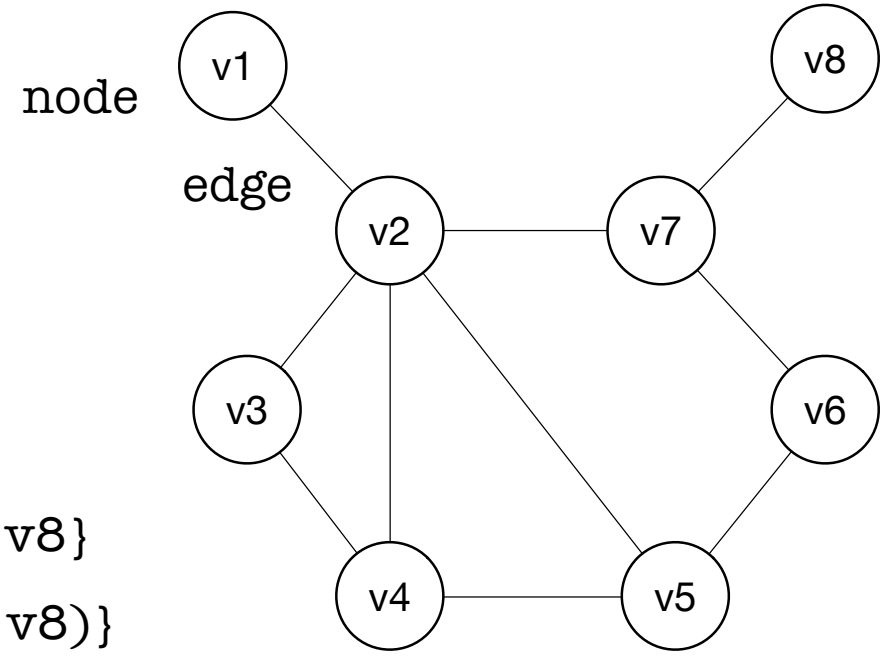
データ構造とアルゴリズム

第４回 グラフ

小池 英樹 (*koike@c.titech.ac.jp*)

グラフ(GRAPH) (復習)

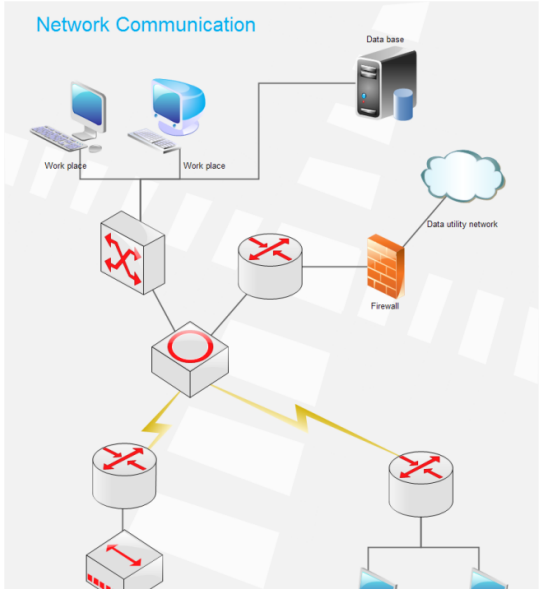
- Graph G=(V, E)
  - V: 有限個の節点(node, vertex, 頂点)の集合
  - E: 有限個の節点対e=(vi, vj) (枝，辺，edge, arc, branch)の集合
    - vi, vj: 端点

