アルゴリズムとデータ構造b
10 - グラフの探索(幅優先探索)

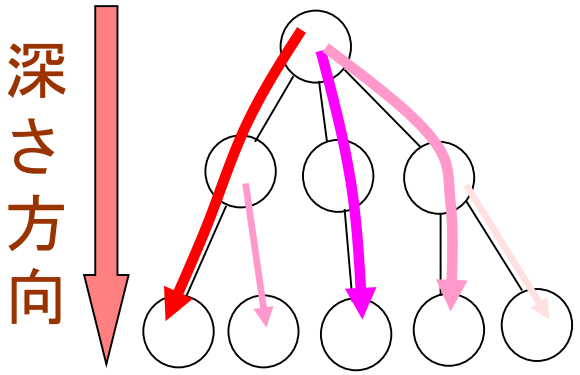
大見　嘉弘

本日の内容

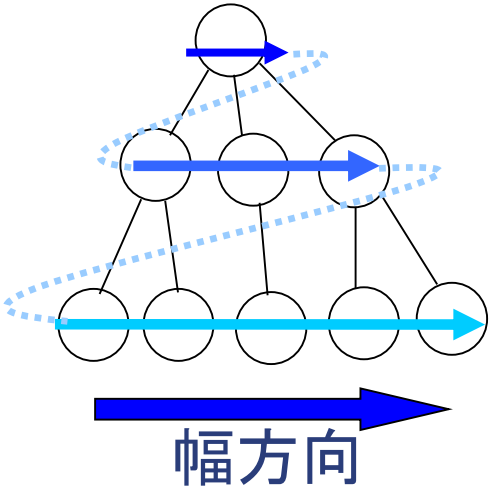
- グラフの探索
  - 幅優先探索

幅優先探索

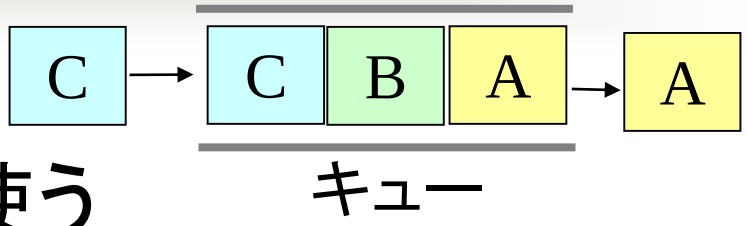
- 縦方向(上→下)を先に辿るのが深さ優先探索
  - 縦型探索ともいう



- 横方向(左→右)を先に辿るのが幅優先探索
  - 横型探索ともいう



幅優先探索

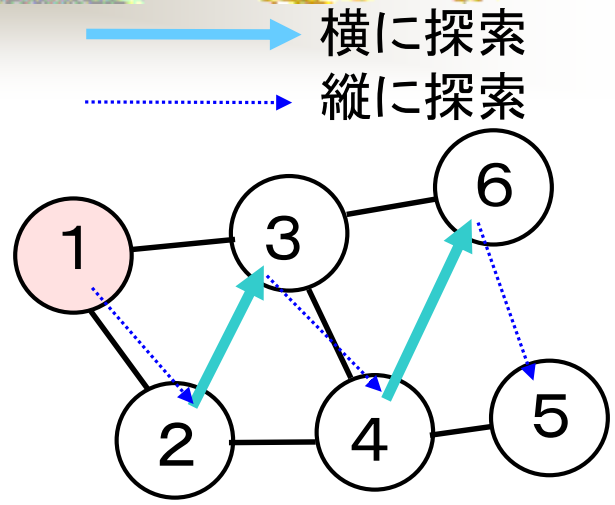


- キュー(待ち行列)を使う

1. 始点をキューに入れる
2. キューから頂点を1つ取り出し、その頂点に隣接している(全ての)未訪問の頂点をキューに入れる。
3. キューが空になったら終了。空でなければ2に行く。

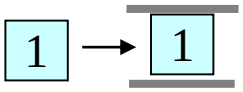
- 頂点をキューに入れる順番が探索順序
(キューから取り出す順序も同様(FIFO))

幅優先探索の例(1)

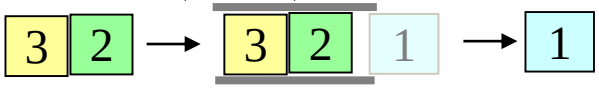


a. 頂点1から開始

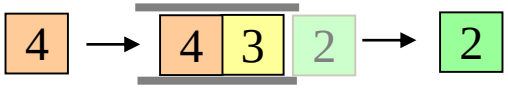
b. 頂点1をキューに入れる



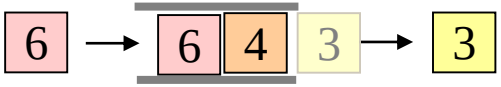
c. キューから頂点1を取り出す。
頂点1に隣接していて未訪問な頂点(2,3)をキューに入れる。



