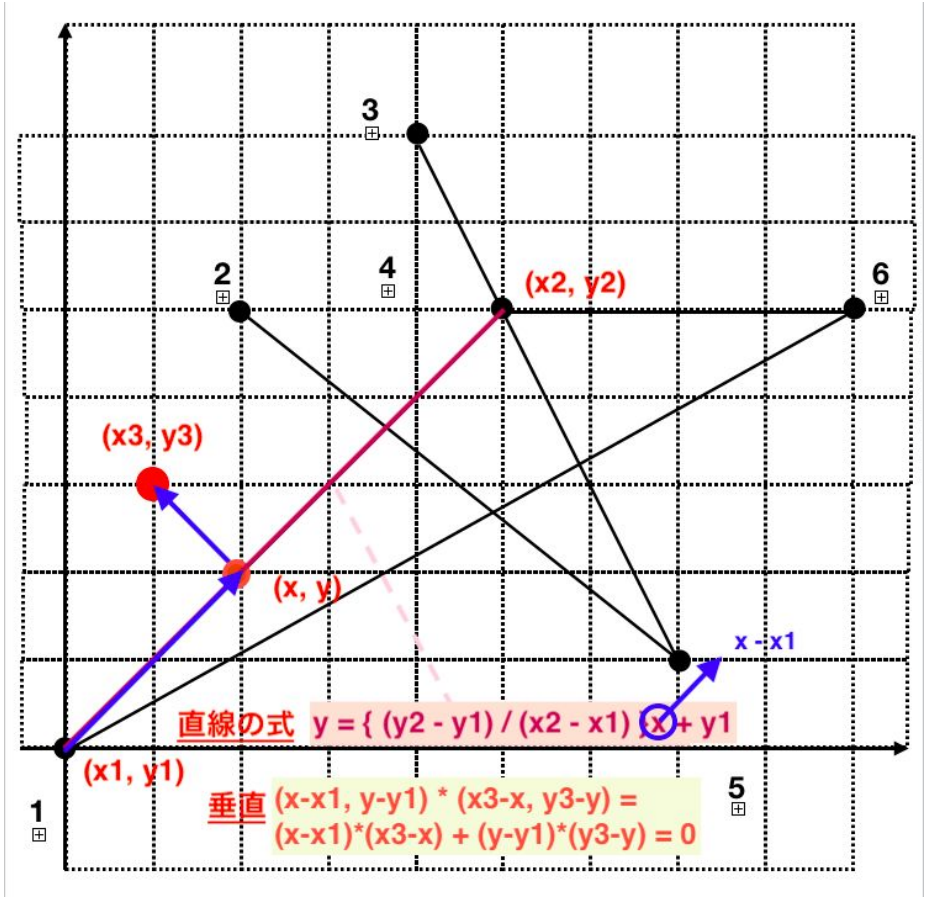
For each road,
we find x and y, such a following conditions

Inner product = 0 AND On the road

And then

The point(x,y) which has the shortest distance from new point

That point is answer



-> Our Product

Algorithm

~ Task 7 ~

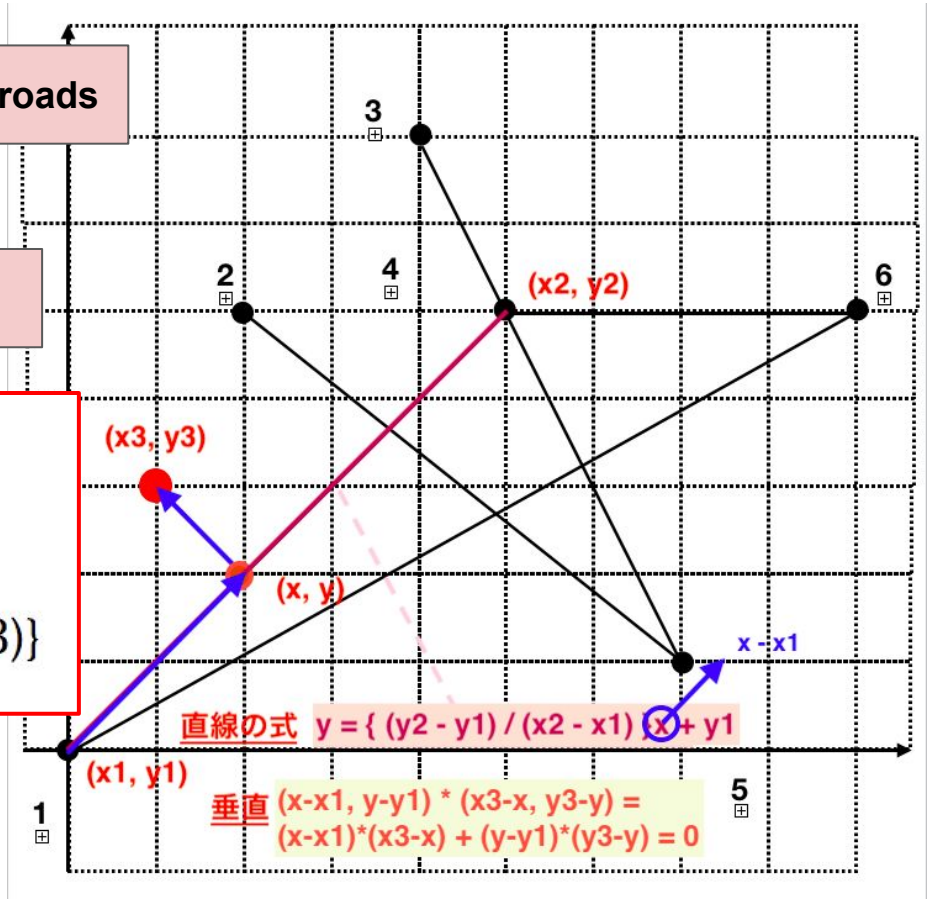
$$y = \frac{(y2 - y1)}{(x2 - x1)}(x - x1) + y1$$

