Integrated Exercise for Software Ⅰ
Final Presentations

7.31 (wed)

Project Outline

Members Introduction

- Team name : powerpuff
- Members :   s1250110 Mai Kawauchi

  s1250132 Sakurako Kimura

  s1240072 Taiga Takahagi

System introduction (summary)

- In this project, we develop a system to assist road network construction.
- We proposed algorithm for this project.

Dijkstra’s algorithm

Distance calculation

Crossing judgement

Q data judgement

(～midterm)

k-shortest path

new point for shortest

The main road

(～final)

System introduction (summary)

Phese1

- Detection of intersection / Cross detection algorithm in slide
- Detection and management of all intersections / Cross detection algorithm in slide
- Shortest path distance / Dijkstra
- Shortest path/ Dijkstra

Phese2

- K-th shortest path distance / Dijkstra,Yen’s algorithm
- Suggestion of several routes / Dijkstra,Yen’s algorithm
- Suggestion of optimal road construction / Inner product is 0
- Detection of highways / DFS

About the final phase (summary)

We can’t finish final phase, so we don’t add system this project.

Development environment

Development environment

- Platform : Mac, Ubuntu
- language : C++
- tool : emacs 25

Deliverable

Developmental status

- Description of the implemented algorithm
- Overview by class diagram
- Source code (only necessary part)
- Test data
- Data generator / verifier

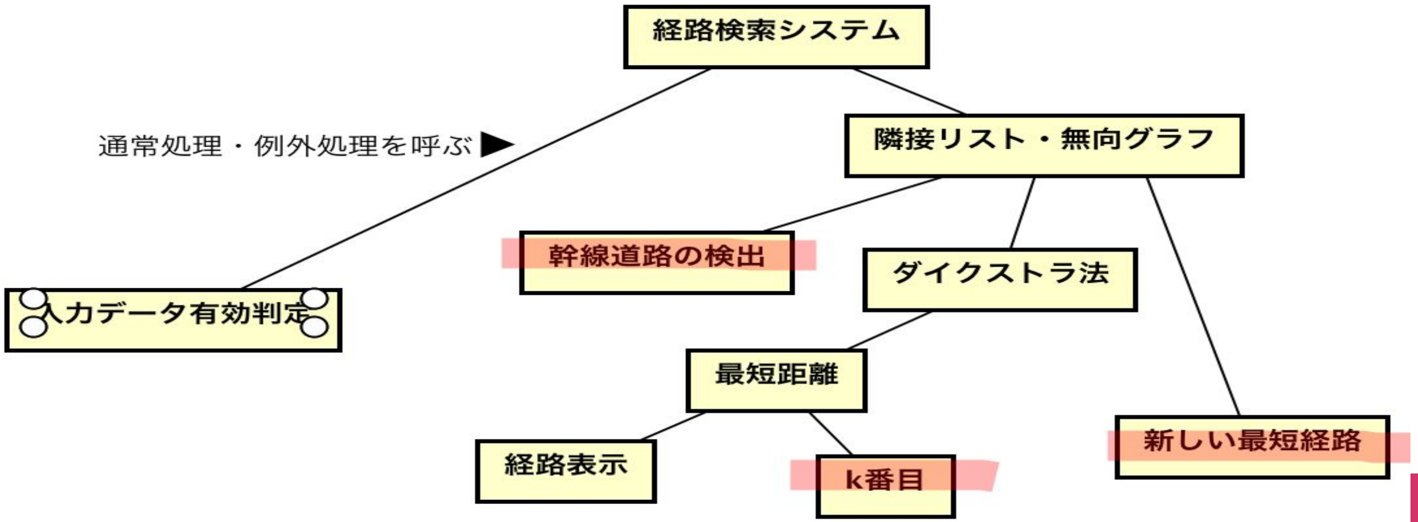
※The final presentation will examine each phase.

Explanation of implemented algorithm / source code

- Correction point (adjacency list creation) ➪ Failure ▲.
- K-shortest path (problem 6) ➪ Failure ×.
- Make new point (problem 7) ➪ Failure ▲.
- Find the main road(problem 8) ➪ Failure.

class

道路建設計画支援システム



通常処理・例外処理を呼ぶ ▶

経路検索システム

隣接リスト・無向グラフ

幹線道路の検出

ダイクストラ法

入力データ有効判定

最短距離

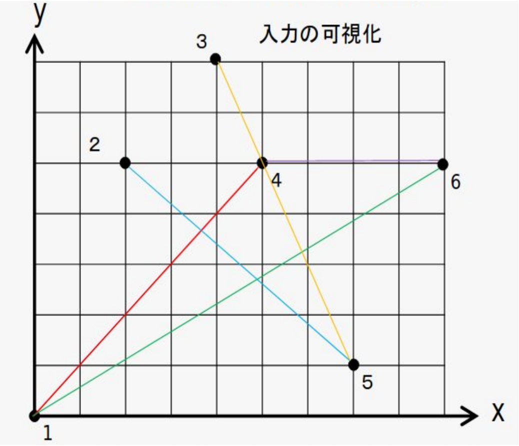
新しい最短経路

経路表示

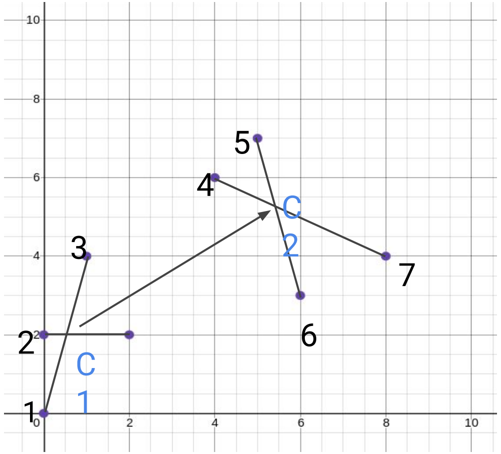
k番目

Midterm Review (Problem)

