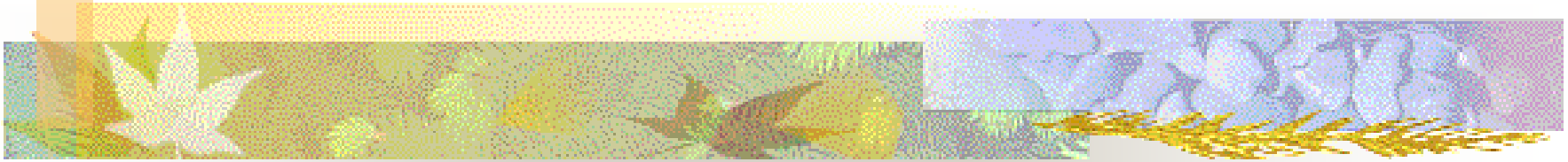
アルゴリズムとデータ構造b
11 - 最短経路問題(ダイクストラ法)



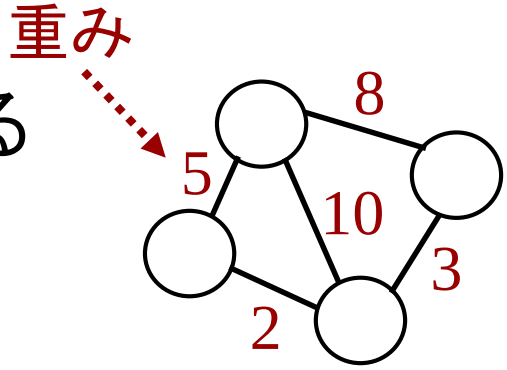
大見　嘉弘

本日の内容

- 最短経路問題
  - 最短経路問題とは
  - ダイクストラ法

最短経路問題とは

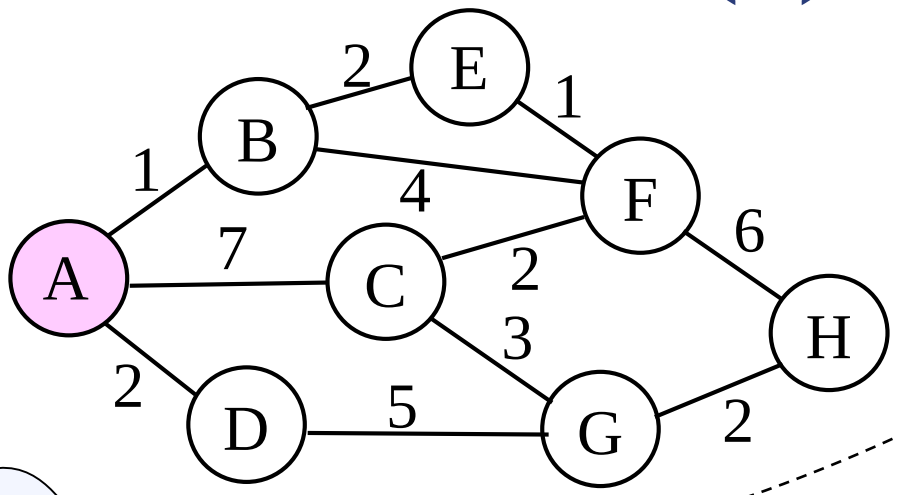
- ある始点からグラフの各点への最短距離を求める問題
- 鉄道等の最適経路、カーナビ、ネットワークの経路制御など実用度が高い
- グラフは重み付きグラフで表される
  - 各辺に重み(距離等)が付いている
- ダイクストラ法を使うのが一般的
  - 終点を決めていれば、グラフの全ての経路を辿らなくても最短経路が求まる
  - 終点に辿りつくまでに経由した各頂点までの最短経路も同時に求まる



ダイクストラ法

1. 始点に隣接している頂点の、始点ー頂点間の距離を求め、最小距離の頂点に印を付けて確定する
2. 印を付けた頂点に隣接している頂点までの距離を求め、この時点で計算されている(印の付いていない)頂点の距離の中で最短の頂点に印を付けて確定する
3. 2を全ての頂点に印が付くまで(あるいは指定した終点に印が付くまで)繰返すと、始点からの最短距離が求まる