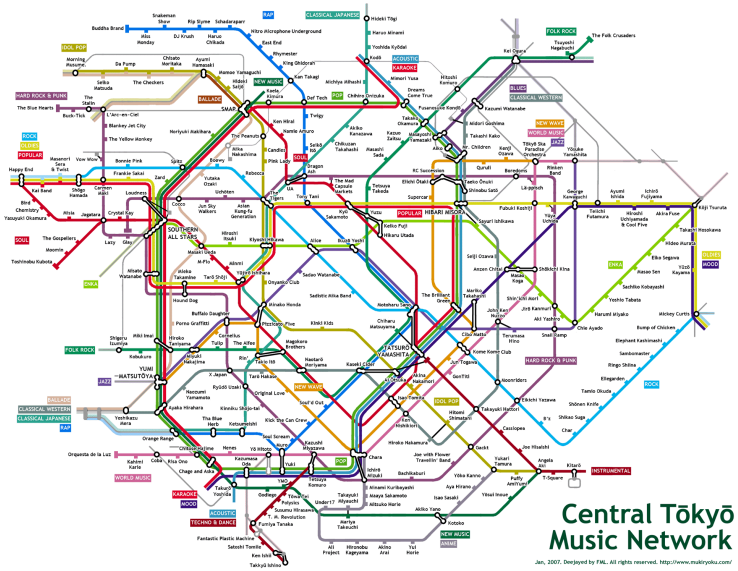


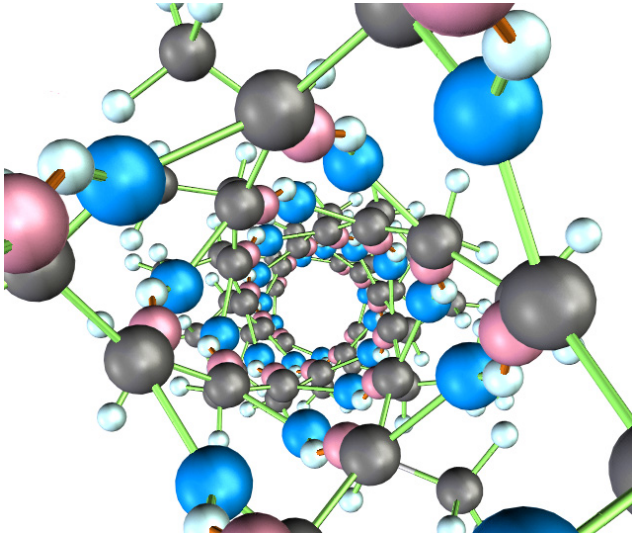
V={v1, v2, v3, v4, v5, v6, v7, v8}

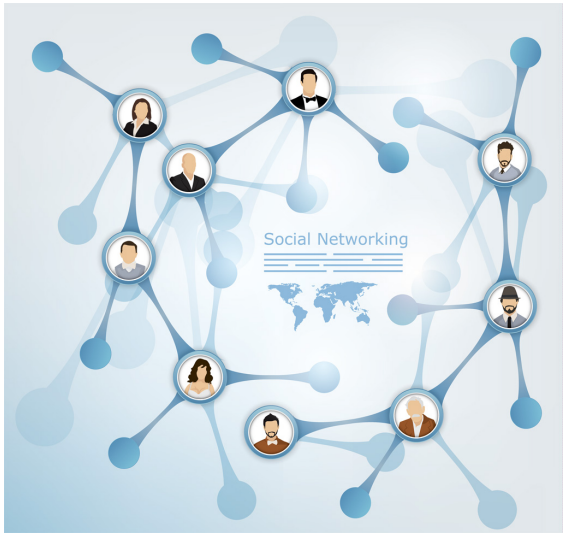
E={(v1, v2), (v2, v3), ..., (v7, v8)}

GRAPHの例


Network Communication
Data base
Work place
Work place
Data utility network
Firewall


Central Tōkyō
Music Network




Social Networking

GRAPH APPLICATION

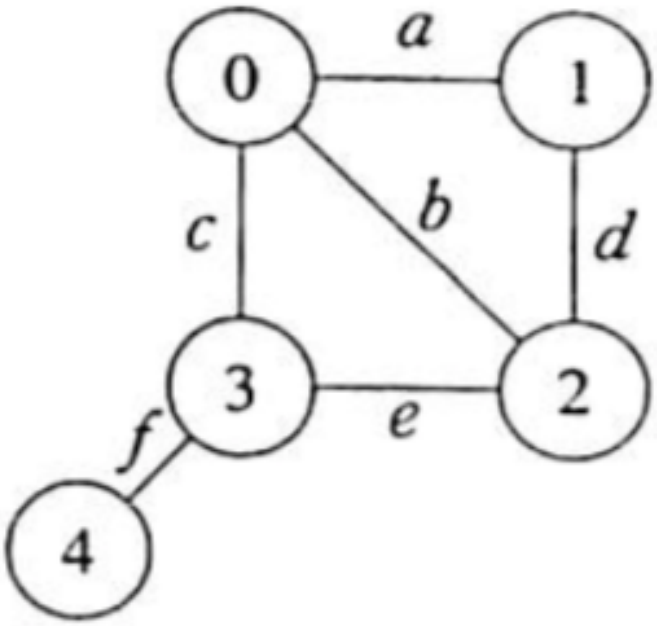
graph	vertex	edge
communication	telephone, computer	cable
circuit	gate, register, processor	wire
mechanical	joint	rod, beam, spring
financial	stock, currency	transactions
transportation	intersection	street
internet	class C network	connection
game	board position	legal move
social relationship	person	friendship
neural network	neuron	synapse
protein network	protein	protein-protein interaction
molecule	atom	boan