The equation of roads

$$(x - x1)(x3 - x) + (y - y1)(y3 - y) = 0$$

The equation of Inner product

$$x = \frac{1}{(x2 - x1)^2 + (y2 - y1)^2}$$
$$\{(x2 - x1)^2 x3 + (y2 - y1)^2 x1$$
$$- (x2 - x1)(y2 - y1)(y1 - y3)\}$$



-> Our Product

Source Code

~ Task 7 ~

```
46    /*
47     * 全ての道に対して内積を求め、
48     * 内積が0の時の座標と距離を出す(その道までの最短距離)
49     * そのなかで一番距離が小さいものを採用
50     */
51    for(i = 1; i <= m; i++) {
52      pointPIndex = searchPointIndex(point, n, road[i][0]);
53      pointQIndex = searchPointIndex(point, n, road[i][1]);
54      // x1, y1 は 線分端点Pの座標
55      x1 = point[pointPIndex].x;
56      y1 = point[pointPIndex].y;
57      // x2, y2 は 線分端点Qの座標
58      x2 = point[pointQIndex].x;
59      y2 = point[pointQIndex].y;
60
61      coef1 = pow(x2-x1, 2.0);
62      coef2 = pow(y2-y1, 2.0);
63
64      // 方程式を解く
65      x[i] = (1 / (coef1 + coef2)) * ( coef1 * x3 + coef2 * x1 - (x2-x1)*(y2-y1)*(y1-y3) );
66      y[i] = ( ( (y2 - y1)/(x2 - x1) ) * (x[i] - x1) ) + y1;
67      // 距離を求める
68      distance[i] = sqrt(fabs( (x3 - x[i])*(x3 - x[i]) + (y3 - y[i])*(y3 - y[i]) ));
69    }
70
```

-> Our Product

Source Code

~ Task 7 ~

```
78    // 最短距離の座標を返す
79    min = INF;
80    minIndex = 0;
81    for(i = 1; i <= m; i++) {
82      if( (min > distance[i]) && (x[i] != x3) && (y[i] != y3) ) {
83        min = distance[i];
84        minIndex = i;
85      }
86    }
87
88    connectPoint.x = x[minIndex];
89    connectPoint.y = y[minIndex];
90
91    return connectPoint;
92  }
```

Task 8

-> Our Product

Algorithm

~ Task 8 ~

How to find the bridge edge in graph?

1) For every edge (u, v), do following

…..a) Remove (u, v) from graph

…..b) See if the graph remains connected (We can either use BFS or DFS)

We use DFS

…..c) Add (u, v) back to the graph.

Reference from **GeekforGeeks**,
https://www.geeksforgeeks.org/bridge-in-a-graph/

-> Test case generator

We implemented test case generator using C language.

```
n = 100;
m = 99;
p = 100;
q = 100;

printf("%d %d %d %d\n", n, m, p, q);
for(i = 0; i < n; i++) {
  // 0から100000までの乱数を発生
  x = rand() % 100000;
  y = rand() % 100000;
  printf("%d %d\n", x, y);
}

for(i = 0; i < m; i++) {
  // 1から99までの乱数を発生
  pointP = rand() % 99 + 1;
  pointQ = rand() % 99 + 1;

  printf("%d %d\n", pointP, pointQ);
}

for(i = 0; i < q; i++) {
  // 1から100までの乱数を発生
  start = rand() % 100 + 1;
  end = rand() % 100 + 1;

  orCross = rand() % 4;
  if(orCross == 0) {
    printf("%d %d 1\n", start, end);
  } else if(orCross == 1) {
    printf("C%d %d 1\n", start, end);
  } else if(orCross == 2) {
    printf("%d C%d 1\n", start, end);
  } else {
    printf("C%d C%d 1\n", start, end);
  }
}
```

-> Visualizer

No.