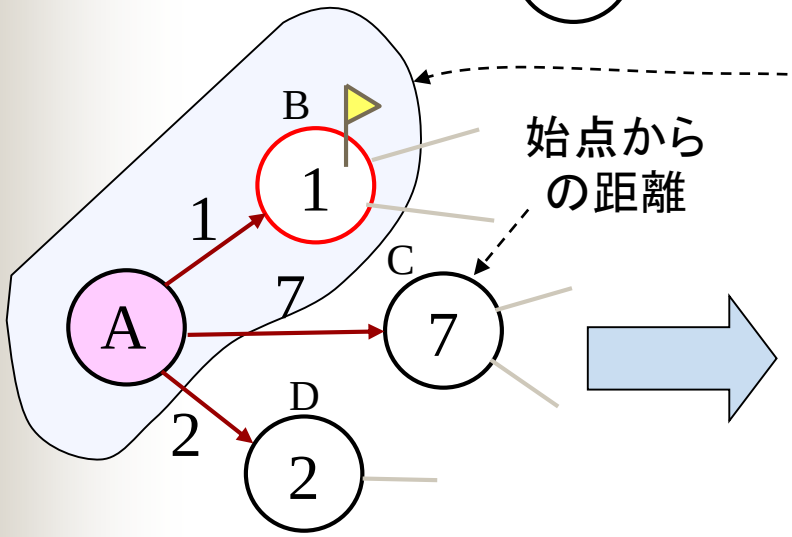
ダイクストラ法の例(1)