- Correct adjacency list
  - Not for just Sample Input
  - illegal cross point connection
  - Not cross intersection but divide the line such as "P4"

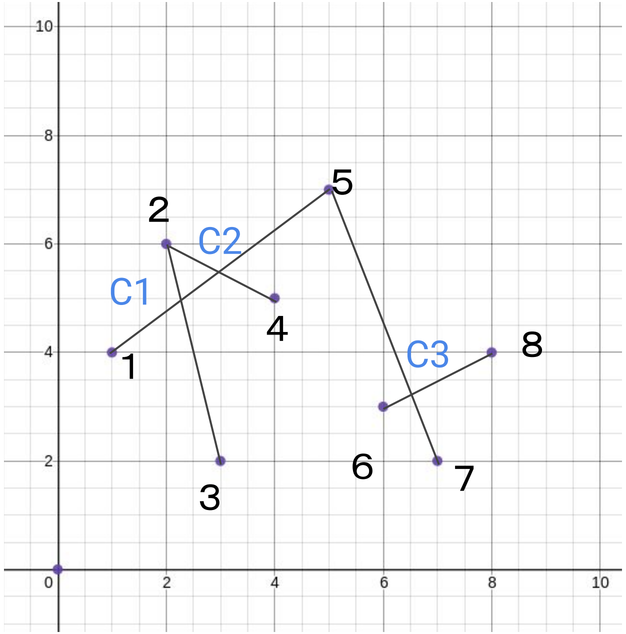


Sorry, Not solute.......



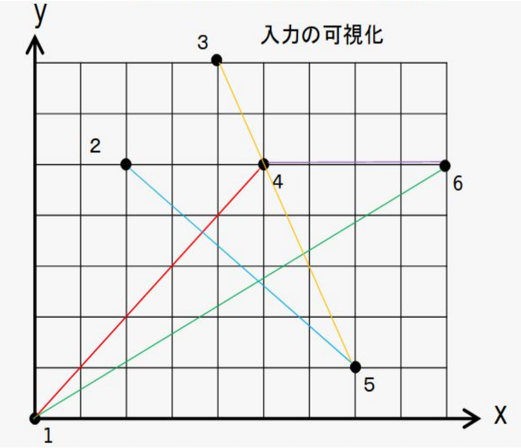
Demonstration(normal case)

1. すべてが最適に交差している



Find the shortest distance
and its route.
Input：　1→C1
　　　　7→C2
　　　　3→4

※Does not include points such as point 4 (the end point is on the line segment)



Ideal Result

1.57895
1 C1
7.88516
7 C3 5 C2
5.07716
3 C1 C2 4

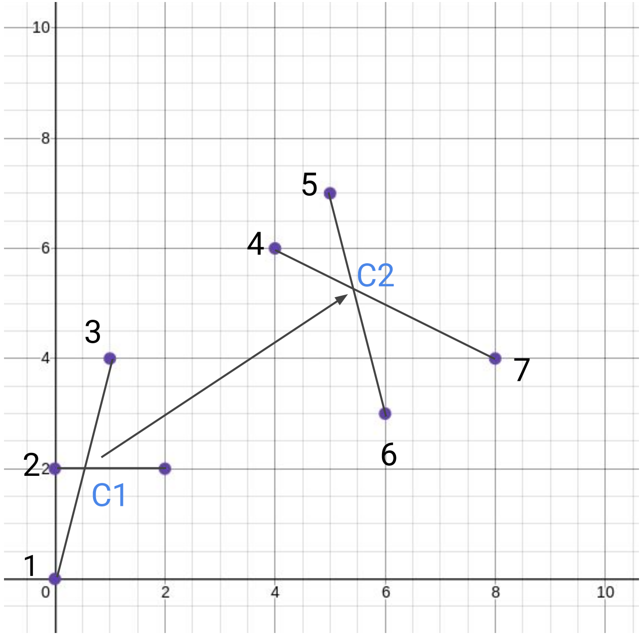
Disagreement

Result

1.57895
1 C1
5.27886
7 C3 C2
5.07716
3 C1 C2 4

Demonstration(exception case)