GRAPH-PROCESSING PROBLEMS

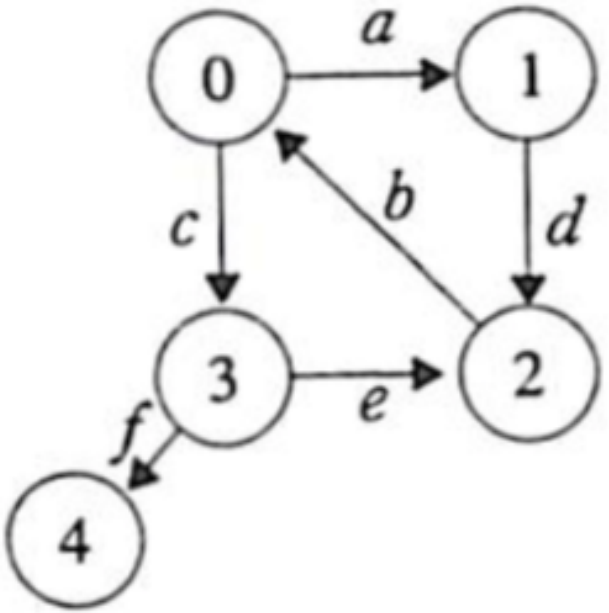
problem	description
s-t path	sとtの間にpathはあるか？
shortest s-t path	sとtの間の最小のpathは何か?
cycle	グラフの中に閉路はあるか?
Euler cycle	各エッジを１回しか使わない閉路はあるか?
Hamilton cycle	各ノードを１回しか使わない閉路はあるか?
connectivity	すべてのノード接続する方法はあるか?
biconnectivity	削除するとグラフが連結でなくなるノードはあるか?
planarity	エッジを交差させることなく２次元平面に描画できるか?
graph isomorphism	２つの隣接リストは同じグラフか?

無向グラフと有向グラフ

無向グラフ

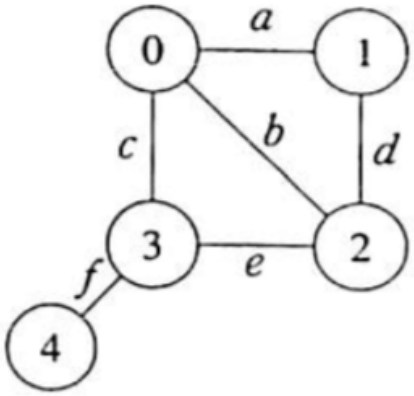


有向グラフ



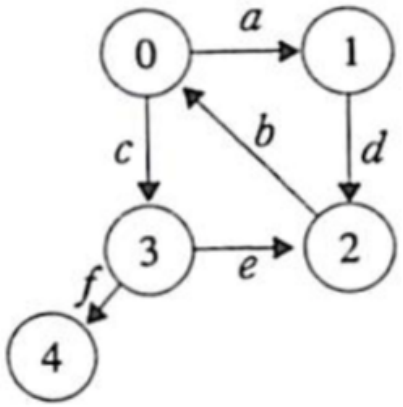
接続行列(INCIDENCE MATRIX)による表現

無向グラフ



	a	b	c	d	e	f
0	1	1	1	0	0	0
1	1	0	0	1	0	0
2	0	1	0	1	1	0
3	0	0	1	0	1	1
4	0	0	0	0	0	1