4. Demo

Task 2

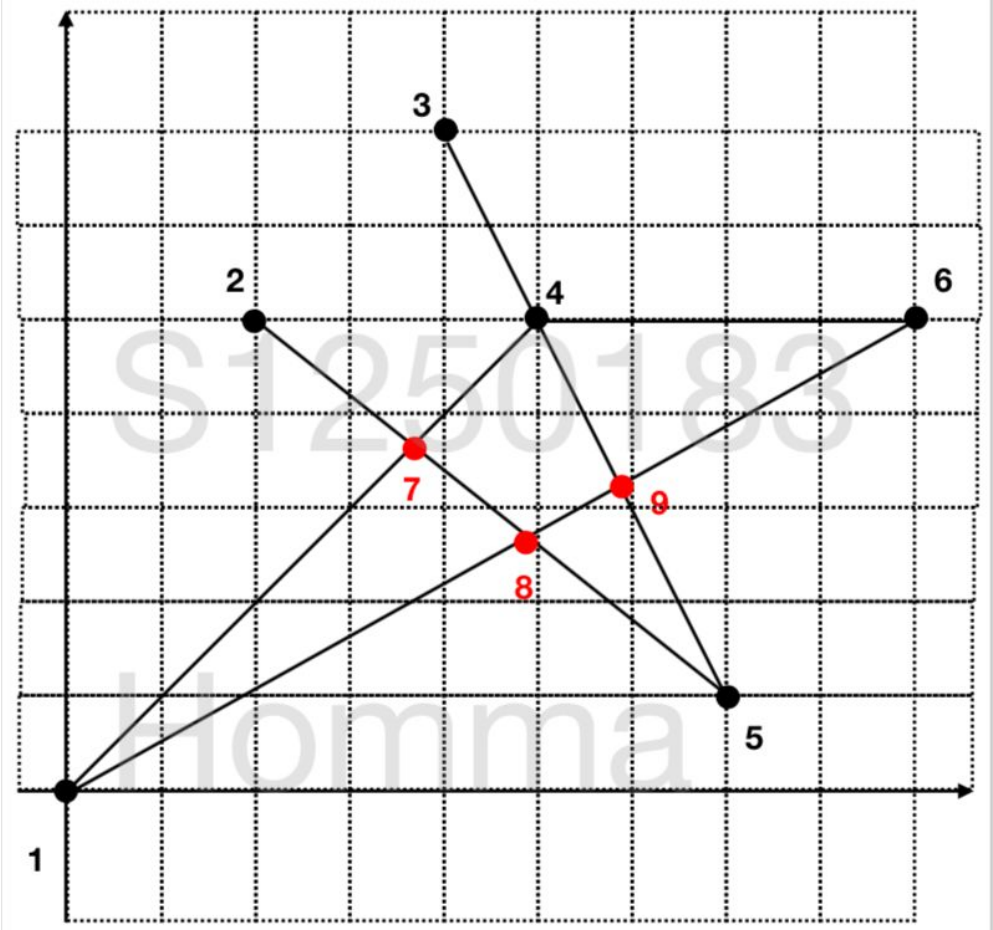
-> Demo

Task 2

Test Case 1

Input
```
6 5 0 0
0 0
2 5
4 7
5 5
7 1
9 5
1 4
1 6
2 5
3 5
4 6
```

Output
```
3.66667 3.66667
4.86885 2.70492
5.86957 3.26087
```



-> Demo

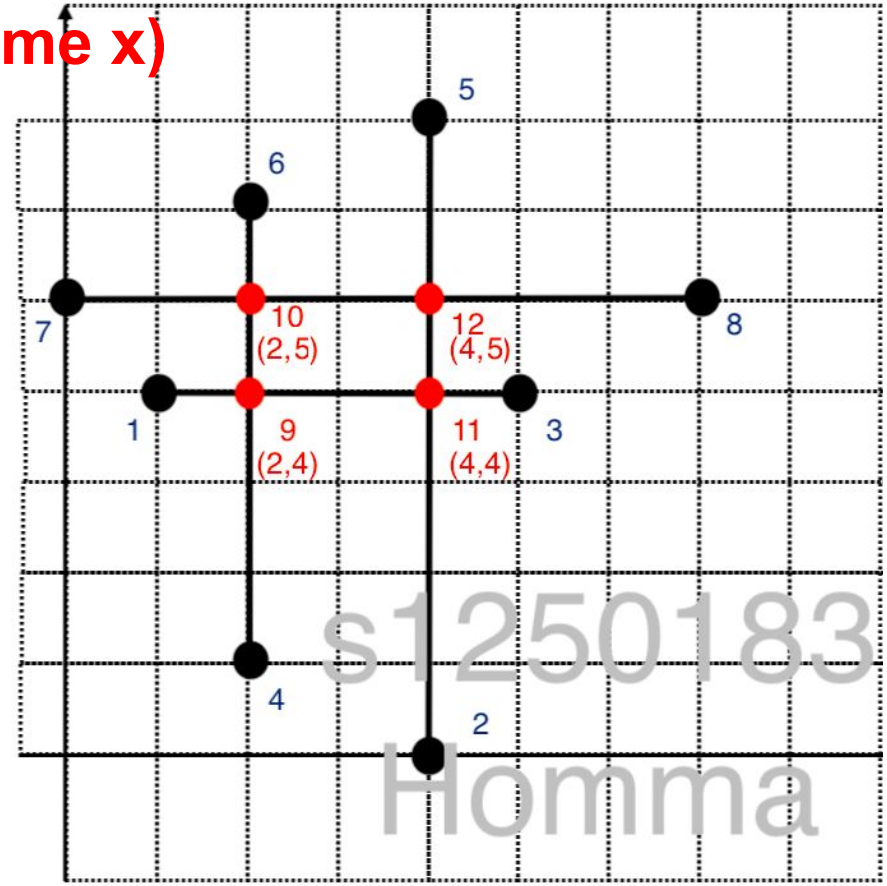
Task 2

Test Case 2(points which has same x)