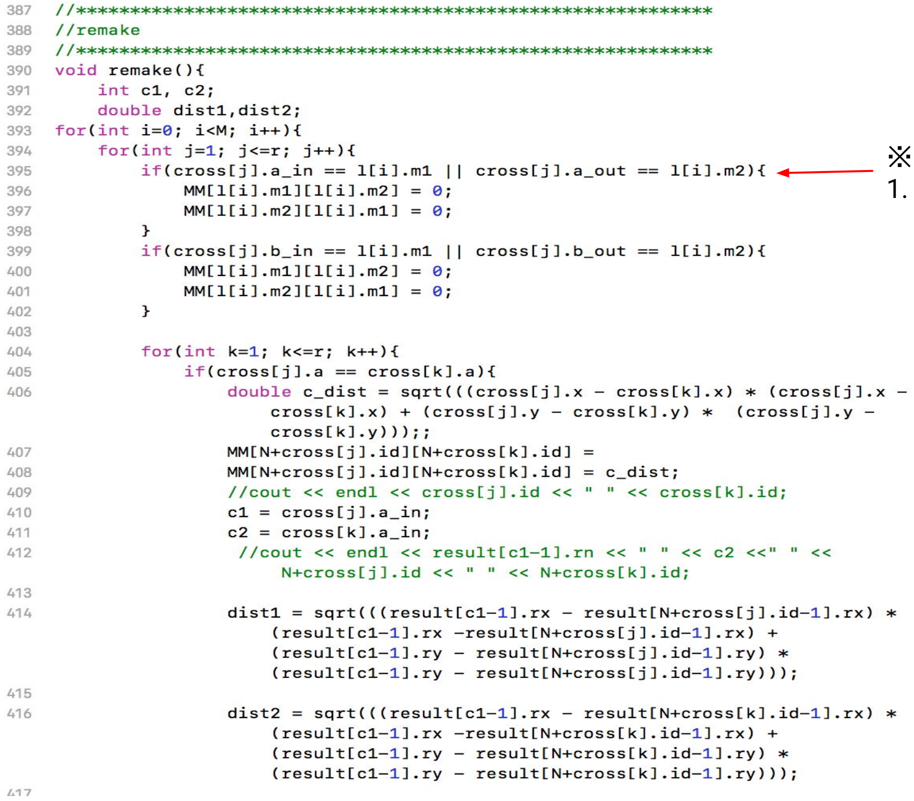
1. There is a point which can not reach each other following a line.



<Problem>

Since the crossing point and the crossing point have been set as sides, it is possible to reach a point that can not be reached.

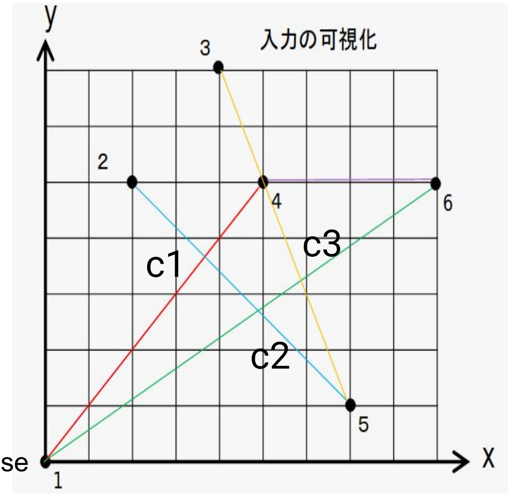
Correction point (adjacency list creation)

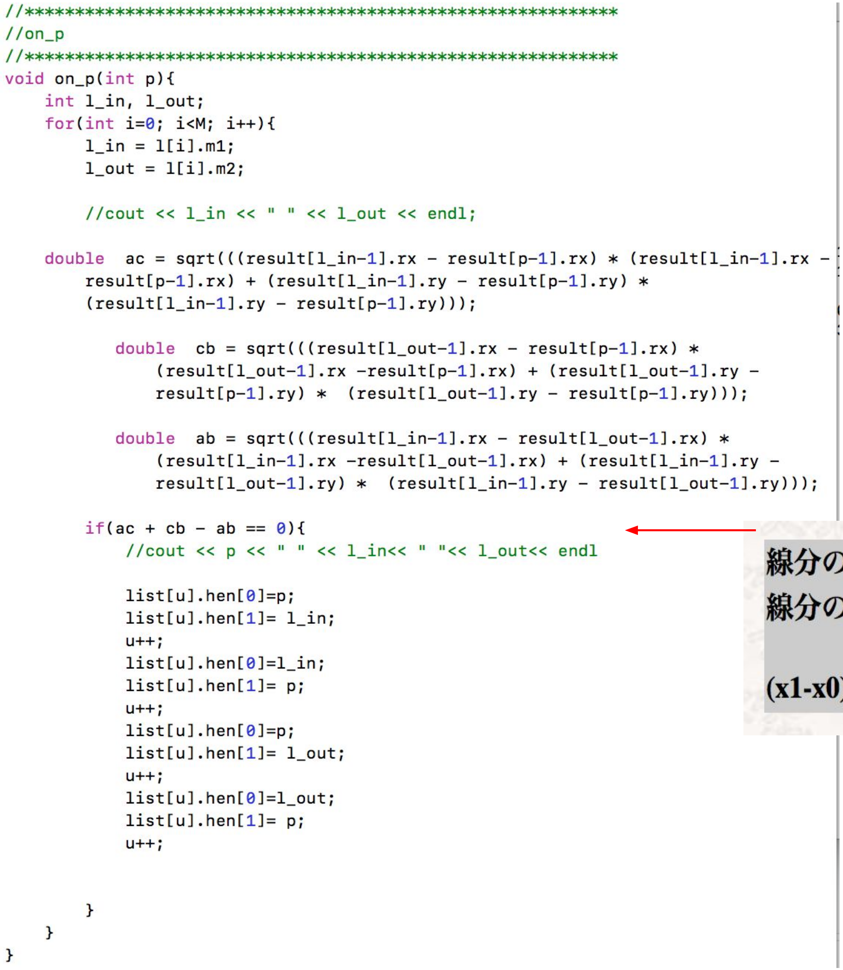


```
//************************************************************
//remake
//************************************************************
void remake(){
    int c1, c2;
    double dist1,dist2;
for(int i=0; i<M; i++){
    for(int j=1; j<=r; j++){
        if(cross[j].a_in == l[i].m1 || cross[j].a_out == l[i].m2){
            MM[l[i].m1][l[i].m2] = 0;
            MM[l[i].m2][l[i].m1] = 0;
        }
        if(cross[j].b_in == l[i].m1 || cross[j].b_out == l[i].m2){
            MM[l[i].m1][l[i].m2] = 0;
            MM[l[i].m2][l[i].m1] = 0;
        }

        for(int k=1; k<=r; k++){
            if(cross[j].a == cross[k].a){
                double c_dist = sqrt(((cross[j].x - cross[k].x) * (cross[j].x -
                    cross[k].x) + (cross[j].y - cross[k].y) *  (cross[j].y -
                    cross[k].y)));;
                MM[N+cross[j].id][N+cross[k].id] =
                MM[N+cross[j].id][N+cross[k].id] = c_dist;
                //cout << endl << cross[j].id << " " << cross[k].id;
                c1 = cross[j].a_in;
                c2 = cross[k].a_in;
                 //cout << endl << result[c1-1].rn << " " << c2 <<" " <<
                     N+cross[j].id << " " << N+cross[k].id;

                dist1 = sqrt(((result[c1-1].rx - result[N+cross[j].id-1].rx) *
                    (result[c1-1].rx -result[N+cross[j].id-1].rx) +
                    (result[c1-1].ry - result[N+cross[j].id-1].ry) *
                    (result[c1-1].ry - result[N+cross[j].id-1].ry)));

                dist2 = sqrt(((result[c1-1].rx - result[N+cross[k].id-1].rx) *
                    (result[c1-1].rx -result[N+cross[k].id-1].rx) +
                    (result[c1-1].ry - result[N+cross[k].id-1].ry) *
                    (result[c1-1].ry - result[N+cross[k].id-1].ry)));
```