有向グラフ

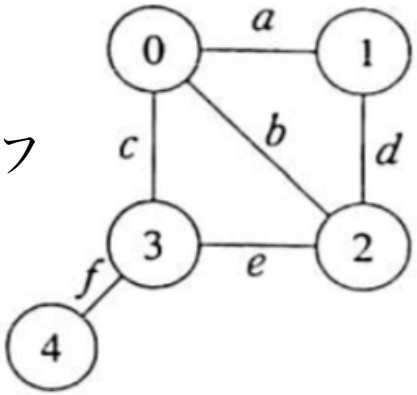


	a	b	c	d	e	f
0	1	−1	1	0	0	0
1	−1	0	0	1	0	0
2	0	1	0	−1	−1	0
3	0	0	−1	0	1	1
4	0	0	0	0	0	−1

節点数n，枝数mに対し，O(mn)の領域を必要とする

隣接行列(ADJACENCY MATRIX)による表現

無向グラフ

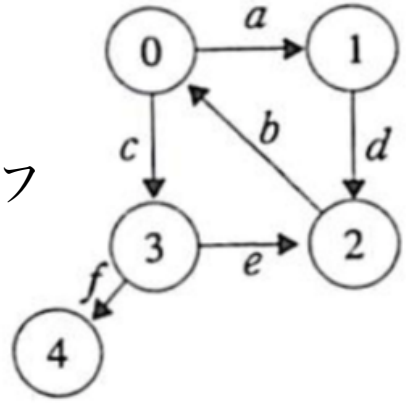


	0	1	2	3	4
0	0	1	1	1	0
1	1	0	1	0	0
2	1	1	0	1	0
3	1	0	1	0	1
4	0	0	0	1	0

	0	1	2	3	4
0	0	a	b	c	0
1	a	0	d	0	0
2	b	d	0	e	0
3	c	0	e	0	f
4	0	0	0	f	0

ラベル付きグラフの場合

有向グラフ



	0	1	2	3	4
0	0	1	0	1	0
1	0	0	1	0	0
2	1	0	0	0	0
3	0	0	1	0	1
4	0	0	0	0	0

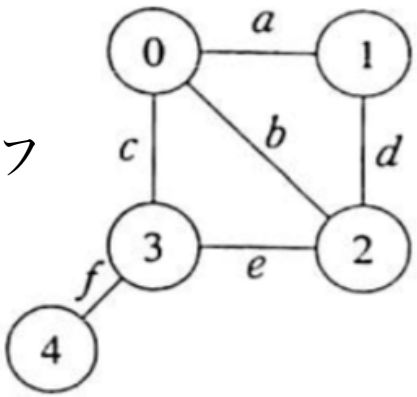
	0	1	2	3	4
0	0	a	0	c	0
1	0	0	d	0	0
2	b	0	0	0	0
3	0	0	e	0	f
4	0	0	0	0	0

