アルゴリズム論
（第６回）

岩手県立大学
Iwate Prefectural University

佐々木研（情報システム構築学講座）
講師　山田敬三
k-yamada@iwate-pu.ac.jp

11/20(金) 中間試験

- 各自が履修しているクラスで受験すること
  （でなければ，採点されない）
- 持ち込み：一切不可
- 解答時間：60分

- 11/27 採点（参加者は+10点）

最短経路問題

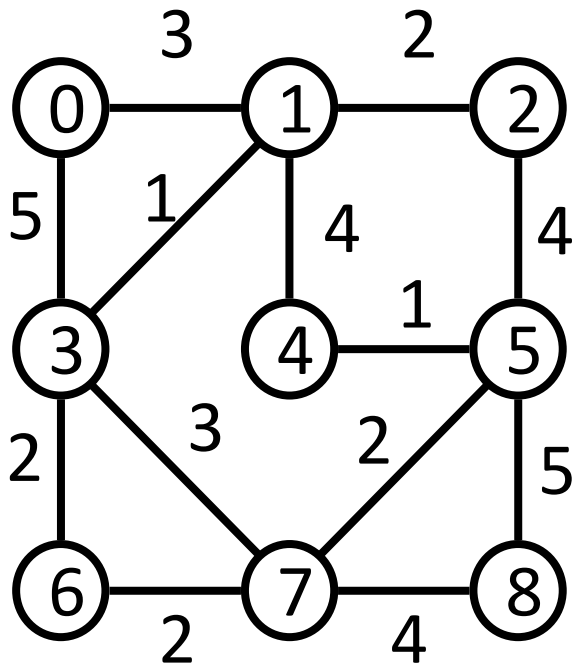
岩手県立大学
Iwate Prefectural University

最短経路問題

- 重み付きグラフを用いる.
- 経路上の各辺に対する重みの総和が最小となる経路を見つける.
  - 最も短い経路，最も安い経路など

隣接行列

- 重み付きグラフ



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

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

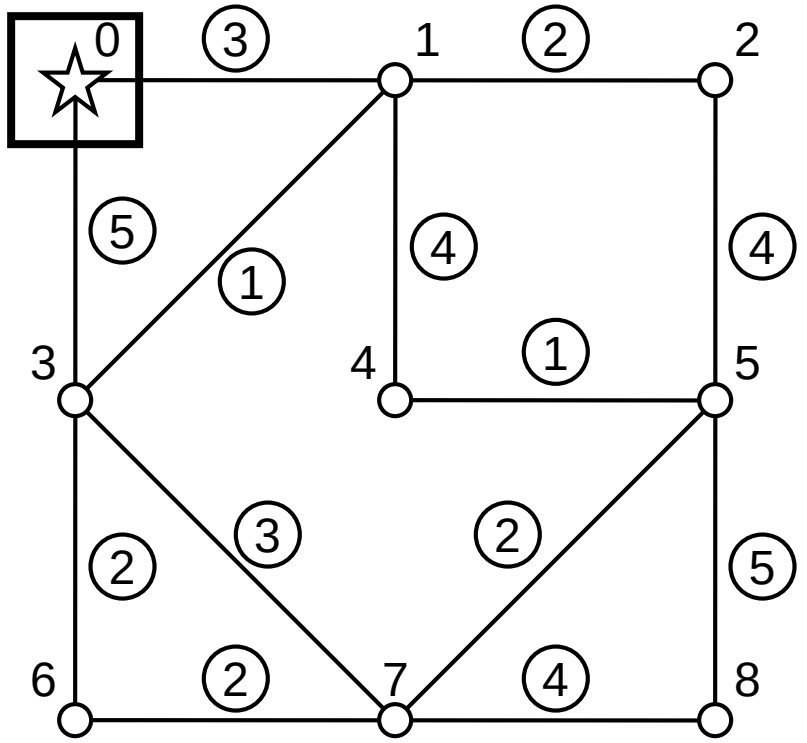
ダイクストラ法（最短経路長）

- あるノードからあるノードへ到達できる最短の距離を求める.
（ノード0→ノード5）

1. 初期状態
   i. 始点（ノード0）の最短経路長を0にし，ノード0を太線で囲う.
   ii. ノード0の隣接ノードにノード0からの距離を最短経路長として与える.
   この距離は更新される可能性がある.
   iii. 他のノードの経路長は∞