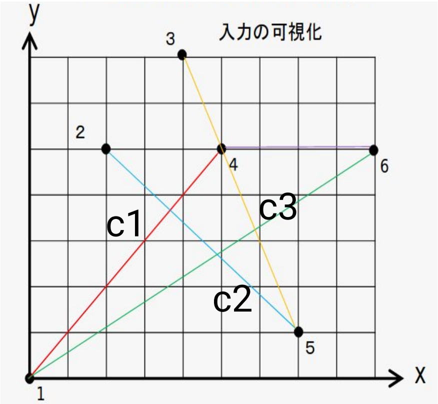
※Line segments have intersections
1. DIST[1][C3] = DIST[1][C2] = 0
(DIST[6][C3] = DIST[6][C2] = 0)

※Only the information
the distance to the end point is close
1. DIST[1][C2] = calc
(DIST[6][C3] = calc)





```
//*************************************************************
//on_p
//*************************************************************
void on_p(int p){
    int l_in, l_out;
    for(int i=0; i<M; i++){
        l_in = l[i].m1;
        l_out = l[i].m2;

        //cout << l_in << " " << l_out << endl;

    double  ac = sqrt(((result[l_in-1].rx - result[p-1].rx) * (result[l_in-1].rx -
        result[p-1].rx) + (result[l_in-1].ry - result[p-1].ry) *
        (result[l_in-1].ry - result[p-1].ry)));

           double  cb = sqrt(((result[l_out-1].rx - result[p-1].rx) *
               (result[l_out-1].rx -result[p-1].rx) + (result[l_out-1].ry -
               result[p-1].ry) *  (result[l_out-1].ry - result[p-1].ry)));

           double  ab = sqrt(((result[l_in-1].rx - result[l_out-1].rx) *
               (result[l_in-1].rx -result[l_out-1].rx) + (result[l_in-1].ry -
               result[l_out-1].ry) *  (result[l_in-1].ry - result[l_out-1].ry)));

        if(ac + cb - ab == 0){
            //cout << p << " " << l_in<< " "<< l_out<< endl

            list[u].hen[0]=p;
            list[u].hen[1]= l_in;
            u++;
            list[u].hen[0]=l_in;
            list[u].hen[1]= p;
            u++;
            list[u].hen[0]=p;
            list[u].hen[1]= l_out;
            u++;
            list[u].hen[0]=l_out;
            list[u].hen[1]= p;
            u++;


        }
    }
}
```