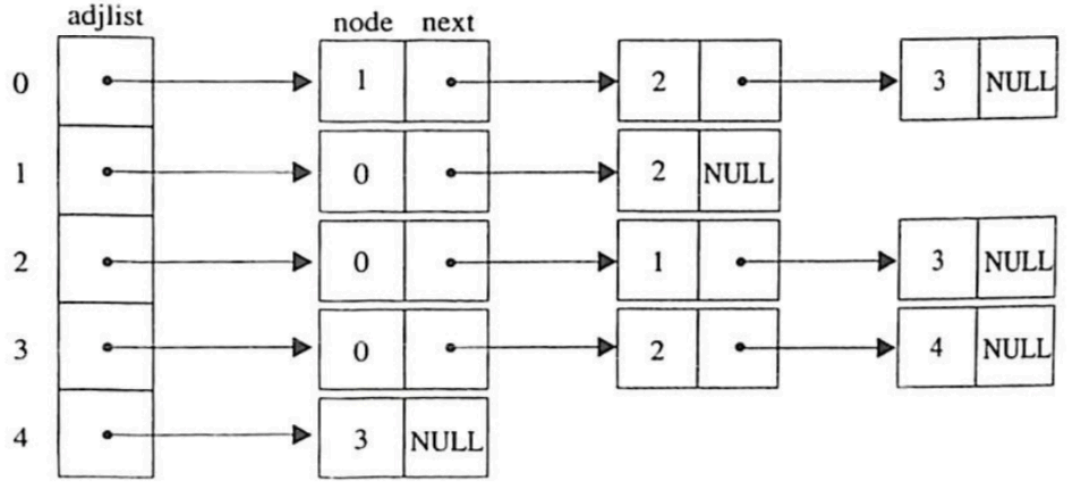
ラベル付きグラフの場合

節点数nに対し，$O(n^2)$の領域を必要とする

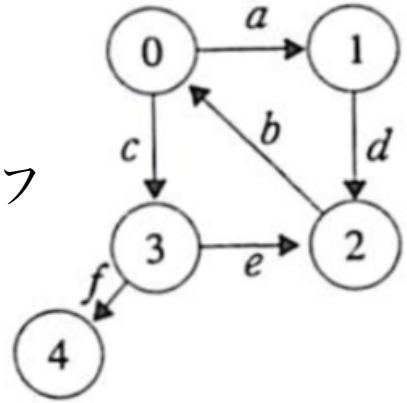
隣接リスト(ADJACENCY LIST)のポインタによる表現

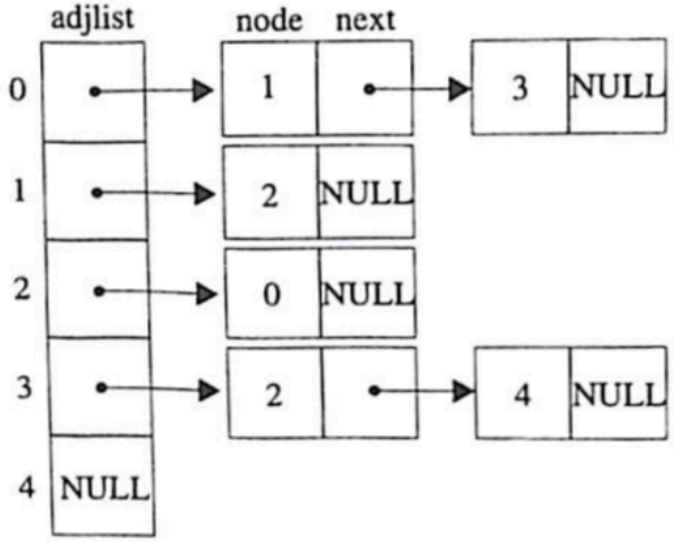
無向グラフ





有向グラフ

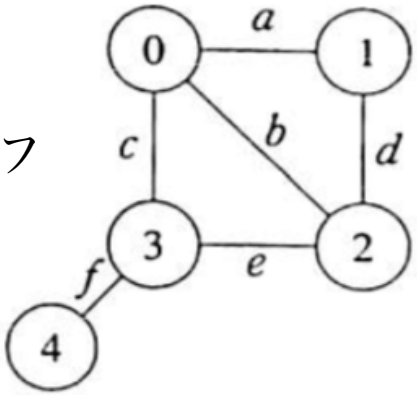


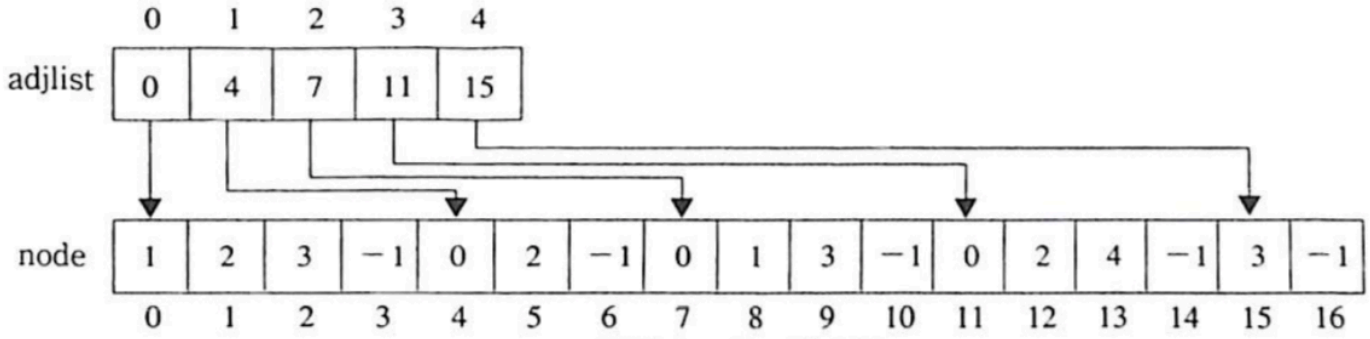


必要な領域量はO(m + n)