Input
8 4 0 5
1 4
4 0
5 4
2 1
4 7
2 6
0 5
7 5
1 3
7 8
6 4
5 2

Output

2.000000 4.000000
2.000000 5.000000
4.000000 4.000000
4.000000 5.000000



-> Demo

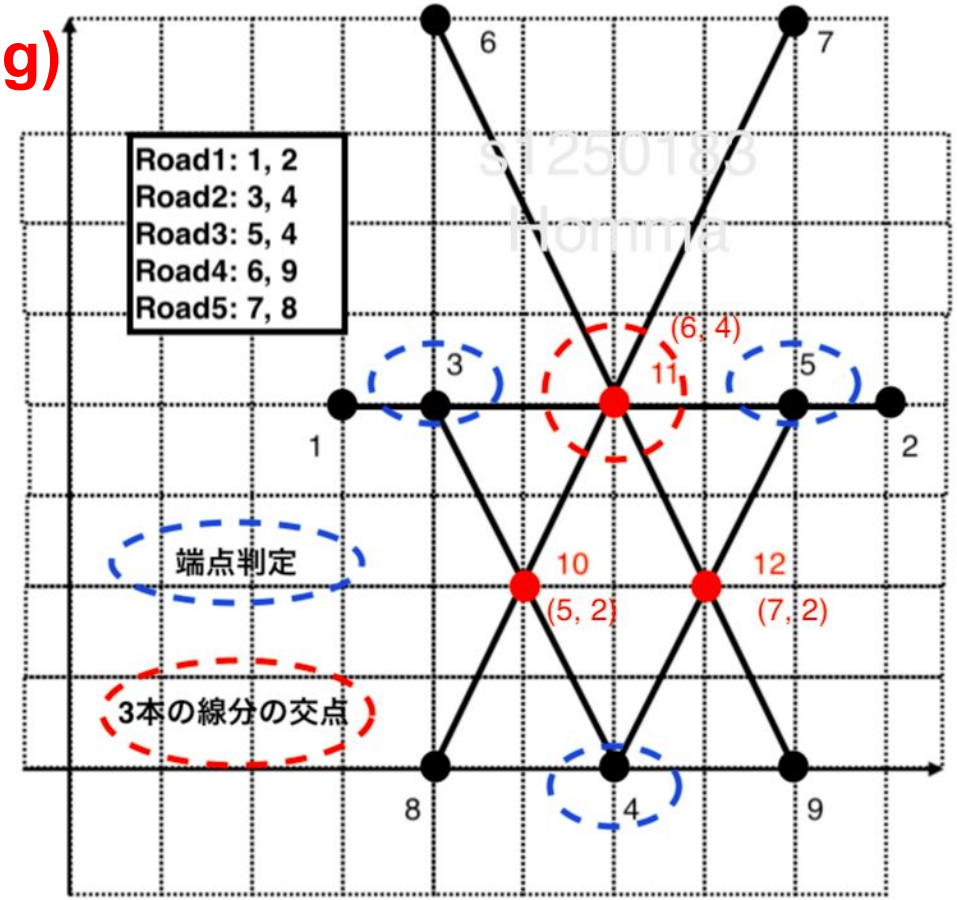
Task 2

Test Case 3(3 roads crossing)

Input
9 5 0 0
3 4
9 4
4 4
6 0
8 4
4 8
8 8
4 0
8 0
1 2
3 4
5 4
6 9
7 8

Output

5.000000 2.000000
6.000000 4.000000
7.000000 2.000000



-> Demo

Task 2

Test Case 4(n = 200 m = 100)

Input
200 100 0 0
148 886
107 87
449 387
609 351
…..
184 59
87 59
152 30
198 73
154 31
150 47
121 99
181 45
52 65

Output

???
(random)