ダイクストラ法（最短経路長）

1



ノード	0	1	2	3	4	5	6	7	8
経路長	0	3	∞	5	∞	∞	∞	∞	∞

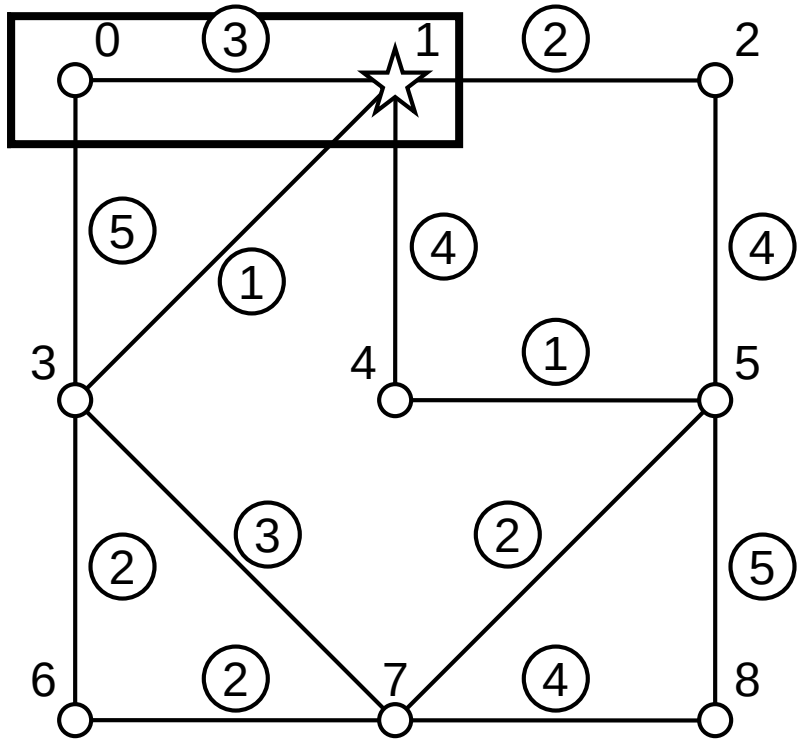
ダイクストラ法（最短経路長）

2. 探索するノードと最短経路長の決定
   以下の条件を満たすノードを調査し，そのノードの現在の最短経路長を正式な値として囲う．
   i. 囲みの中のいずれかのノードの隣接ノード
   ii. 囲みの外の全ノードのうち，経路長が最も小さいノード