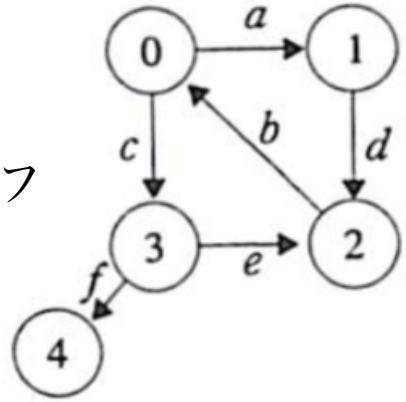
隣接リストの別表現

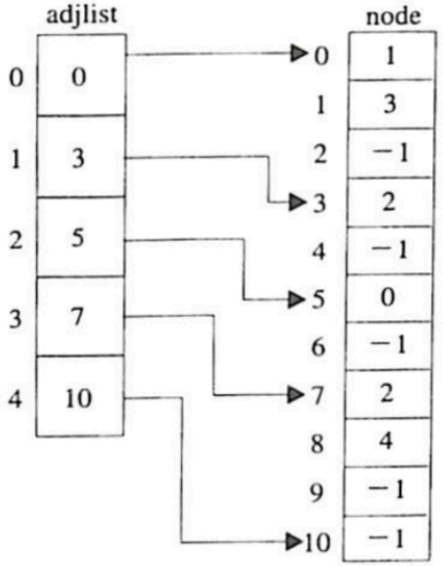
無向グラフ





有向グラフ





隣接行列と隣接リストの利点・欠点

	利点	欠点
隣接行列	２頂点間に辺があるか否かをO(1)でチェック可能	O(n^2)の記憶領域が必要 １つの頂点の隣接頂点を求めるのにO(n)時間必要
隣接リスト	O(m+n)の記憶領域ですむ １つの頂点の隣接頂点を求めるのには，その隣接頂点数に比例した時間だけで可能	２頂点間に辺があるか否かをチェックするのに，隣接頂点数に比例した時間が必要

現実世界のグラフは疎(sparse)なものが多いので，隣接リストが多く使われる．

グラフの探索

- グラフが与えられたとき，そのすべての頂点を訪問すること

- 応用：
  - ゲーム（迷路，ボードゲーム），画像処理，etc.

- 基本的探索法：
  - 深さ優先探索(depth-first-search)
  - 幅優先探索(breadth-first-search)

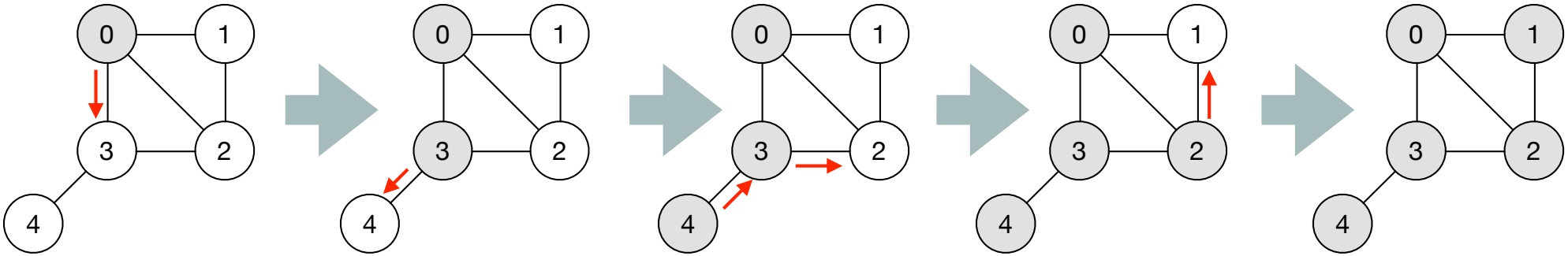
深さ優先探索

- 基本的な考え方
  - 出発点から始めて，長兄を調べ，次に長兄の長兄を調べ，と縦方向を優先して調べる．
  - 木の走査(前順，中順，後順）と同じ．
  - ただし，閉路(cycle)があるので，一度調べた(visited)頂点は再度訪問しない．