Task 4

-> Demo

Task 4

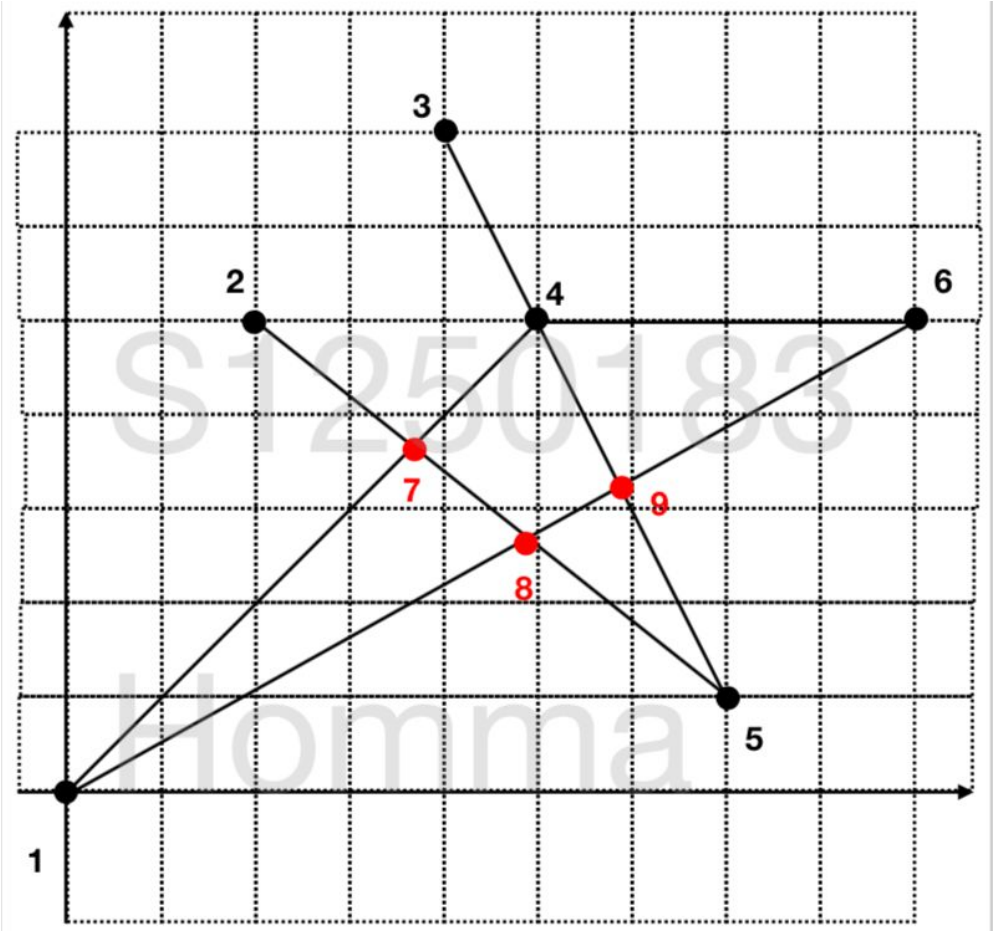
Test Case 1

Input

```
6 5 0 5
0 0
2 5
4 7
5 5
7 1
9 5
1 4
1 6
2 5
3 5
4 6
1 4 1
5 6 1
C1 6 1
C1000 1 1
C1 C3 1
```

Output

```
7.07107
1 C1 4
6.10882
5 C3 6
5.88562
C1 4 6
NA
2.68432
C1 C2 C3
```



-> Demo

Task 4

Test Case 2(multiple Path)

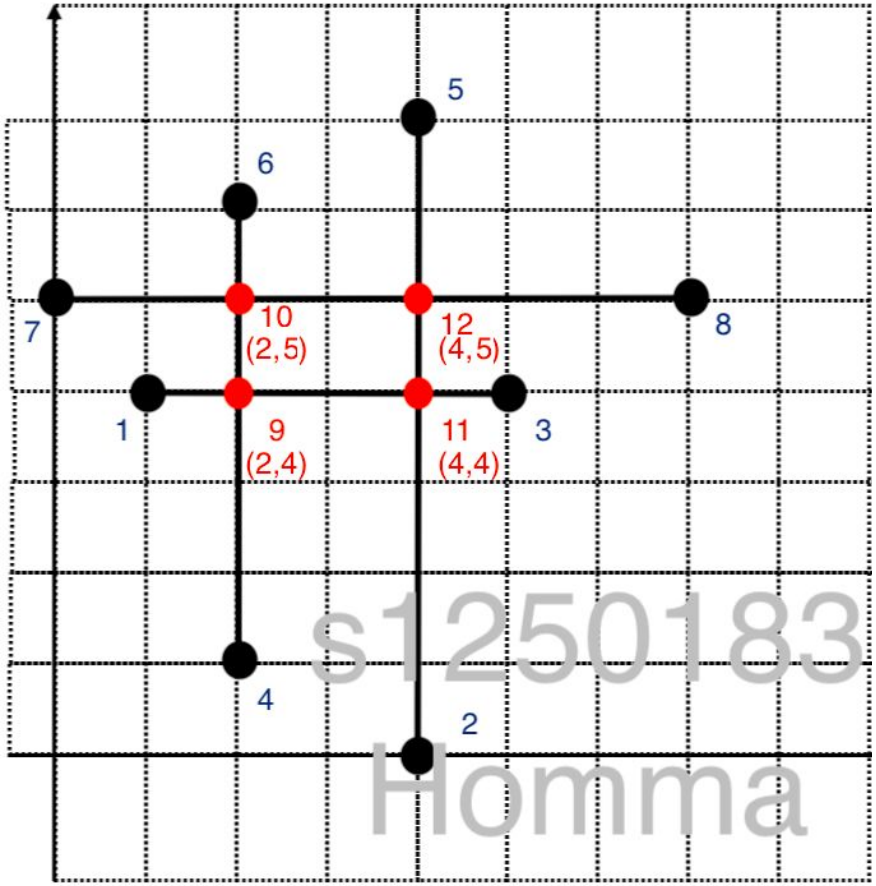
Input

```
8 4 0 5
1 4
4 0
5 4
2 1
4 7
2 6
0 5
7 5
1 3
7 8
6 4
5 2
4 5 1
7 C3 1
4 8 1
2 6 1
1 C1 1
```

Output

Dictionary Order

```
8.000000
4 C1 C3 C4 5  or  4 C1 C2 C4 5
5.000000
7 C2 C4 C3  or  7 C2 C1 C3
9.000000
4 C1 C2 C4 8  or  4 C1 C3 C4 8
8.000000
2 C3 C4 C2 6  or  2 C3 C1 C2 6
1.000000
1 C1
```



-> Demo

Task 4

Test Case 4(n = 200 m = 100)