   ここで選択されたノードを**u**とする．

ダイクストラ法（最短経路長）

2



ノード	0	1	2	3	4	5	6	7	8
経路長	0	3	∞	5	∞	∞	∞	∞	∞

ノード1がuとなる

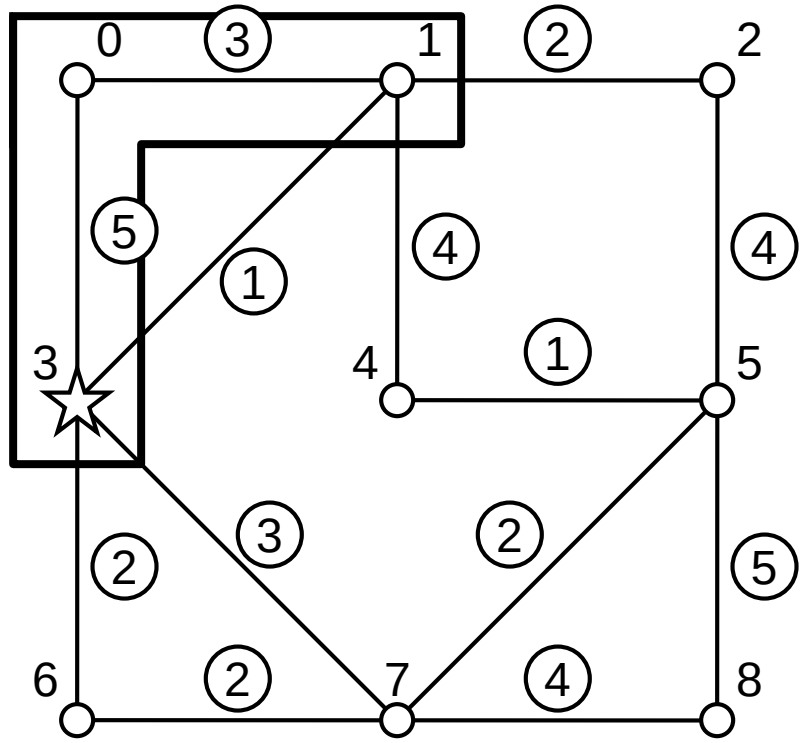
ダイクストラ法（最短経路長）

3. 隣接ノードの最短経路長の更新
   ノード**u**の隣接ノード**v**に対して最短経路長を更新する.
   i. ノードuの最短経路長にu,v間の距離を加えた距離を計算
   ii. その距離がvの今の最短経路長よりも小さければ値を更新

- 以上の処理をノード**u**の全隣接ノードについて行う.

ダイクストラ法（最短経路長）

3

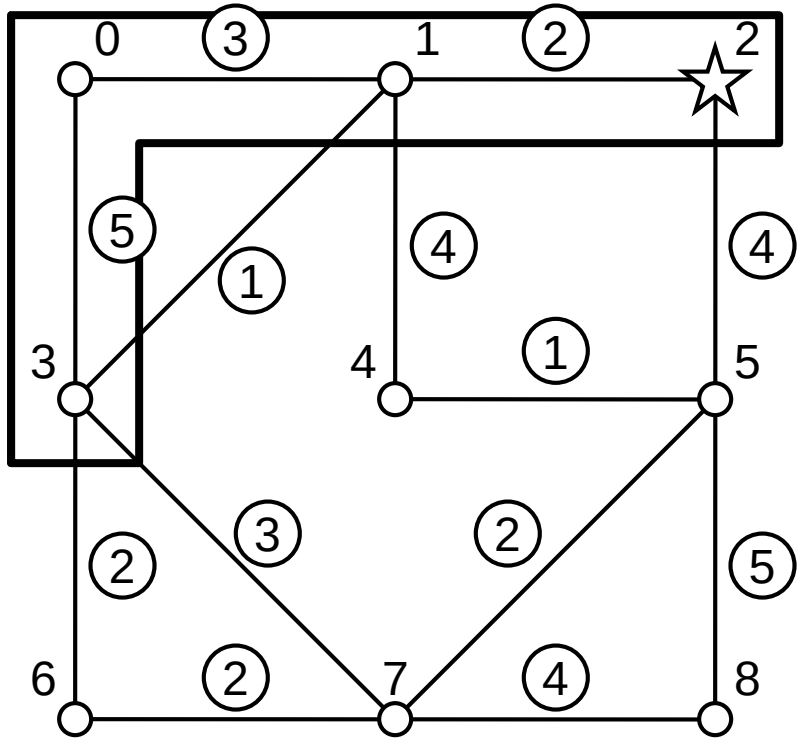


ノード	0	1	2	3	4	5	6	7	8
ノード	0	1	2	3	4	5	6	7	8
更新前経路長	0	3	∞	4	∞	∞	∞	∞	∞
	0	3	5	4	7	∞	∞	∞	∞
更新後	0	3	5	4	7	∞	∞	∞	∞