※ it is not an intersection point,
How to divide and culbulate?
⇨ failure

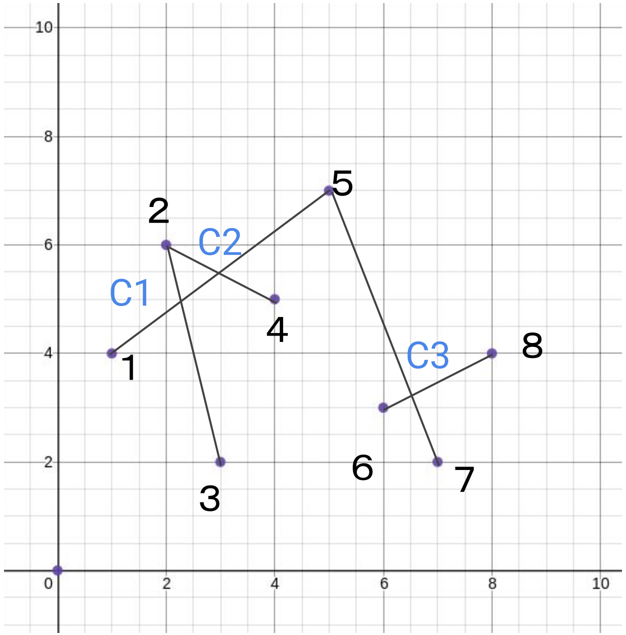
```
0 0 0 0 0 0 5.18545 5.56977 0
0 0 0 0 0 0 2.13437 3.67392 0
0 0 0 0 0 0 0 0 4.18047
0 0 0 0 0 0 1.88562 0 0
0 0 0 0 0 0 4.26875 2.7292 2.52773
0 0 0 0 0 0 4.72586 3.58109
5.18545 2.13437 0 1.88562 4.26875 0 0 1.53955 0
5.56977 3.67392 0 0 2.7292 4.72586 1.53955 0 1.14477
0 0 4.18047 0 2.52773 3.58109  1.14477 0
```

線分の長さ L1 = sqrt((x1-x0)^2 + (y1-y0)^2)
線分の始点から点までの長さ L2 = sqrt((x2-x0)^2 + (y2-y0)^2)

(x1-x0)*(x2-x0) + (y1-y0)*(y2-y0) が L1*L2 に等しく、かつL1≧L2の時衝突している

× distance including the end point on the line

another case



```
0 0 0 0 0 0 0 0 1.57895 0 0
0 0 0 0 0 0 0 0 1.08503 1.11803 0
0 0 0 0 0 0 0 0 3.03808 0 0
0 0 0 0 0 0 0 0 0 1.11803 0
0 0 0 0 0 0 0 0 0 2.5 4.03887
0 0 0 0 0 0 0 0 0 0 0.559017
0 0 0 0 0 0 0 0 0 0 1.34629
0 0 0 0 0 0 0 0 0 0 1.67705
1.57895 1.08503 3.03808 0 0 0 0 0 0 0.921053 0
0 1.11803 0 1.11803 2.5 0 0 0 0.921053 0 0
0 0 0 0 4.03887 0.559017 1.34629 1.67705 0 0 0
```

1.57895
1 C1

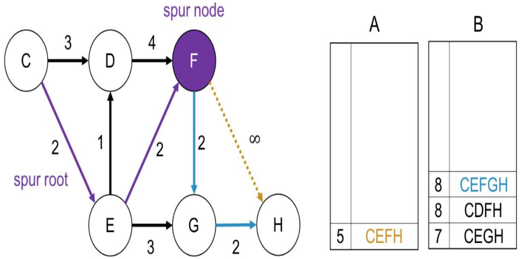
7.88516
7 C3 5 C2

5.07716
3 C1 C2 4

Ideal Result

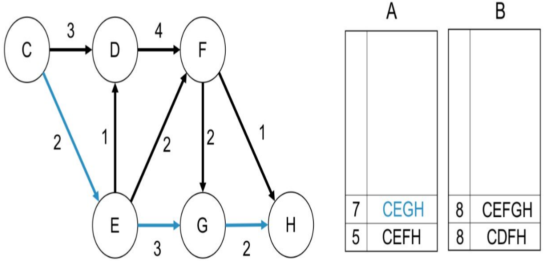
1.57895
1 C1
7.88516
7 C3 5 C2
5.07716
3 C1 C2 4

K-shortest path (problem 6)



A	
5	CEFH

B	
8	CEFGH
8	CDFH
7	CEGH



A	
7	CEGH
5	CEFH

B	
8	CEFGH
8	CDFH

× save, transit
super node or route

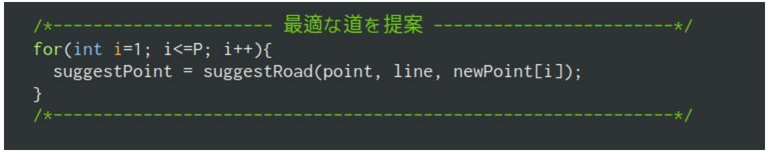
Based on the n-th shortest path, store the candidate of the (n + 1) -th shortest path in the array B while shifting spur-root and spur-node

From the array B, the most appropriate path is taken out and stored in the array A.