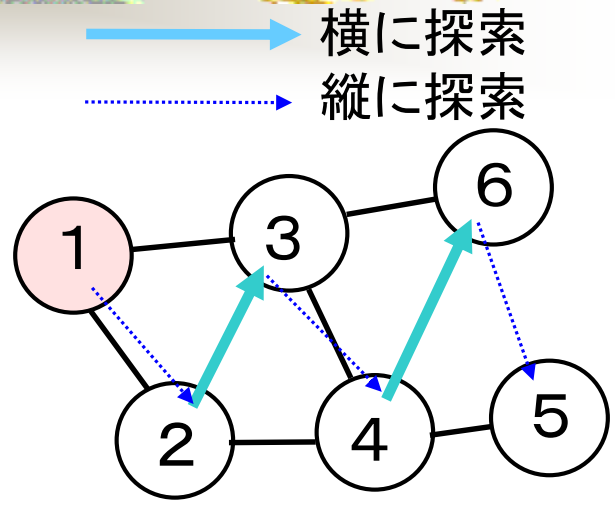
d. キューから頂点2を取り出す。頂点2に隣接していて未訪問な頂点4をキューに入れる。



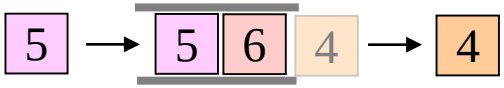
e. キューから頂点3を取り出す。頂点3に隣接していて未訪問な頂点6をキューに入れる。



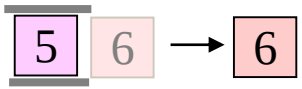
幅優先探索の例(2)



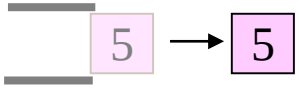
f. キューから頂点4を取り出す。
頂点4に隣接していて未訪問な頂点5をキューに入れる。



g. キューから頂点6を取り出す。
頂点6に隣接していて未訪問な頂点は無いので、キューには何も入れない。



h. キューから頂点5を取り出す。
頂点5に隣接していて未訪問な頂点は無いので、キューには何も入れない。



i. キューが空になったので終了。

プログラム例(1)

- 深さ優先探索と同様な部分
  - グラフは隣接行列で表す
  - 訪問フラグを設ける

```
int[][] g = {{0,0,0,0,0,0,0},
             {0,0,1,1,0,0,0},
             {0,1,0,0,1,0,0},
             {0,1,0,0,1,0,1},
             {0,0,1,1,0,1,0},
             {0,0,0,0,1,0,0},
             {0,0,0,1,0,0,0}};
final int N = 6; // 頂点の数
int[] v = new int[N+1]; // 訪問フラグ
                        // 0:未訪問,1:訪問済
```

使わない部分

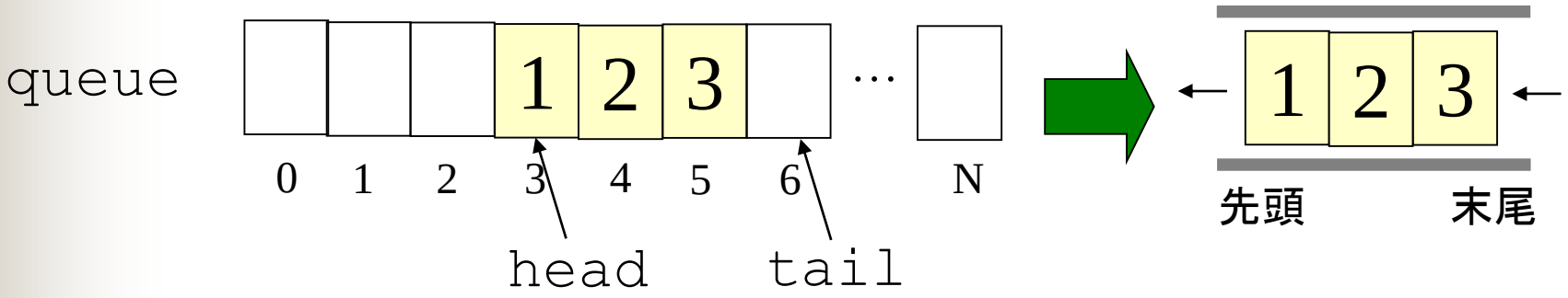
プログラム例(2)