ダイクストラ法（最短経路長）

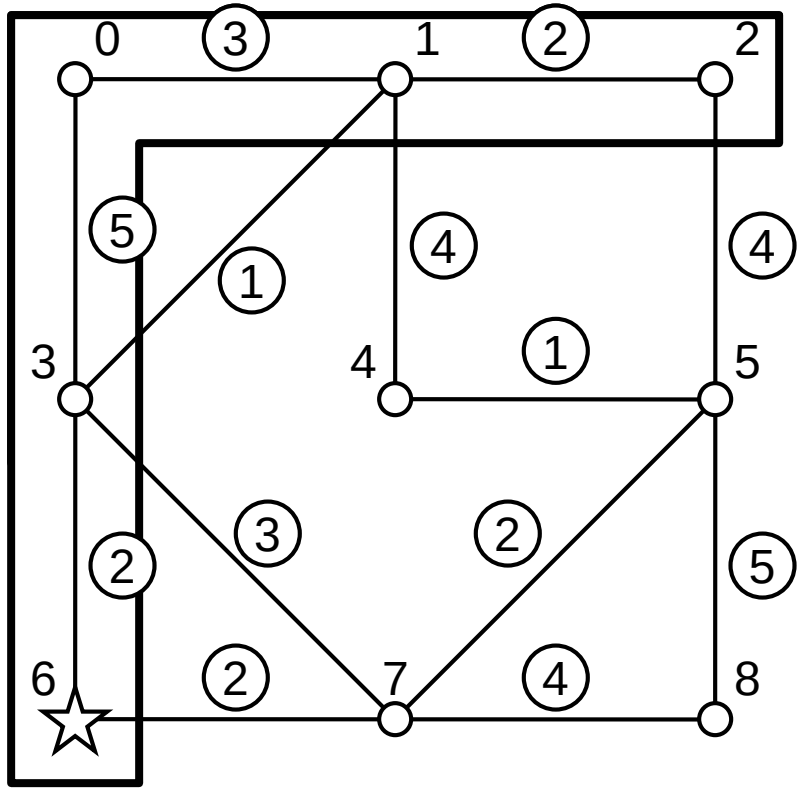
4. 繰り返し&終了条件
   手順2.からの処理を繰り返す.
   終点（ノード5）の最短経路長が確定した時点で終了.
   このとき，未探索ノードが存在する場合もある.

ダイクストラ法（最短経路長）



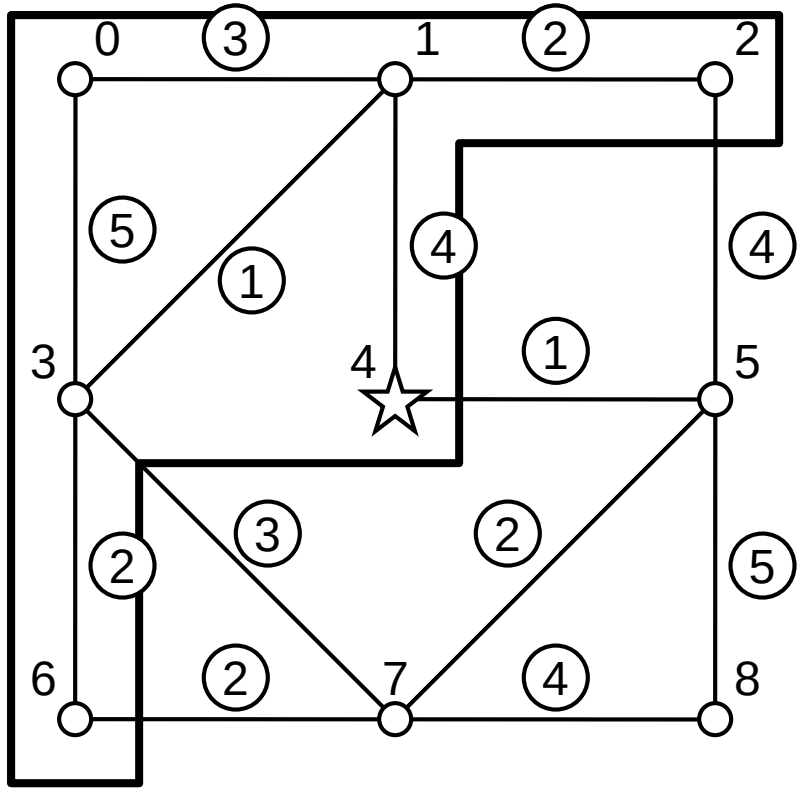
ノード	0	1	2	3	4	5	6	7	8
更新前	0	3	5	4	7	∞	∞	∞	∞
更新後	0	3	5	4	7	∞	6	7	∞

