アルゴリズムとデータ構造
演習第 13 回
グラフ

今日の授業で配られたハンドアウトを参考に、以下の問題を解いてください。

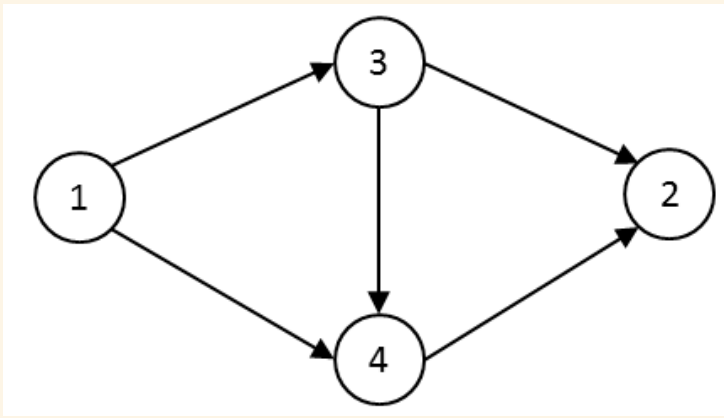
問題 1 [印刷用PDF]

1. 解答用紙に書かれた図1のグラフの隣接行列法による表現を書きなさい。

2. 図1のグラフに対してノード1から深さ優先探索を行った結果を書きなさい。各ノードの 発見時刻と終了時刻をグラフに書き込むこと。ただし、図1のグラフを表す隣接リストでは、 ノードは昇順に並んでいるものとする。 また、最初に訪問するノードの開始時刻を 1 とする。

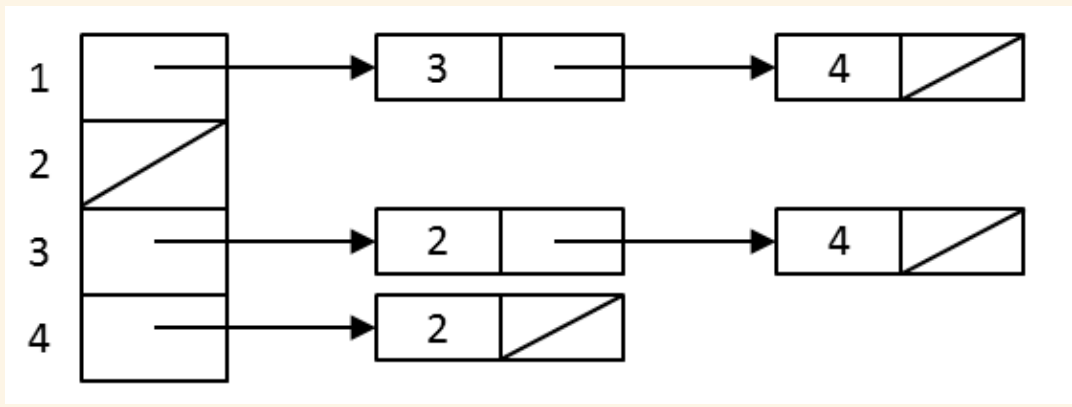
3. 図1のグラフに対してノード1から幅優先探索を行った結果を書きなさい。 各ノードvについてd[v]の値をグラフに書き込むこと。