深さ優先探索

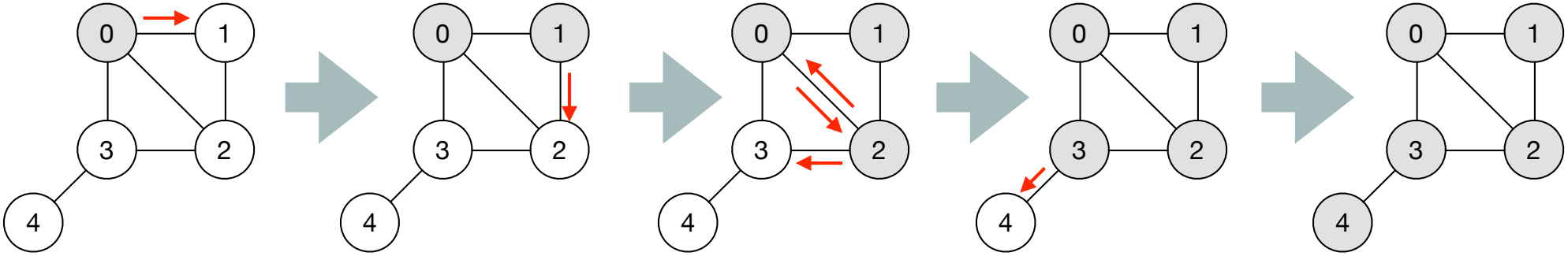
動作の概要

訪問の順番: 0->3->4->2->1



別なケース（データの定義に依存する）

訪問の順番: 0->1->2->3->4



深さ優先探索

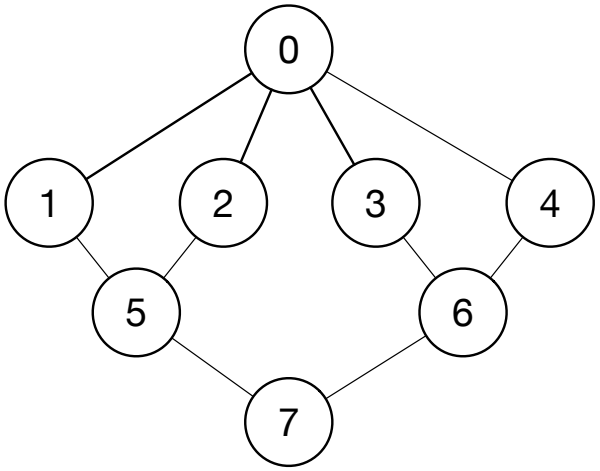
アルゴリズムの概要（再帰版）

```
dfs(v) {
    vをvisitedとマークする
    vに関する処理を行う
    for (vに接続しているすべての頂点 i)
        if (iがunvisited)
            dfs(i)
}
```

深さ優先探索：隣接行列によるプログラム例

```
void dfs(int i)
{
    int j;

    printf("\n%d",i);
    visited[i]=1;

    for(j=0;j<n;j++)
        if(!visited[j]&&G[i][j]==1)
            dfs(j);
}
```

```
void main() {
    int i;

    for (i=0; i<8; i++) /* 初期化 */
        visited[i] == 0;

    dfs(0);
}
```




```
int G[8][8] = {
    0, 1, 1, 1, 1, 0, 0, 0,
    1, 0, 0, 0, 0, 1, 0, 0,
    1, 0, 0, 0, 0, 1, 0, 0,
    1, 0, 0, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 0, 0, 1, 0,
    0, 1, 1, 0, 0, 0, 0, 1,
    0, 0, 0, 1, 1, 0, 0, 1,
    0, 0, 0, 0, 0, 1, 1, 0,
};

int visited[8];
```