Input
200 100 0 100
239 296
92 163
148 903
578 669
557 656
322 559
…..
371 895 1
917 C589 1
83 C406 1
99 C760 1
C487 542 1
C124 C903 1

Output

???
(random)

Task 7

-> Demo

Task 7

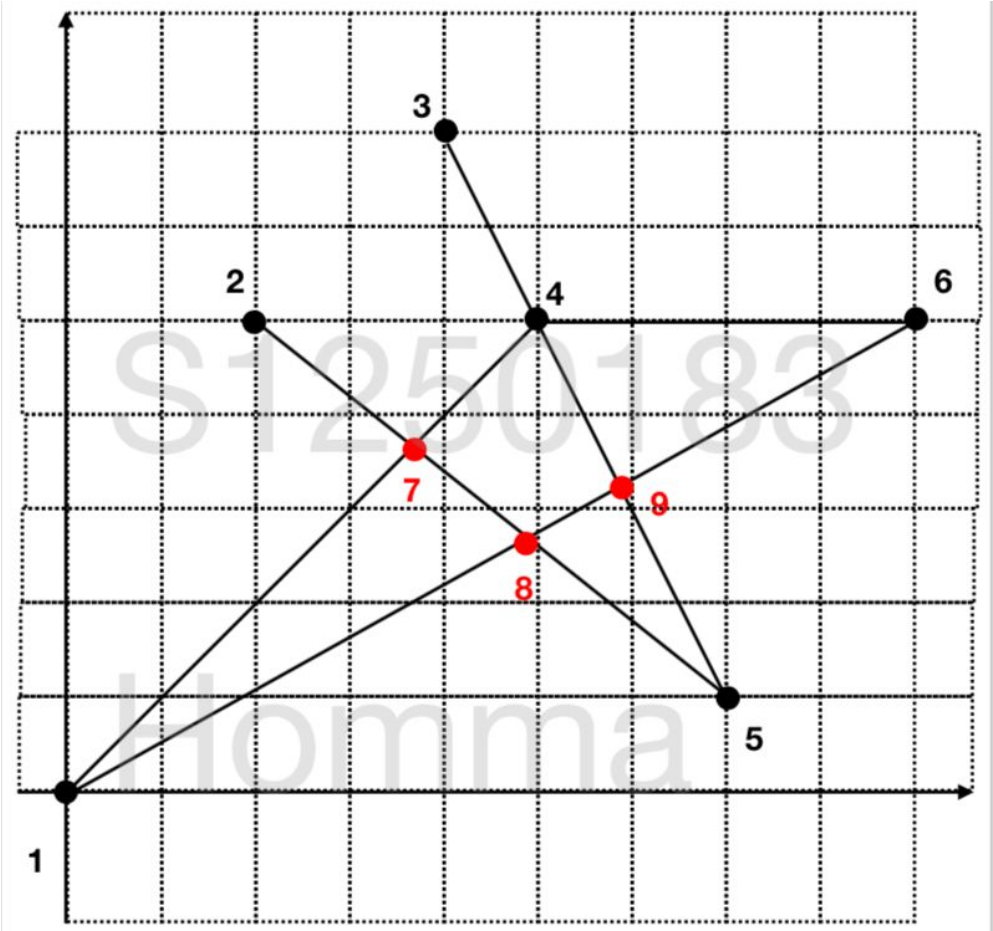
Test Case 1

Input

```
6 5 4 0
0 0
2 5
4 7
5 5
7 1
9 5
1 4
1 6
2 5
3 5
4 6
5 1
11 5
5 4
3 6
```