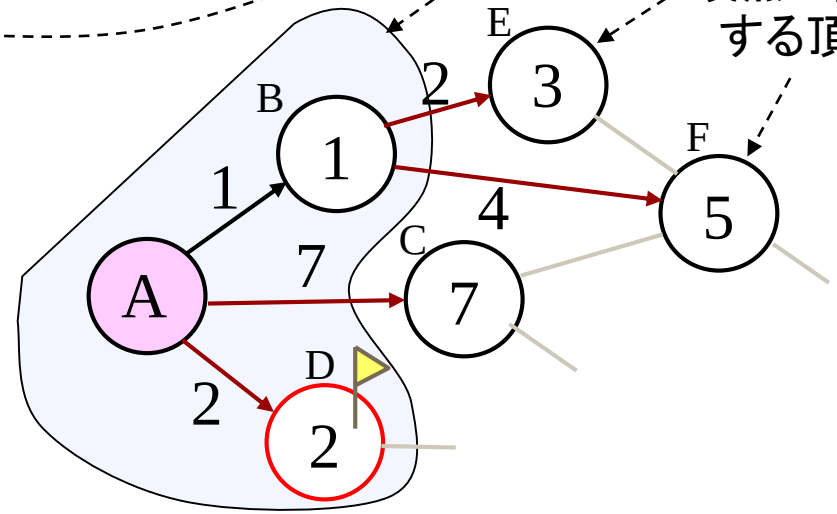


(最短距離が)
確定された範囲

印を付けた
頂点に隣接
する頂点

始点から
の距離

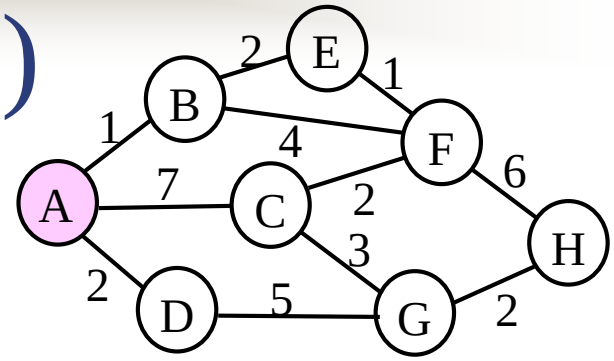


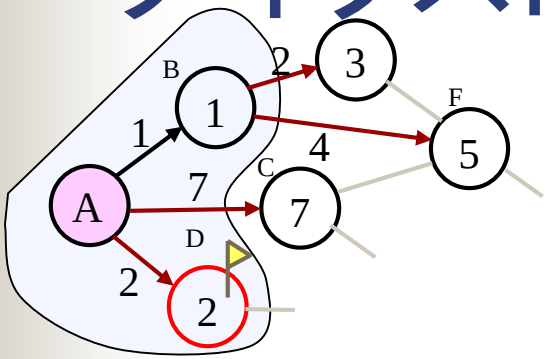


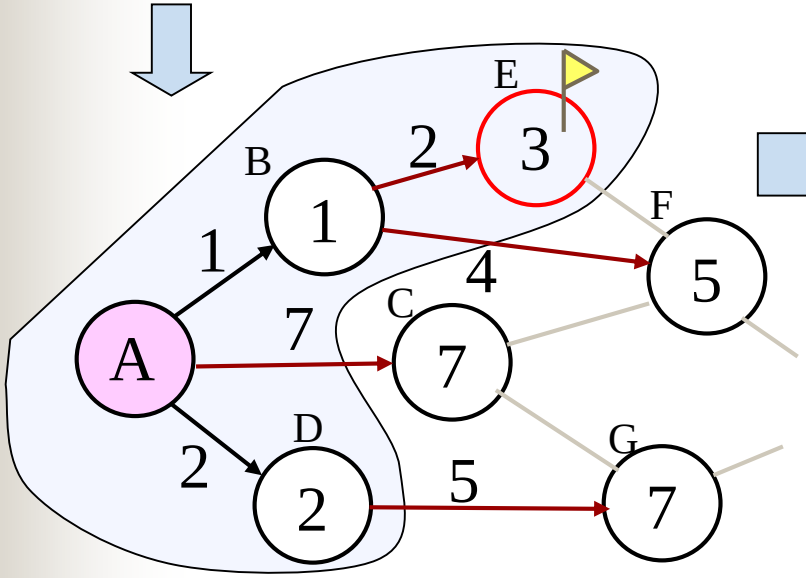
始点Aに隣接していて最短のBが確定

確定していない頂点で最短のDが確定

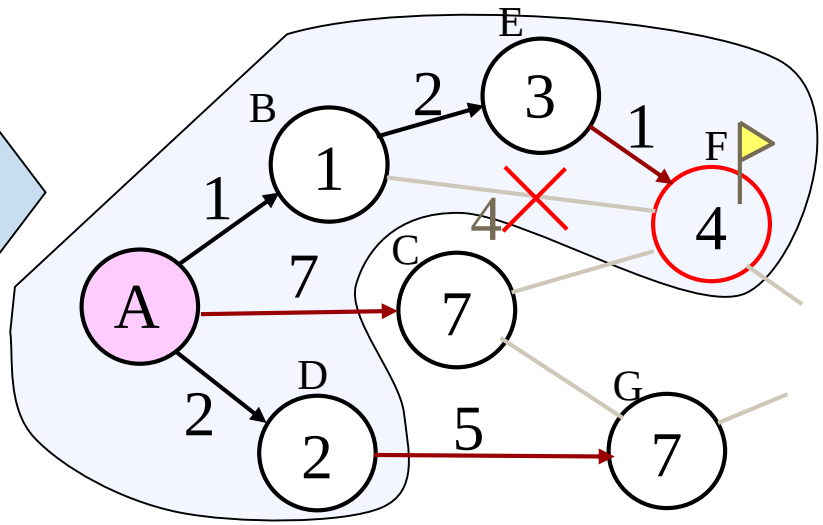
ダイクストラ法の例(2)