深さ優先探索

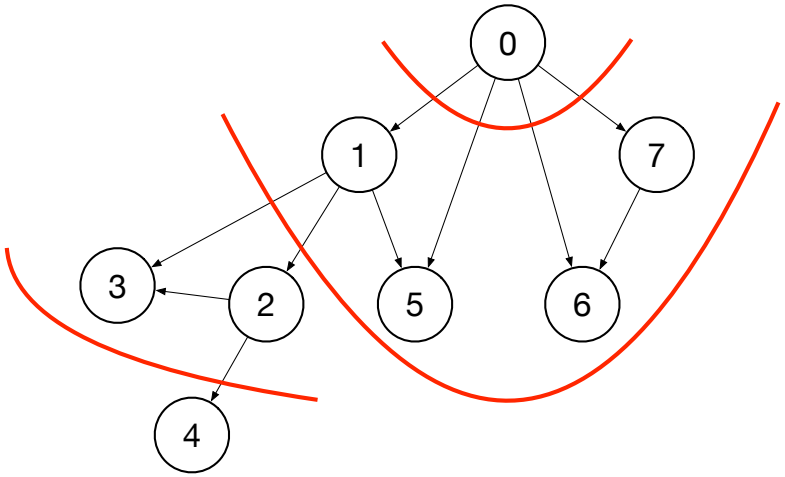
アルゴリズムの概要（非再帰版）

```
dfs(v) {
    S = 空のスタック
    vをvisitedとマークする
    vをSに追加(push)
    while (Sが空でない)
        v = Sから取り出す(pop)
        vに関する処理を行う
        for (vに接続しているすべての頂点 i)
            if (iがunvisited)
                iをvisitedとマークする
                iをSに追加(push)
}
```

幅優先探索

- 基本的な考え方
  - 出発点から始めて，子供をすべて調べる．子供を調べ終わってから，孫をすべて調べる．

動作の概略



幅優先探索

アルゴリズムの概要

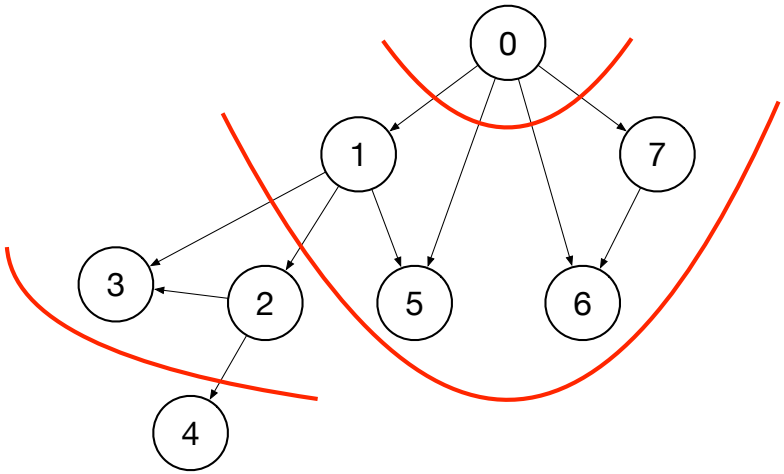
```
bfs(v) {
    Q = 空のqueue
    vをvisitedとマークする
    vをQに追加(enqueue)
    while (Qが空でない)
        v = Qから取り出す(dequeue)
        vに関する処理
        for (vに接続する頂点i)
            if (iがunvisited)
                iをvisitedとマークする
                iをQに追加する
}
```

幅優先探索

アルゴリズムのstep数にともなうQの状態

step	visited	adjacent vertex	Q
0	-		0
1	0	1, 5, 6, 7	1, 5, 6, 7
2	1	3, 2, 5	5, 6, 7, 3, 2
3	5	null	6, 7, 3, 2
4	6	null	7, 3, 2
5	7	6	3, 2
6	3	null	2
7	2	3, 4	4
8	4	null	-

*下線は次に取り出される頂点，赤字は新たに追加された頂点



隣接リスト

vertex	adjacent vertex
0	1, 5, 6, 7
1	3, 2, 5
2	3, 4
3	null
4	null
5	null
6	null
7	6

レポート課題

- (1)1つ前のスライドのグラフを隣接リストで表すプログラムを書き，これを，深さ優先探索（再帰）するプログラムを書き，実行しなさい.

- (2)上記グラフを，待ち行列(queue)を用いた幅優先探索で探索するプログラムを書き，実行しなさい.

- 締切：1/26(金) 17:00
- 提出先：algo2017@vogue.c.titech.ac.jp

- 告知：2/6(火)は期末試験