- キューは一次元配列で表現
  - N個の一次元配列

```
int[] queue = new int[N];
```

※ キューに入る頂点の数は最大でもN個を上回らないため

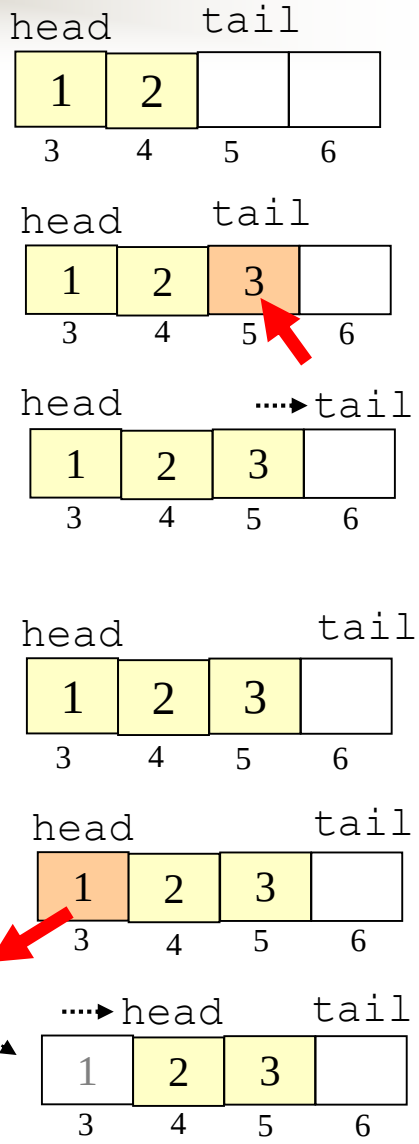
  - キューの先頭の位置を示す head と末尾を示す tailを導入する　　int head=0,tail=0;



head～tail-1 が現在のキューの内容

プログラム例(3)

- キューにデータを入れる
  - queue[tail]にデータを代入
  - tailを1つ右へ(tail++)
    queue[tail++]=data;
- キューからデータを取り出す
  - queue[head]の値を取り出す
  - headを1つ右へ(head++)
    data=queue[head++];
- head = tailの時、キューは空



プログラム例(4)　実際の処理部

```
void visit(){
    int i,j;
    queue[tail++]=1;v[1]=1;//頂点1をキューに入れる
    while(head < tail) {//キューが空でない
        i=queue[head++];//キューから1つ取り出す
        for (j=1; j<=N; j++) {
            if (g[i][j]==1 && v[j]==0) {
                System.out.println(
                    i+"->"+j);
                queue[tail++]=j;
                //頂点jをキューに入れる
                v[j]=1;//頂点jを訪問済みに
            }
        }
    }
}
```

頂点iと頂点jは隣接している

キューに入れる頂点を表示

頂点jは未訪問

プログラムリスト(全体)

```
public class GraphBreadthFirstSearch {
    int[][] g = {{0,0,0,0,0,0,0},
                 {0,0,1,1,0,0,0},
                 {0,1,0,0,1,0,0},
                 {0,1,0,0,1,0,1},
                 {0,0,1,1,0,1,0},
                 {0,0,0,0,1,0,0},
                 {0,0,0,1,0,0,0}};
    final int N = 6; // 頂点の数
    int[] v = new int[N+1]; // 訪問フラグ
    int[] queue = new int[N]; // キュー
    int head=0,tail=0;
    void visit(){ //頂点を訪問
        int i,j;
        queue[tail++]=1; v[1]=1;
        while(head < tail) {
            i=queue[head++];
            for(j=1;j<=N;j++) {
                if (g[i][j]==1 && v[j]==0) {
                    System.out.println(i+"->"+j+"を訪問");
                    queue[tail++]=j;
                    v[j]=1;
                }
            }
        }
    }
    public static void main(String args[]) {
        GraphBreadthFirstSearch prog = new GraphBreadthFirstSearch();
        prog.start();
    }
    public void start() {
        for (int i=1; i<=N; i++) { v[i]=0; }
        visit();
    }
}
```

プログラム例のトレース(1)

i	j	v	1	2	3	4	5	6	head	tail	queue	0	1	2	3	4	5	6	備考
			0	0	0	0	0	0	0	0									head=0,tail=0
			1	0	0	0	0	0	0	1		1							頂点1訪問
1									1	1									キューから頂点1を取り出す
	1																		g[1][1]=0
	2		1	1	0	0	0	0	1	2		1	2						g[1][2]=1,v[2]=0 頂点2訪問
	3		1	1	1	0	0	0	1	3		1	2	3					g[1][3]=1,v[3]=0 頂点3訪問
	4																		g[1][4]=0
	5																		g[1][5]=0
	6																		g[1][6]=0
2									2	3		1	2	3					キューから頂点2を取り出す
	1																		v[1]=1
	2																		g[2][2]=0
	3																		g[2][3]=0
	4		1	1	1	1	0	0	2	4		1	2	3	4				g[2][4]=1,v[4]=0 頂点4訪問
	5																		g[2][5]=0
	6																		g[2][6]=0