Output

```
5.78049 1.97561
9 5
5.4 4.2
4.2 6.6
```



Task 8

-> Demo

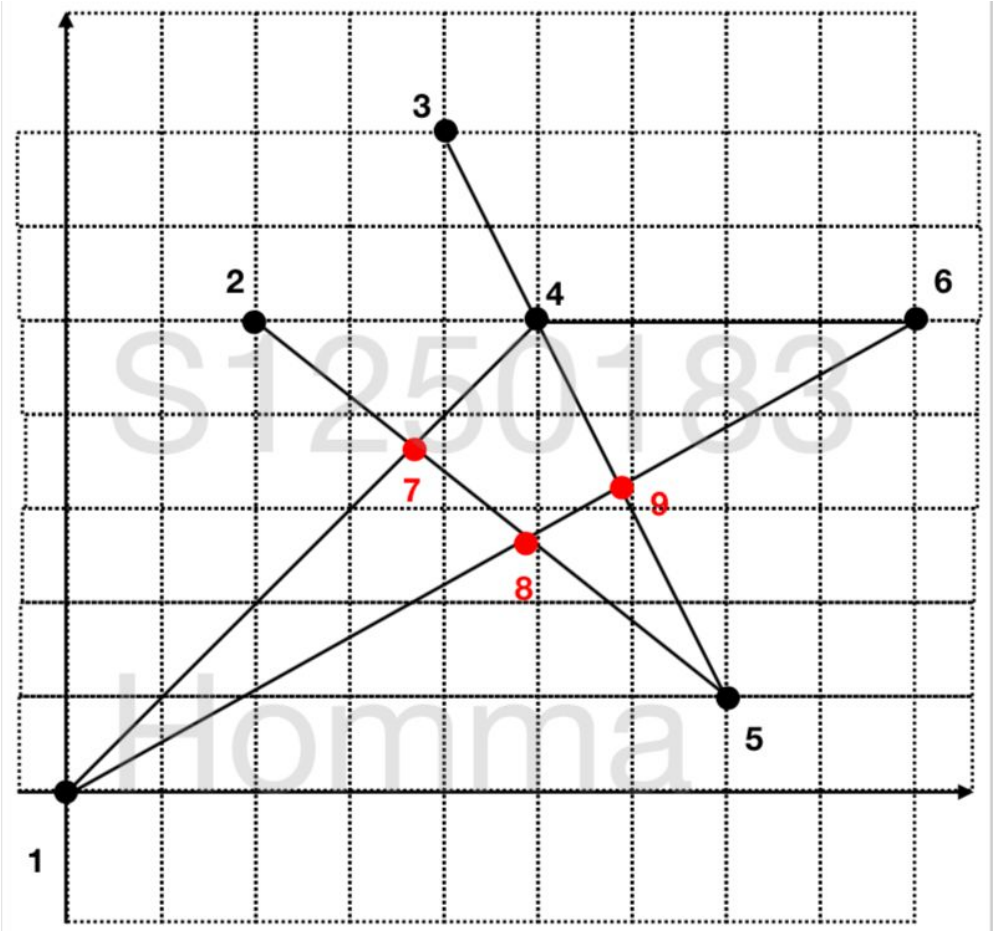
Task 8

Test Case 1

Input
6 5 0 5
0 0
2 5
4 7
5 5
7 1
9 5
1 4
1 6
2 5
3 5
4 6
1 4 1
5 6 1
C1 6 1
C1000 1 1
C1 C3 1

Output

2, 7
3, 9



-> Demo

Test Case 2

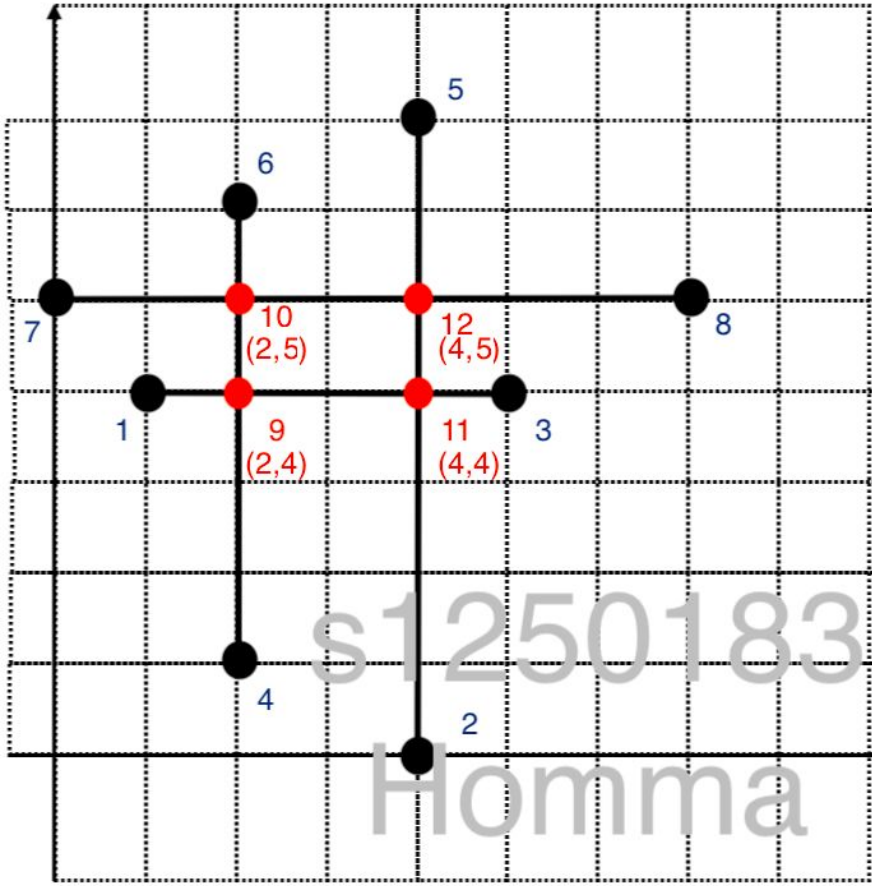
Input

```
8 4 0 5
1 4
4 0
5 4
2 1
4 7
2 6
0 5
7 5
1 3
7 8
6 4
5 2
4 5 1
7 C3 1
4 8 1
2 6 1
1 C1 1
```