ダイクストラ法（最短経路長）



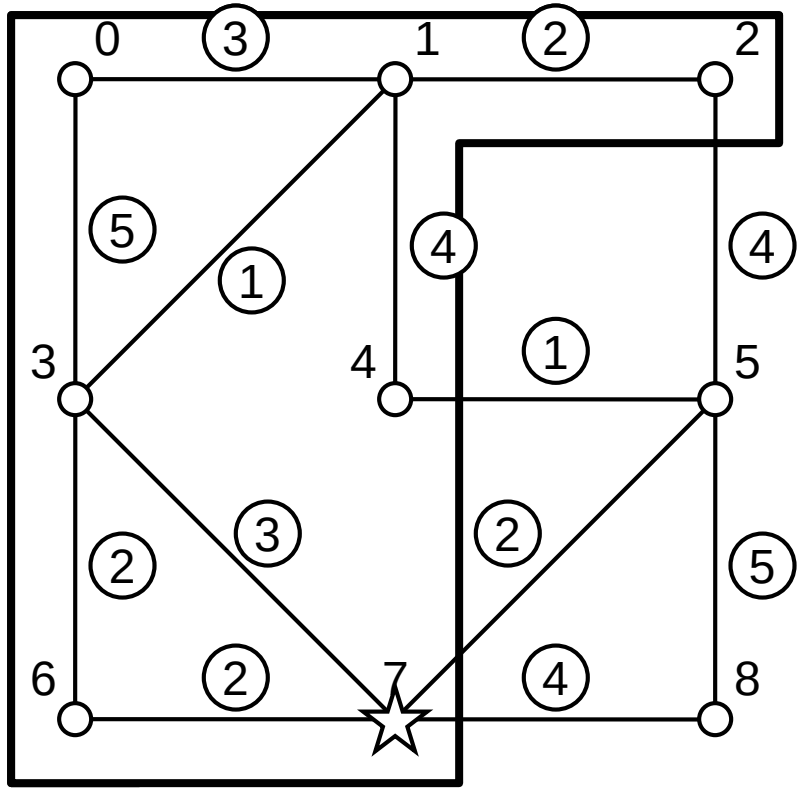
ノード	0	1	2	3	4	5	6	7	8
更新前	0	3	5	4	7	∞	6	7	∞
更新後	0	3	5	4	7	9	6	7	∞