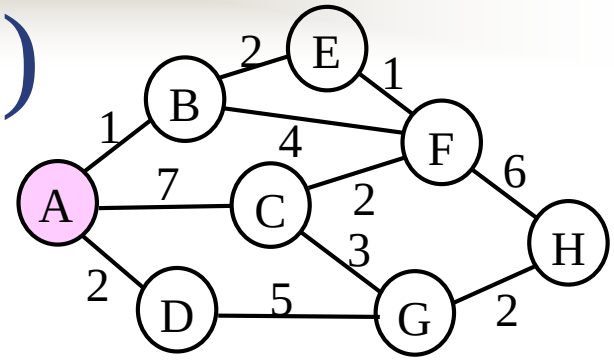


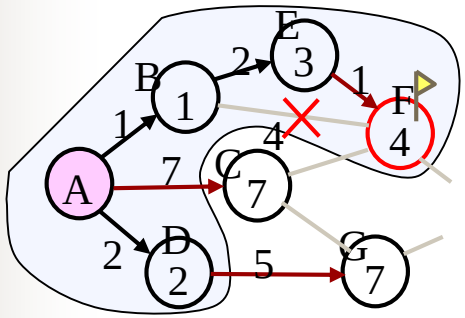
確定していない頂点で最短のEが確定

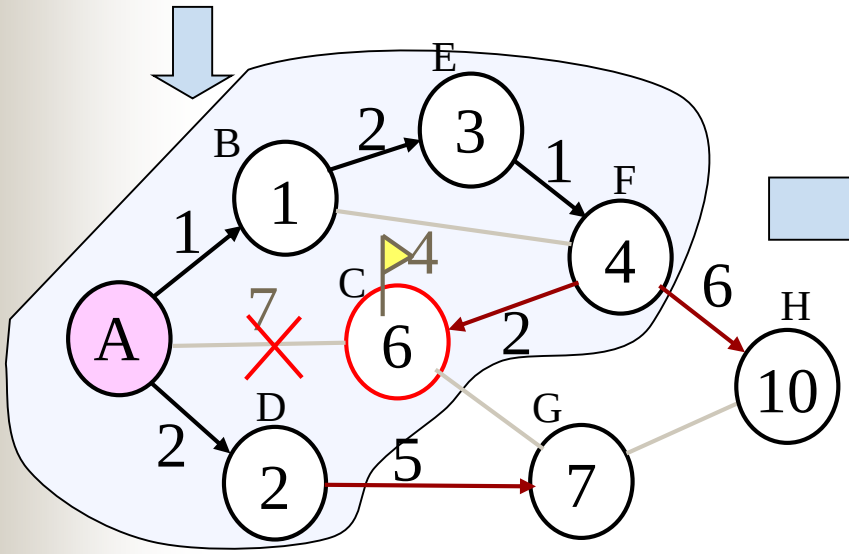


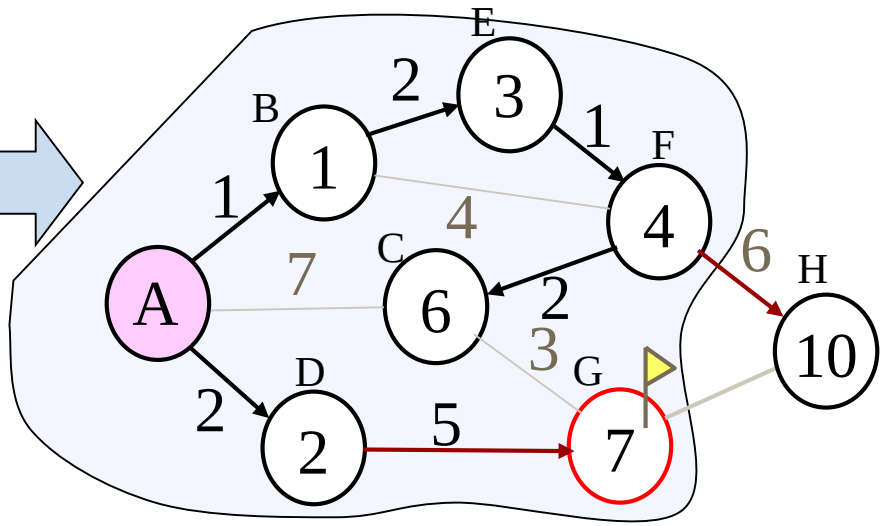
Fの距離は、印の付いたEを経由すると
4となり、前の5より小さいので経路
B→Fを切り、経路B→E→Fとする

ダイクストラ法の例(3)