yen's algorithm

Make new point (problem 7)

main



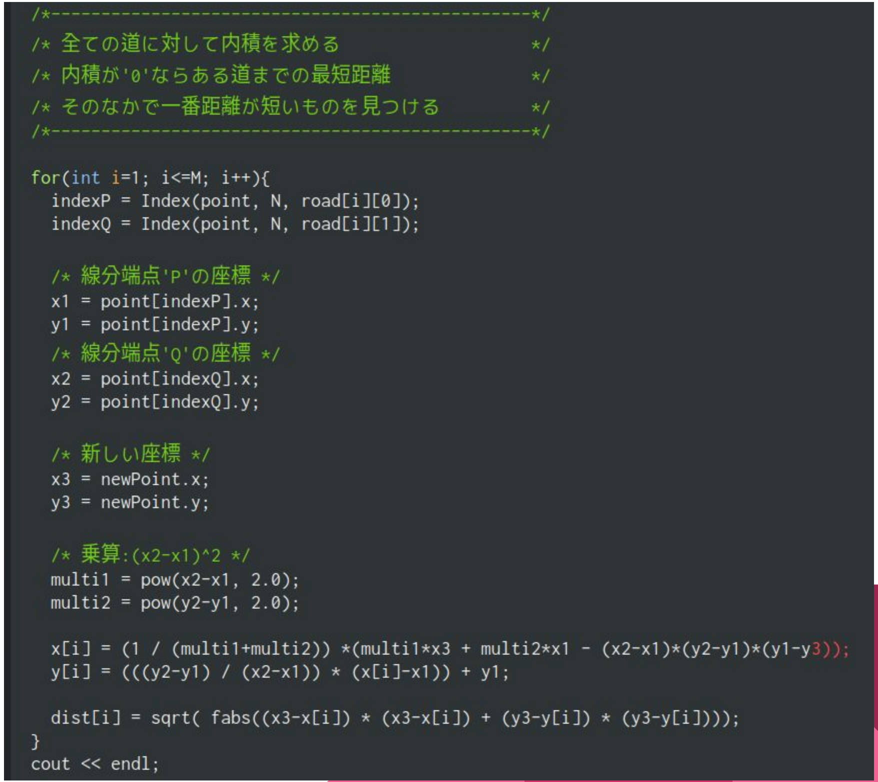
```
/*----------------------- 最適な道を提案 -------------------------*/
for(int i=1; i<=P; i++){
  suggestPoint = suggestRoad(point, line, newPoint[i]);
}
/*---------------------------------------------------------------*/
```

Add_Point suggestRoad(Add_Point *point, int road[][2], Add_Point newPoint)

This is a function that asks where the path of least rumor can be made.

Find the inner product for all roads. If the inner product is '0', it is show that it is the shortest distance to a certain road. Find the one with the shortest distance among them.

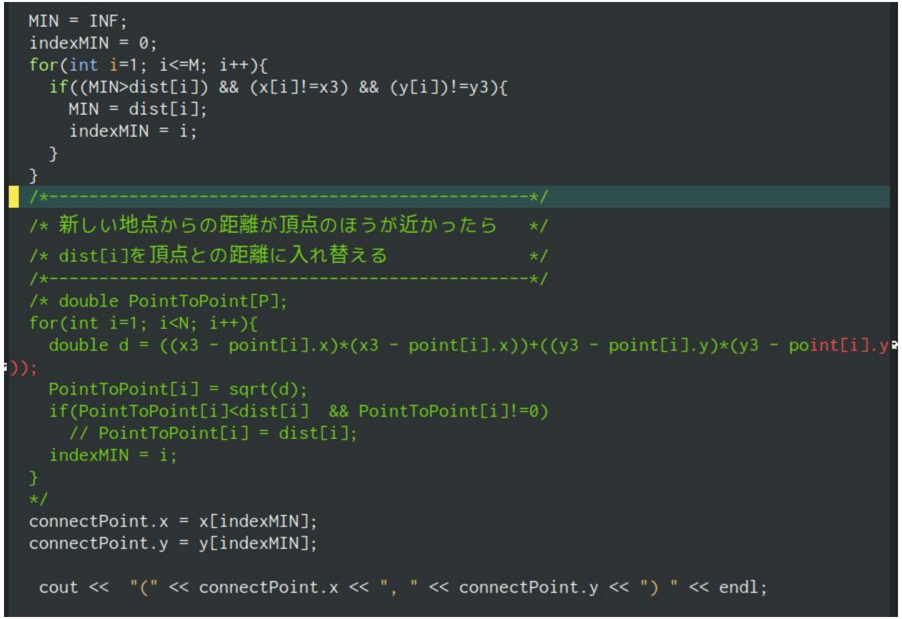
suggestRoad



```
/*--------------------------------------------*/
/* 全ての道に対して内積を求める                 */
/* 内積が'0'ならある道までの最短距離            */
/* そのなかで一番距離が短いものを見つける       */
/*--------------------------------------------*/

for(int i=1; i<=M; i++){
  indexP = Index(point, N, road[i][0]);
  indexQ = Index(point, N, road[i][1]);

  /* 線分端点'P'の座標 */
  x1 = point[indexP].x;
  y1 = point[indexP].y;
  /* 線分端点'Q'の座標 */
  x2 = point[indexQ].x;
  y2 = point[indexQ].y;

  /* 新しい座標 */
  x3 = newPoint.x;
  y3 = newPoint.y;

  /* 乗算:(x2-x1)^2 */
  multi1 = pow(x2-x1, 2.0);
  multi2 = pow(y2-y1, 2.0);

  x[i] = (1 / (multi1+multi2)) *(multi1*x3 + multi2*x1 - (x2-x1)*(y2-y1)*(y1-y3));
  y[i] = (((y2-y1) / (x2-x1)) * (x[i]-x1)) + y1;

  dist[i] = sqrt( fabs((x3-x[i]) * (x3-x[i]) + (y3-y[i]) * (y3-y[i])));
}
cout << endl;
```