ダイクストラ法（最短経路長）



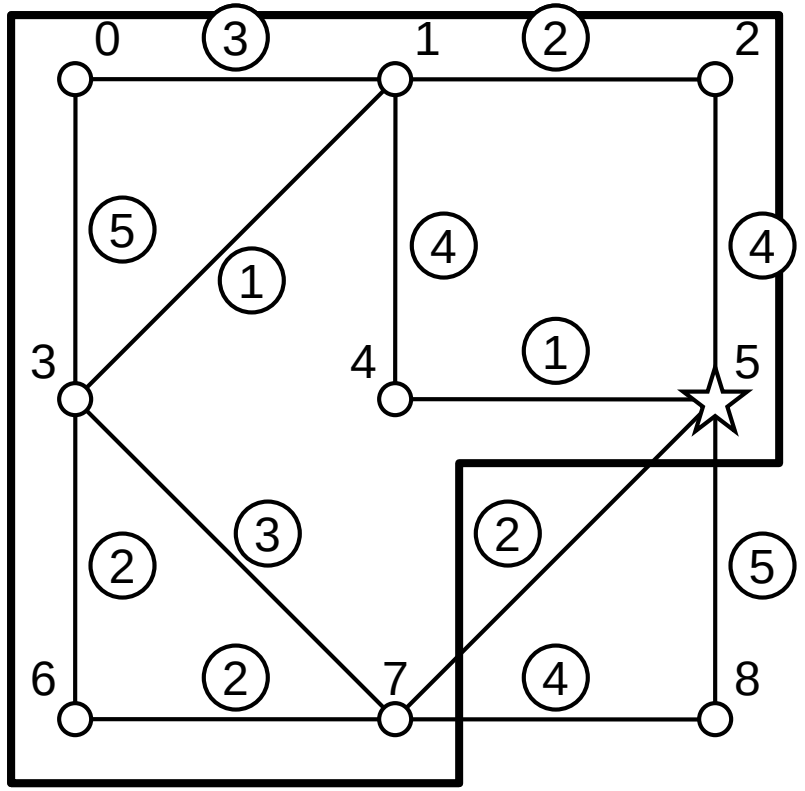
ノード	0	1	2	3	4	5	6	7	8
更新前	0	3	5	4	7	9	6	7	∞
更新後	0	3	5	4	7	9	6	7	∞

ダイクストラ法（最短経路長）



ノード	0	1	2	3	4	5	6	7	8
更新前	0	3	5	4	7	9	6	7	∞
更新後	0	3	5	4	7	8	6	7	∞

ダイクストラ法（最短経路長）



ノード	0	1	2	3	4	5	6	7	8
更新前	0	3	5	4	7	8	6	7	∞
更新後	0	3	5	4	7	8	6	7	11

使用するデータ

```
#define NODES 9     /* ノードの数 */
#define INF   999   /* 無限大 */

struct node {       /* ノード */
  int dist;         /* 始点からの距離 */
  int flag;         /* 囲みの中かどうか */
};

int matrix[NODES][NODES];     /* 隣接行列 */
struct node node_dat[NODES]; /* ノード情報 */
```

処理手順

1. 隣接行列を作成
   graph2.datを読込み，隣接行列作成
   辺が存在しないノード間は無限大（**INF**）
2. 初期設定
   ノード情報を表す**node_dat[]**のメンバの，**dist**を無限大に，**flag**に0を代入
3. 探索処理
   ノード0を始点，ノード5を終点とする探索
4. 結果を出力