以下のグラフについて考えます。



このグラフの隣接リスト法と隣接行列法による表現はそれぞれ以下のようになります。

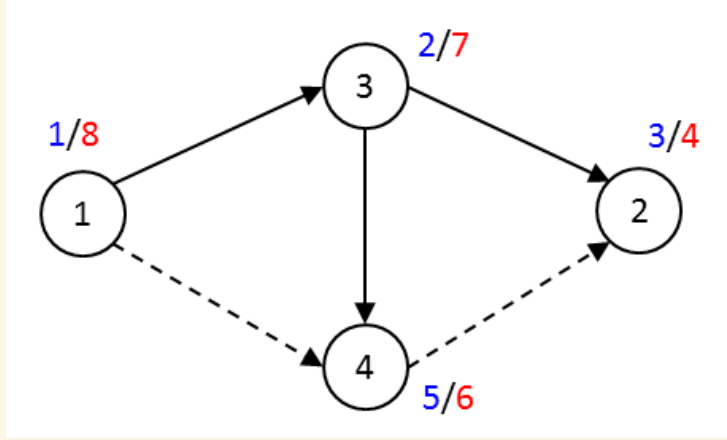
隣接リスト法による表現



隣接行列法による表現

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

このグラフに対して、 上で示した隣接リスト法による表現を使って深さ優先探索(DFS)を行った結果は以下のようになります。

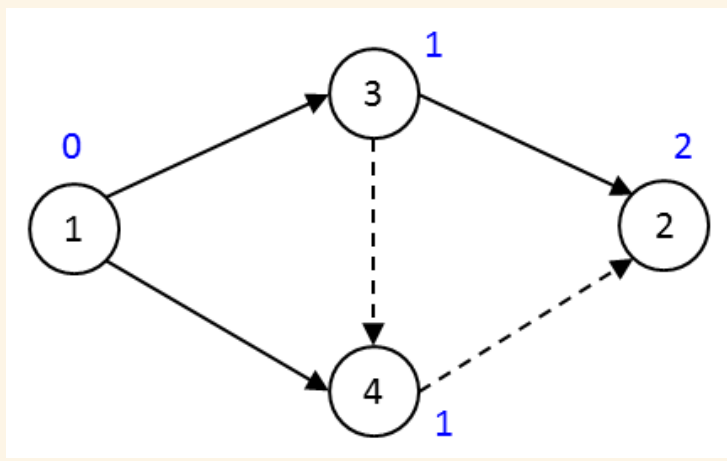


各ノード v の近くに書かれた青字がノードの発見時刻（d[v]）、 赤字がノードの終了時刻（f[v]）になります。 これらの時刻は以下のようにして記録されたタイムスタンプです。

- 発見時刻 d[v]: ノード v を最初に発見した時刻を記録する。
- 終了時刻 f[v]: ノード v の隣接リストを調べ終えた時刻を記録する。

図の実線は深さ優先探索のときに通った連結を、点線は通らなかった連結をそれぞれ表します。 深さ優先探索では、各ノード u の隣接リスト Adj[u] を先頭から順番に見ていきます。 そのため、上の例では 1 → 3 → 2 → 4 の順番でノードを訪問します。 ノード1からノード3を訪問したときにはまだノード2は未訪問（色がWHITE）なので、 ノード2からノード3を訪問します（このため、ノード3から2への連結は実線）。 ノード2から出て行く連結はないので、後戻り（バックトラック）してノード3に戻ります。 ノード3からノード4を訪問したときには、既にノード2は訪問済（色がWHITEでない）なので、 ノード4からノード2を訪問することはありません（そのため、ノード2から4への連結は点線）。

上のグラフに対して幅優先探索を行った結果(BFS)は以下のようになります。



各ノード v の近くに書かれた青字が d[v] の値になります。 図から分かる様に、d[u] の値は始点からの「距離」を表します。始点から出発して、 連結を 1 つ通るたびに距離が1増えます。

図の実線は幅優先探索のときに通った連結を、点線は通らなかった連結をそれぞれ表します。 幅優先探索では、はじめに始点となるノードをキューに入れます。 キューから取り出したノード u の隣接リスト Adj[u] を先頭から順番に見ていき、 未訪問のノード（色がWHITE）だけをキューに入れ、 その後ノード u を訪問済み（色が BLACK）にします。これをキューが空になるまで 繰り返します。

そのため、上の例では 1 → 3 → 4 → 2 の順番でノードを訪問します。 ノード1をキューから取り出したときには隣接するノード3と4はまだ未訪問（色が WHITE）なので、 ノード3と4をこの順番にキューに入れます（そのため、ノード1からノード3と4への連結はそれぞれ実線）。 このときに、ノード3と4の d[] の値をノード1の d[] の値（始点なので0）に 1 を加えた値にします。 次にキューから取り出されるのは3ですが、このときノード2は未訪問ですがノード4は訪問済みです。 したがって、ノード2だけをキューに入れます（そのため、 ノード3からノード2への連結は実線で、ノード3からノード4への連結は点線）。

問題 2

有向グラフの隣接リスト法による表現が与えられたとき、 その隣接行列法による表現を出力するプログラムを作成しなさい。 プログラムは以下の条件を満たすこと。 (ex13-2-skel.c)