Make new point (problem 7)

continuation of A



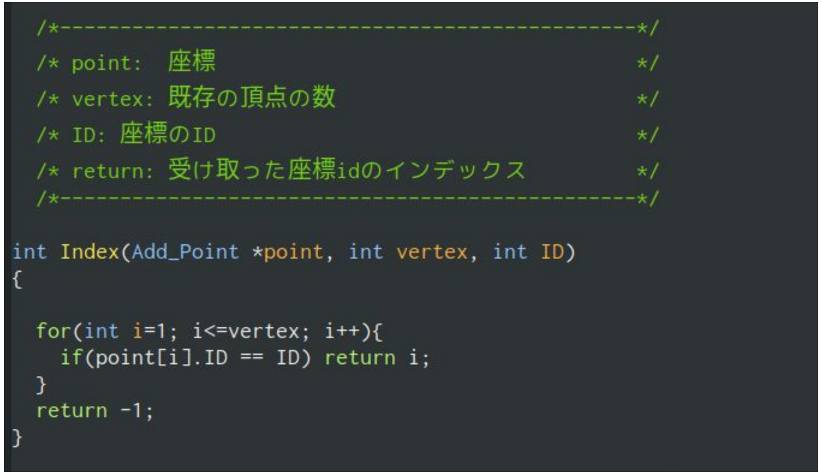
```
MIN = INF;
indexMIN = 0;
for(int i=1; i<=M; i++){
  if((MIN>dist[i]) && (x[i]!=x3) && (y[i])!=y3){
    MIN = dist[i];
    indexMIN = i;
  }
}
/*-----------------------------------------------*/
/* 新しい地点からの距離が頂点のほうが近かったら   */
/* dist[i]を頂点との距離に入れ替える              */
/*-----------------------------------------------*/
/* double PointToPoint[P];
for(int i=1; i<N; i++){
  double d = ((x3 - point[i].x)*(x3 - point[i].x))+((y3 - point[i].y)*(y3 - point[i].y
));
  PointToPoint[i] = sqrt(d);
  if(PointToPoint[i]<dist[i]  && PointToPoint[i]!=0)
    // PointToPoint[i] = dist[i];
  indexMIN = i;
}
*/
connectPoint.x = x[indexMIN];
connectPoint.y = y[indexMIN];

 cout <<  "(" << connectPoint.x << ", " << connectPoint.y << ") " << endl;
```

We succeeded in outputting the shortest distance from the new point to the existing line. However, we did not consider the case where the point (vertex) is the shortest.

When a point(vertex) was near, I decided to write a code to overwrite dist [i]. But I could not.

Make new point (problem 7)

Index



```
  /*--------------------------------------------------*/
  /* point:   座標                                     */
  /* vertex: 既存の頂点の数                            */
  /* ID: 座標のID                                      */
  /* return: 受け取った座標idのインデックス            */
  /*--------------------------------------------------*/

int Index(Add_Point *point, int vertex, int ID)
{

  for(int i=1; i<=vertex; i++){
    if(point[i].ID == ID) return i;
  }
  return -1;
}
```

Returns the index of the received coordinate id.

Result

(5.78049, 1.97561)

(10.5283, 5.84906) ←

(5.4, 4.2)

(4.2, 6.6)

The shortest path from the new point to the side.

Ideal result

(5.78049, 1.97561)

(9, 5) ←

(5.4, 4.2)

(4.2, 6.6)

(9, 5) is the shortest when considering existing points.

Find Highway(problem 8)

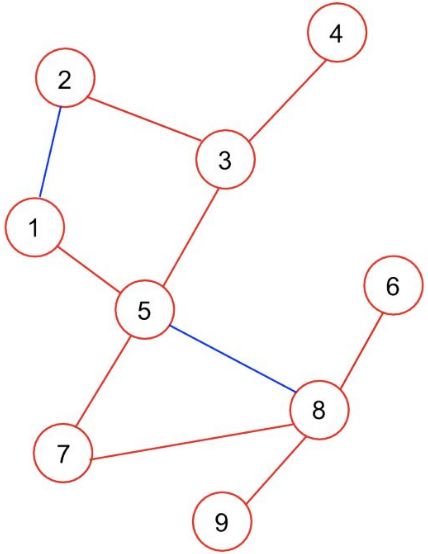
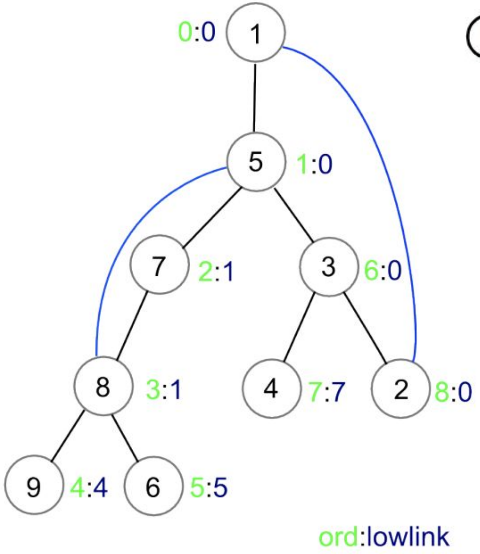
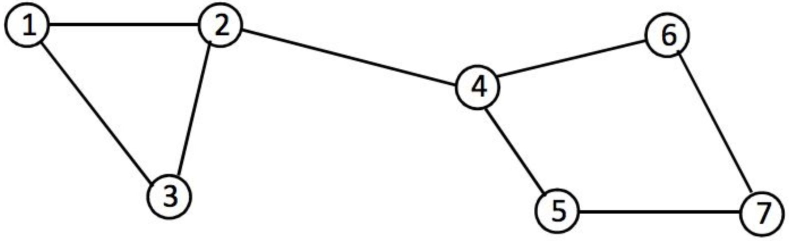