探索処理

`void dijkstra(int start, int end)`

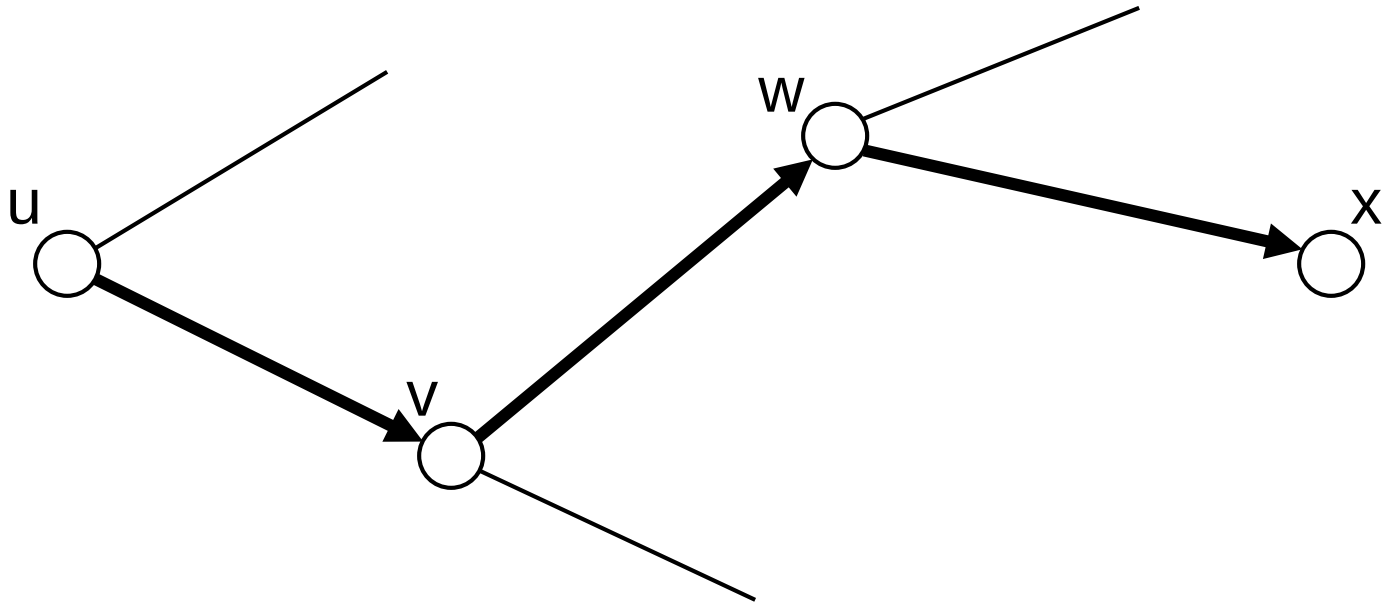
1. 始点の囲みへの登録
   `node_dat[start].flag`に囲みの中にあることを示す1を代入
2. 始点と隣接ノードの経路長の設定
   始点の経路長を0に，隣接ノードの，そのノードまでの距離をメンバ**`dist`**に代入
3. 囲み外の経路長が最小のノードの調査
   `node_dat[x].flag`と
   `node_dat[x].dist`を参照し，囲み外の最小経路長のノード**`u`**を求める.

探索処理

4. ノードuの囲みへの登録
   手順1.のように，ノード**u**を囲み内に登録
5. ノードuの隣接ノードの経路長の更新
   ノード**u**の隣接ノードであり，囲み外の
   ノード**v**に対し経路長を計算し，
   `node_dat[v].dist`よりも小さい場合
   は更新
6. 次のノードの探索
   手順3.に戻り，次のノードを探索
   ノード**u**が終点である場合は終了

最短経路

- これまでのアルゴリズムでは，最短経路長は求められるが，最短経路は求められない．
- ダイクストラ法を改良することで最短経路を求める．



使用するデータ

```
#define NODES 9             /* ノードの数 */
#define INF   999           /* 無限大 */
#define END_OF_PATH -1      /* 終端を示す */

struct node {   /* ノード */
  int dist;     /* 始点からの距離 */
  int flag;     /* 囲みの中かどうか */
  int path;     /* 前の隣接ノード */
};

int matrix[NODES][NODES];     /* 隣接行列 */
struct node node_dat[NODES]; /* ノード情報 */
```

追加点

- 手順2. 始点と隣接ノードの経路長の設定
  ノード**start**の**path**を**END_OF_PATH**に設定
- 手順5. ノードuの隣接ノードの経路長更新
  ノード**u**の隣接ノード**v**の更新時に，ノード**u**はノード**v**までの最短経路を導くノードなので，**node_dat[v].path**に**u**を代入
- 終点から逆順にパスを表示