プログラム例のトレース(2)

i	j	v	1	2	3	4	5	6	head	tail	queue	0	1	2	3	4	5	6	備考
3									3	4		1	2	3	4				キューから頂点3を取り出す
	1																		v[1]=1
	2																		g[3][2]=0
	3																		g[3][3]=0
	4																		v[4]=1
	5																		g[3][5]=0
	6		1	1	1	1	0	1	3	5		1	2	3	4	6			g[3][6]=1,v[6]=0 頂点6訪問
4									4	5		1	2	3	4	6			キューから頂点4を取り出す
	1																		g[4][1]=0
	2																		v[2]=1
	3																		v[3]=1
	4																		g[4][4]=0
	5		1	1	1	1	1	1	4	6		1	2	3	4	6	5		g[4][5]=1,v[5]=0 頂点5訪問
	6																		g[4][6]=0

プログラム例のトレース(3)

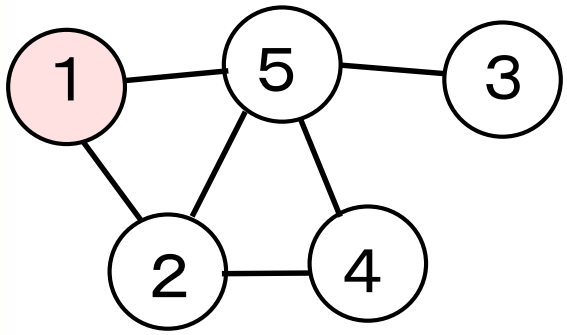
i	j	v	1	2	3	4	5	6	head	tail	queue	0	1	2	3	4	5	6	備考
6									5	6		1	2	3	4	6	5		キューから頂点6を取り出す
	1																		g[6][1]=0
	2																		g[6][2]=0
	3																		v[3]=1
	4																		g[6][4]=0
	5																		g[6][5]=0
	6																		g[6][6]=0
5									6	6		1	2	3	4	6	5		キューから頂点5を取り出す
	1																		g[5][1]=0
	2																		g[5][2]=0
	3																		g[5][3]=0
	4																		v[4]=0
	5																		g[5][5]=0
	6																		g[5][6]=0
									6	6									キューが空なので終了

プログラムリスト2(論理型変数を使用した場合)

```
public class GraphBreadthFirstSearch2 {
    final boolean F=false, T = true;
    boolean[][] g = {{F,F,F,F,F,F,F},
                     {F,F,T,T,F,F,F},
                     {F,T,F,F,T,F,F},
                     {F,T,F,F,T,F,T},
                     {F,F,T,T,F,T,F},
                     {F,F,F,F,T,F,F},
                     {F,F,F,T,F,F,F}};
    final int N = 6; // 頂点の数
    boolean[] v = new boolean[N+1]; // 訪問フラグ
    int[] queue = new int[N]; // キュー
    int head=0,tail=0;
    void visit(){ //頂点を訪問
        int i,j;
        queue[tail++]=1; v[1]=true;
        while(head < tail) {
            i=queue[head++];
            for(j=1;j<=N;j++) {
                if (g[i][j] && !v[j]) {
                    System.out.println(i+"->"+j+"を訪問");
                    queue[tail++]=j;
                    v[j]=true;
                }
            }
        }
    }
    public static void main(String args[]) {
        GraphBreadthFirstSearch2 prog = new GraphBreadthFirstSearch2();
        prog.start();
    }
    public void start() {
        for (int i=1; i<=N; i++) { v[i]=false; }
        visit();
    }
}
```

宿題

- 下図のグラフを幅優先探索で辿れ
  解答例: 1→5→4→3→2
- 下図のグラフを幅優先探索のプログラム例で探索した場合のトレース表を書け



i	j	v	1	2	3	4	5	6	head	tail	queue	0	1	2	3	4	5	6	備考
			0	0	0	0	0	0	0	0									head=0,tail=0
			1	0	0	0	0	0	0	1		1							頂点1訪問
1									1	1									キューから頂点1を取り出す
	1																		g[1][1]=0
	2		1	1	0	0	0	0	1	2		1	2						g[1][2]=1,v[2]=0 頂点2訪問
	3		1	1	1	0	0	0	1	3		1	2	3					g[1][3]=1,v[3]=0 頂点3訪問
	4																		g[1][4]=0
	5																		g[1][5]=0
	6																		g[1][6]=0
2									2	3		1	2	3					キューから頂点2を取り出す
	1																		v[1]=1
	2																		g[2][2]=0
	3																		g[2][3]=0
	4		1	1	1	1	0	0	2	4		1	2	3	4				g[2][4]=1,v[4]=0 頂点4訪問
	5																		g[2][5]=0
	6																		g[2][6]=0

解答例