- 入力として与えられた有向グラフの情報を読み、隣接リスト法によるデータ構造を作る。
- 有向グラフの情報は隣接リスト法の表現に沿って与えられる。 グラフのノード数を n としたとき、各ノードには1からnまでの番号がふられている。
- ノードの最大数は100とする
- プログラムが作った隣接リスト法によるデータ構造から、隣接行列法によるデータ構造を作る。
- 作成した隣接行列の内容を出力する。

注： 隣接リスト法に沿った表現が入力として与えられたとき、 実際には隣接リスト法によるデータ構造を作らずに、 隣接行列法によるデータ構造を直接作ることは可能です。 この問題では、隣接リスト法と隣接行列によるデータ構造の作り方を学ぶために、 いったん隣接リスト法によるデータ構造を作ってから 隣接行列法によるデータ構造を作っています。

問題１の例で示したグラフに対する実行例を示します(緑がプログラムへの入力、 赤がプログラムの出力)

```
% ./a.out
4
2 4 3
0
2 4 2
1 2
matrix
0 0 1 1
0 0 0 0
0 1 0 1
0 1 0 0
```

プログラムへの入力の意味は以下の通りです。

- １行目がグラフのノード数 n を表します。
- ２行目から n+1 行目までの各行について、 i+1 行目の先頭がノード i に隣接するノード数、 i+1 行目の2つ目以降（あれば）がノード i に隣接するノードの並びを表します。

実行例でプログラムが作る隣接リスト法によるデータ構造は、問題１の例で示したものになります。 プログラムへの入力では、ノードに隣接するノードの並びは降順で与えられていますが、 プログラムが作る隣接リストのデータ構造では、各ノードの隣接リストは逆の並び（昇順）になっています。 ex13-2-skel.c中の関数 insert を使って、 単純にプログラムを書くとそのようになるはずです。

実行例でプログラムが出力した隣接行列は、問題１の例で示したものになります。

隣接リストの初期化や隣接リストへのデータの挿入は、 第２回演習で使った関数 listinitialize と insert とほぼ同じものを使います（ex13-2-skel.cの中で定義されています）。 ただし、第２回演習とは違って今回は各ノードについてリストを用意するので、 変数 head と tail はグローバル変数ではなく、構造体 list のメンバーになっています。 そのため、関数 listinitialize と insert には初期化や挿入を行う対象となる隣接リストを表す構造体 list へのポインタを与える必要があります。

> 注意: ノード番号と添字の対応付けについて
> 入力ではノードには1から n までの番号がふられていますが、C言語では配列の添字は0から始まります。 そのため、入力値と隣接リストや隣接行列のデータ構造を表す配列の添字は１つずれることに注意してください。

問題３

与えられた有向グラフに対して深さ優先探索を行うプログラムを作成しなさい。 プログラムは以下の条件を満たすこと。 (ex13-3-skel.c)

- 入力として与えられた有向グラフの情報を読み、隣接リストのデータ構造を作る。
- 有向グラフの情報は隣接リスト法の表現に沿って与えられる。 グラフのノード数を n としたとき、各ノードには1からnまでの番号がふられている。
- ノードの最大数は100とする
- プログラムは各ノードの発見時刻と終了時刻を報告する。 ただし、最初に訪問するノードの開始時刻を1とする。

問題１の例で示したグラフに対する実行例を示します(緑がプログラムへの入力、 赤がプログラムの出力)

```
% ./a.out
4
2 4 3
0
2 4 2
1 2
timestamps
1: 1 8
2: 3 4
3: 2 7
4: 5 6
```

プログラムへの入力の意味は問題２と同じです。 プログラムは各ノードの発見時刻と終了時刻を１行に１つずつ、ノードにふられた番号の順番に出力します。 各行の内容は先頭から順にノード番号 i、ノード i の発見時刻、ノード i の終了時刻です。

入力処理は、問題２で書いたものをそのままコピーしてかまいません。 その場合は問題２と同様に、プログラムが作る各ノードの隣接リストは、 入力で与えられた隣接リストの表現と逆の並びになります（上の実行例もそのようになっています）。