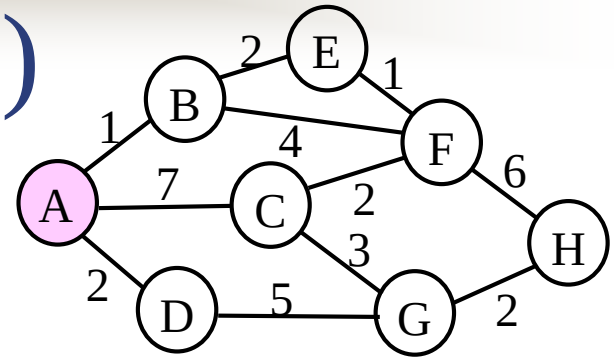


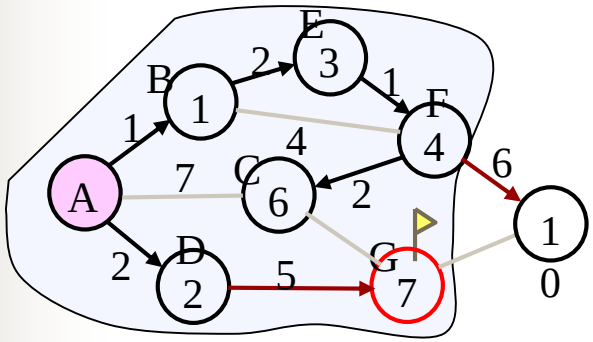


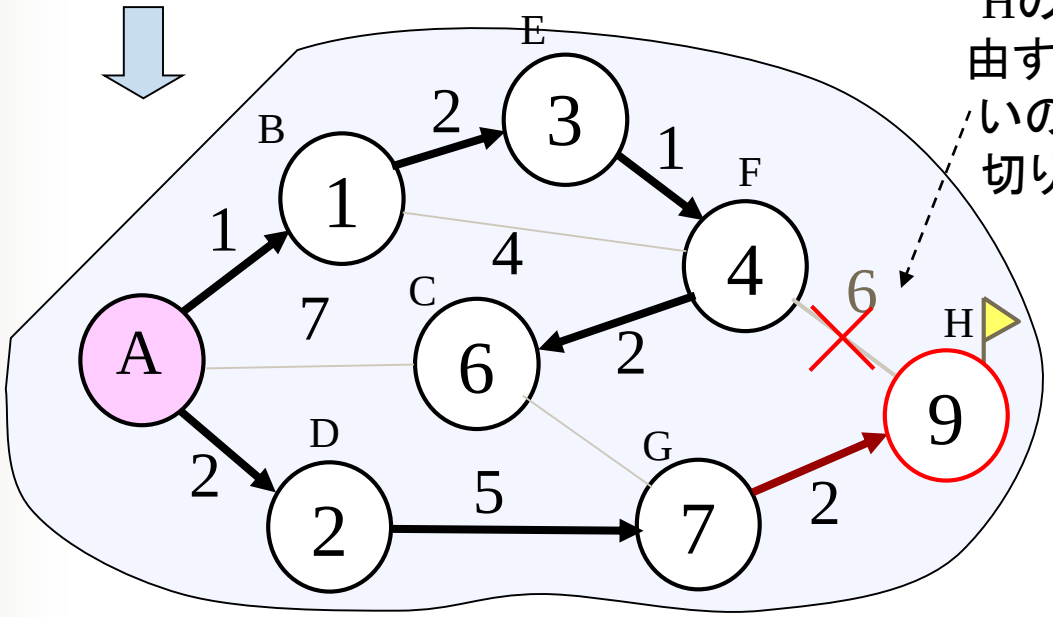


Cの距離は、印の付いたFを経由すると6となり、
前の7より小さいので経路A→Cを切り、経路
A→B→E→F→Cとする

ダイクストラ法の例(4)