Output

```
1, 9
7, 10
6, 10
5, 12
8, 12
3, 11
2, 11
4, 9
```

Task 8



Task 9

-> Demo Test Case 1

Task 9

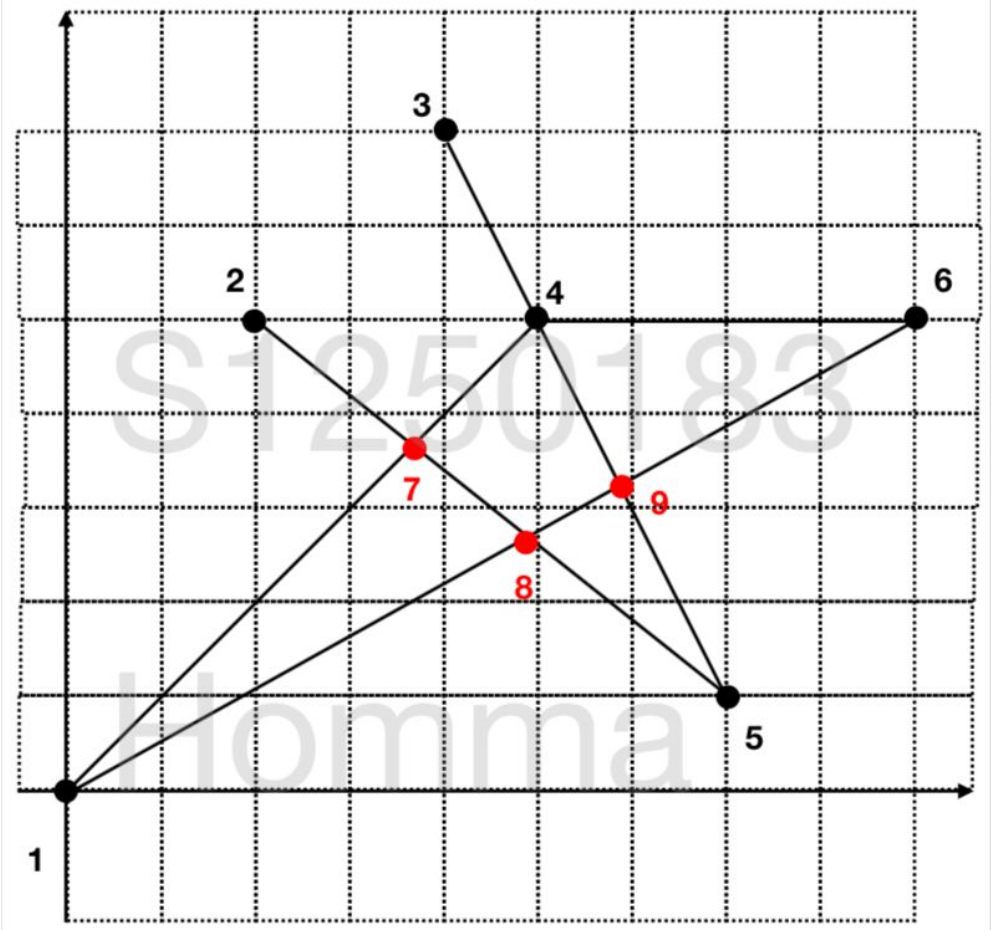
Input
6 5 4 5
0 0
2 5
4 7
5 5
7 1
9 5
1 4
1 6
2 5
3 5
4 6
1 4 1
5 6 1
C1 6 1
C1000 1 1
C1 C3 1
5 1
11 5
5 4
3 6

Output

最短経路
1 C1 4
7.071068
5 C3 6
6.108818
C1 4 6
5.885618
NA
C1 C2 C3
2.684323

最適な道の建設提案
5.780488 1.975610
10.528302 5.849057
5.400000 4.200000
4.200000 6.600000

橋の検出
bridge: 7, 2
bridge: 3, 9



-> Demo

(n = 100 m = 99, p = 100 q = 100)

Task 9

Test Case (n = 200 m = 199, p = 100 q = 100)

Input
100 99 100 100
239 296
92 163
148 903
578 669
557 656
322 559
…..
371 895 1
917 C589 1
83 C406 1
99 C760 1
C487 542 1
C124 C903 1

Output

???
(random)

5. Contribution

-> Member’s Role

s1250125: Takahisa Watanabe

- Task 1, 2 (Input part)
- Task 5, 6 (not working)

s1250183: Yuki Homma

- Task 1, 2 (Detecting crossing part)
- Task 3, 4, 7, 8, 9
- Created the test cases for Task 1, 2, 3, 4, 7, 8, 9
- Created the slides for midterm and final presentation

-> Contribution

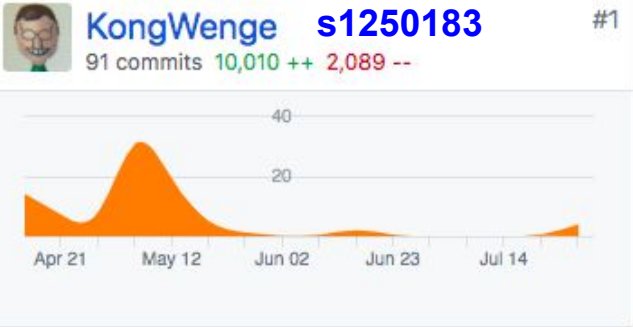
Apr 14, 2019 – Jul 31, 2019

Contributions: Commits

Contributions to master, excluding merge commits

s1250183 Yuki Homma: 80%

s1250125: Takahisa Watanabe: 20%



6. Summary

-> Summary

~ DIfficulty in development system ~

It is hard for us to satisify the constraints on each tasks.

~ What is the best thing in our system ? ~

Our system can detect the point where many roads are crossing.

-> Summary

~ What we learned from this project ~

It is too hard to complete project without corporation.

~ What knowledge do we want to share with other groups? ~

We want to share the knowledge of tools.

Thank you.