図 III.3　DFS 木 (赤) と後退辺 (青)



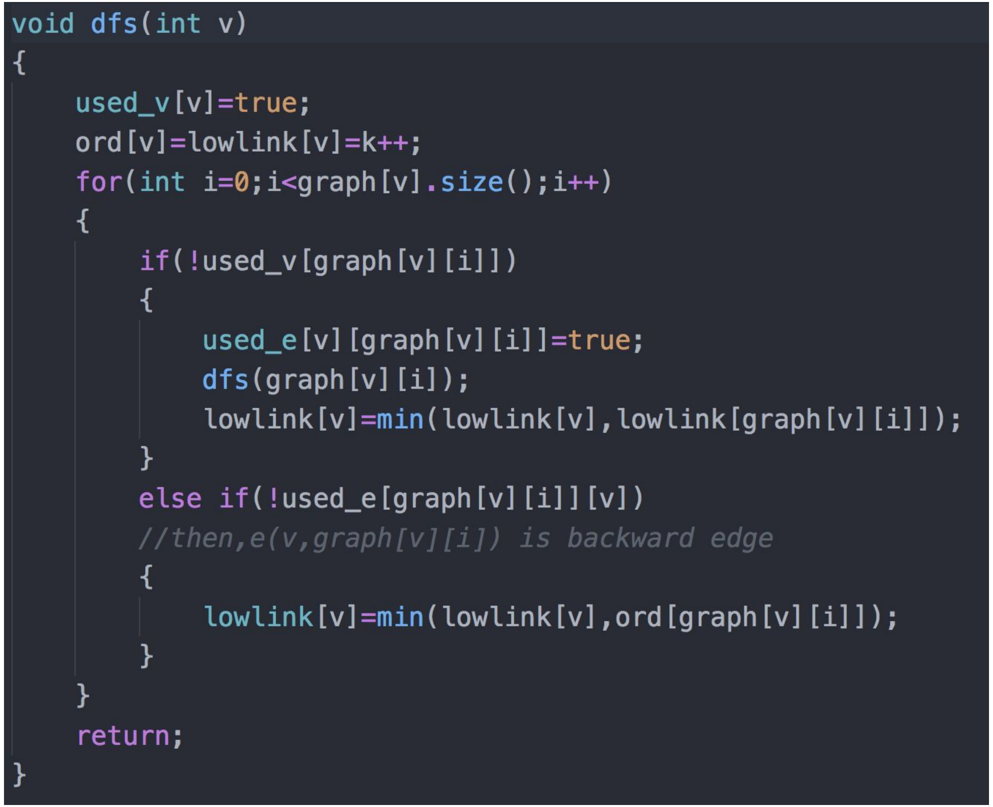
ord,lowlink



I used dfs algorithm.

if (ord[u] < lowlink[v]) = Highway

Find Highway(problem 8)



```
void dfs(int v)
{
    used_v[v]=true;
    ord[v]=lowlink[v]=k++;
    for(int i=0;i<graph[v].size();i++)
    {
        if(!used_v[graph[v][i]])
        {
            used_e[v][graph[v][i]]=true;
            dfs(graph[v][i]);
            lowlink[v]=min(lowlink[v],lowlink[graph[v][i]]);
        }
        else if(!used_e[graph[v][i]][v])
        //then,e(v,graph[v][i]) is backward edge
        {
            lowlink[v]=min(lowlink[v],ord[graph[v][i]]);
        }
    }
    return;
}
```

Contribution

contribution and work(midterm ～)

s1250110:

・ code lines： 505

(About issue 7)

・ Test data ・ Input / output example

・ Creating sample coordinate graph

(About normal processing and corner processing)

・ Discussion of test results

s1250132:

・code lines： 575

(About correction point in middle review, about the k-th shortest path)

・(modification from mid-term review)

①Added processing when two points are on the same line

②Added processing to determine if it is a point on a line segment

s1240072:

・ code lines： 94

・(About issue 8)

・Create template for presentation materials

・Find a useful document for problem

contribution and work (all)

we did not use git well and we were not able to soon share code.

Having distributed the work from Phase2, we were able to tackle each problem. However, the difference shown in the table.

Name	code(line)	task(ratio)
Kawauchi	621	4
Kimura	829	4.5
Takahagi	94	1.5

Project-wide consideration

Reflection point for the project

▪It is tempting to think that only the sample data given the task is to be successful.
⇨ The real problem have to consider various exceptions.

▪We made a lot of variable names and function.

⇨ hard to see, it overlooks the process to save, more processing time.

▪Put together in one code ⇨ Split into header files