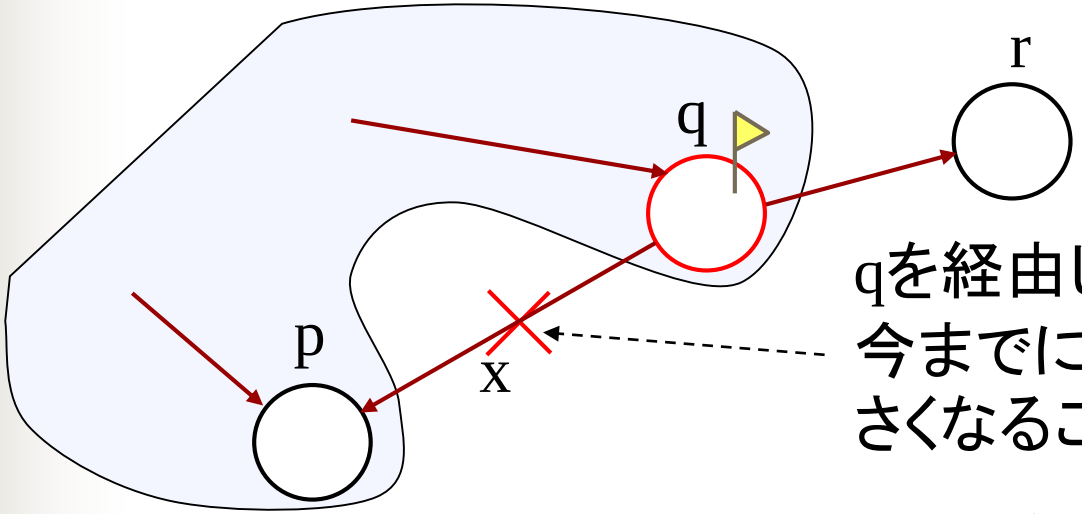


Hの距離は、印の付いたGを経由すると9となり、前の10より小さいので経路A→B→E→F→Hを切り、経路A→D→G→Hとする

全ての頂点の最短距離と最短経路が求まった

ダイクストラ法の仕組み

- pは以前に確定していて、qが確定(印を付けた)とする。pはqより前に確定した頂点であるから、p≦qである。qを経由してpに行く新たな経路(距離x)があったとしても、今までのpの距離より短くなることはありえない(p≦q+x)

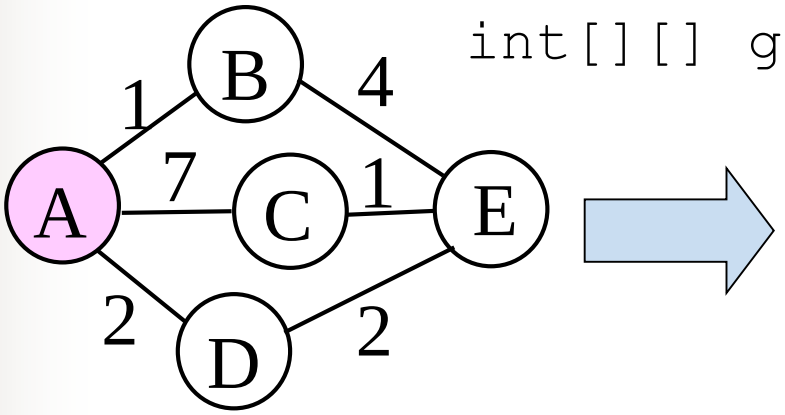


qを経由してpに行く距離は、今までに確定した距離より小さくなることはない

∵ p ≦ q であるから p ≦ q+x

ダイクストラ法のプログラム

- 重み付きグラフの配列表現
  - 隣接行列の値として重み(距離等)を入れる
  - 隣接していない場合はできるだけ大きな値を入れる(例：鉄道の距離の場合100万キロ)




```
final int M = 999999; // 大きな数
int[][] g = {{0,0,0,0,0,0},
             {0,0,1,7,2,M},
             {0,1,0,M,M,4},
             {0,7,M,0,M,1},
             {0,2,M,M,0,2},
             {0,M,4,1,2,0}};
```

使わない部分

プログラム例(1)

- その他、変数や配列

```
final int N = 5;              //  頂点の数
int[] leng=new int[N+1];      //  頂点までの距離
int[] v=new int[N+1];         //  確定フラグ
int start = 1;                //  始点番号
```

- 初期化

```
int i;
for (i = 1; i <= N; i++) {
     leng[i] = M;
     v[i] = 0;
}
leng[start] = 0;
```

距離を非常に大きくする

未確定にする(0:未確定,1:確定)

始点→始点の距離は0

プログラム例(2)

未確定の頂点のうち最小距離の頂点を求める。その頂点番号をp、距離をminに入れる

```
void solve(){
    int i,j,p=0,min;
    for (i=1; i<=N; i++) {
        min=M;
        for (j=1; j<=N; j++) {
            if (v[j]==0 && leng[j]<min) {
                p=j;
                min=leng[j];
            }
        }
        v[p]=1;
        if (min==M) {
            System.err.println("到達できません");return;
        }
        for (j=1; j<=N; j++) {
            if (leng[p]+g[p][j] < leng[j]) {
                leng[j] = leng[p]+g[p][j];
            }
        }
    }
    …
```

頂点pを確定
(印を付ける)

pを経由して、jに至る距離が、それまでの最短距離より短ければ更新

プログラムリスト(前半)

```
public class ShortestPathSolver {
    final int N = 5; // 頂点の数
     final int M = 999999; // 非常に大きい数
    int[][] g = {{0,0,0,0,0,0},
                 {0,0,1,7,2,M},
                 {0,1,0,M,M,4},
                 {0,7,M,0,M,1},
                 {0,2,M,M,0,2},
                 {0,M,4,1,2,0}};
    int[] leng=new int[N+1]; // 頂点までの距離
    int[] v=new int[N+1]; // 確定フラグ
     int start = 1; // 始点番号

    void solve(){
           int i,j,p=0,min;

           for (i=1; i<=N; i++) {
                      min=M; // 最小の頂点を探す
                      for (j=1; j<=N; j++) {
                                 if (v[j]==0 && leng[j]<min) {
                                            p=j;
                                            min=leng[j];
                                 }
                      }
                      v[p]=1;
                      if (min==M) {
                                 System.err.println("到達できません");
                                 return;
                      }
                      for (j=1; j<=N; j++) {
                                 if (leng[p]+g[p][j] < leng[j]) {
                                            leng[j] = leng[p]+g[p][j];
                                 }
                      }
           }
```

プログラムリスト(後半)

```
            // 結果表示
            for (i=1; i<=N; i++) {
                        System.out.println(start + "->" + i + " : " + leng[i]);
            }
    }

    public static void main(String args[]) {
        ShortestPathSolver prog = new ShortestPathSolver();
        prog.start();
    }

     public void start() {
            int i;
            for (i = 1; i <= N; i++) {
                        leng[i] = M;
                        v[i] = 0;
            }
            leng[start] = 0;
            solve();
     }
}
```

プログラム例のトレース(1)

i	j	min	p	v	1	2	3	4	5	leng	1	2	3	4	5	備考
					0	0	0	0	0		M	M	M	M	M	
											0	M	M	M	M	leng[start]=0
			0													
1																
		M														
	1	0	1													leng[1]<min
	2															
	3															
	4															
	5															
					1	0	0	0	0							頂点1が確定 v[1]=1
	1															
	2										0	1	M	M	M	leng[1]+g[1][2]は1
	3										0	1	7	M	M	leng[1]+g[1][3]は7
	4										0	1	7	2	M	leng[1]+g[1][4]は2
	5															leng[1]+g[1][5]はM

プログラム例のトレース(2)

i	j	min	p	v	1	2	3	4	5	leng	1	2	3	4	5	備考
2																
		M														
	1															v[1]==1
	2	1	2													leng[2]<min
	3															leng[3]>min
	4															leng[4]>min
	5															leng[5]>min
					1	1	0	0	0							頂点2が確定 v[2]=1
	1															leng[2]+g[2][1]は2
	2															leng[2]+g[2][2]は1
	3															leng[2]+g[2][3]は1+M
	4															leng[2]+g[2][4]は1+M
	5										0	1	7	2	5	leng[2]+g[2][5]は5
3																
		M														
	1															v[1]==1
	2															v[2]==1
	3	7	3													leng[3]<min
	4	2	4													leng[4]<min
	5															leng[5]>min
					1	1	0	1	0							頂点4が確定 v[4]=1
	1															leng[4]+g[4][1]は4
	2															leng[4]+g[4][2]はM+2
	3															leng[4]+g[4][3]はM+2
	4															leng[4]+g[4][4]は2
	5										0	1	7	2	4	leng[4]+g[4][5]は4

プログラム例のトレース(3)

i	j	min	p	v	1	2	3	4	5	leng	1	2	3	4	5	備考
4																
		M														
	1															v[1]==1
	2															v[2]==1
	3	7	3													leng[3]<min
	4															v[4]==1
	5	4	5													leng[5]<min
					1	1	0	1	1							頂点5が確定 v[5]=1
	1															leng[5]+g[5][1]はM+4
	2															leng[5]+g[5][2]は8
	3										0	1	5	2	4	leng[5]+g[5][3]は5
	4															leng[5]+g[5][4]は6
	5															leng[5]+g[5][5]は4
5																
		M														
	1															v[1]==1
	2															v[2]==1
	3	5	3													leng[3]<min
	4															v[4]==1
	5															v[5]==1
					1	1	1	1	1							頂点3が確定 v[3]=1
	1															leng[3]+g[3][1]は12
	2															leng[3]+g[3][2]はM+5
	3															leng[3]+g[3][3]は5
	4															leng[3]+g[3][4]はM+5
	5															leng[3]+g[3][5]は6
											0	1	5	2	4	

宿題

- 下図のグラフの各頂点の最短距離をダイクストラ法により求めよ(始点をAとする)
  解答例:
  Aの最短距離: 0, Bの最短距離: 2, …