# 人工知能･音声処理実験
# 作業日報

# 第1回 AI-1

学生番号:09430509

氏名:今田　将也

実施日:2020年10月29日

# AI-1の概要

問題1

- ソースコードを示しながら，スカラー，ベクトル（配列），行列，構造体について，簡潔に説明
- 制御文として，while文とif文の使い方を，実例をあげながら，簡潔に説明
- ソースコードを示しながら，スタックやキューの動作例を示した．
- スタックとキューの共通点や差異について，機能面や実装面から考察
- ソースコードを示しながら，1つ以上のグラフを表現
- 人工知能実験の課題を進めるために用いた，MATLAB/Octaveの「演算子」や「組み込み関数」の一覧を作成

問題2

- 深さ有線探索と幅優先探索のソースコードを，それぞれ示した
- 探索対象となるグラフを1つ以上用意し，探索結果を示した
- 実装や結果の，共通点や差異に注目して，二つの探索方式の，実装面と機能面について考察

その他

- (その他，作業したことがあればまとめておく)

# 問題1

MATLAB/Octaveを用いて，次の3つの課題に取り組みなさい．

1. MATLAB/Octaveの基本的な使い方を理解しなさい．
2. スタックとキューの動作を確認し，説明しなさい．
3. 行列を利用して，グラフを表現する方法について，説明しなさい．

# 問題1の小問題

1. MATLAB/Octaveのソースコードを示しながら，スカラー，ベクトル（配列），行列，構造体について，簡潔に説明せよ.
2. MATLAB/Octaveにおける制御文として，while文とif文の使い方を，実例をあげながら，簡潔に説明せよ.
3. MATLAB/Octaveのソースコードを示しながら，スタックやキューの動作例を示せ.
4. スタックとキューの共通点や差異について，機能面や実装面から考察せよ.
5. MATLAB/Octaveのソースコードを示しながら，1つ以上のグラフを表現せよ.
6. 人工知能実験の課題を進めるために用いた，MATLAB/Octaveの「演算子」や「組み込み関数」の一覧を作成せよ.

※実施していないものは青字とした

# 1-1. MATLAB/Octaveの基本的な使い方(1/2)

**スカラー**

変数は，代入した時点で，暗黙的に宣言される．

- x = 10;
- y = x * 3;
- x  % x=10
- y  % y=10
- y = z * 3

宣言されていない変数を参照すると，エラーが表示される

**配列（ベクトル）**

配列の場合も，スカラーと同じように，変数に代入して利用できる

- z1 = [10 20 30] %配列

ベクトルの長さや，行列のサイズは，length や size という関数で得られる．

- length(z1)　% ans = 3
- size(z1)　% ans = 1 3

MATLABの場合，原則として**すべての変数は行列である．**

**行列の添え字（インデックス）は1から**始まる．

- x = [100 200 300];
- x(4)  % errorになる
- x(4) = 400　% 代入はできる．大きさが3から4へ拡張される．

# 1-1. MATLAB/Octaveの基本的な使い方(2/2)

## 行列

### 行列のスライス

- A = [10 20 30 40; 50 60 70 80; 90 100 110 120]
- A(2, 3)　% 70 と表示される
- A(:, 3)　% 30 70 110 が縦並びで表示される
- A(2, :)　% 50 60 70 80 が横並びで表示される

### 行列とスカラーの演算

- A = [1 2 3; 4 5 6]　% [1 2 3] の下に [4 5 6] が並ぶ行列
- C = 2　% C = 2　スカラー
- Z = A * C
- % Z =　2　4　6
- %　　8 10 12　Aの全ての要素にCが掛けられている

## 構造体

- node = struct('id', 1, 'parent', 3);% C言語と同じように参照できる
- node.id　　% ans = 1
- node.parent　% ans = 3

## 構造体の配列

### 配列の要素として構造体を格納するイメージ

- nodes(1) = struct('id', 1, 'parent', 3);
- nodes(2) = struct('id', 2, 'parent', 5);
- nodes(2).id　　% ans = 2が出力される
- nodes(1).parent　% ans = 3が出力される

# 1-2. MATLAB/Octaveにおけるwhile文とif文の使い方(1/2)

if文

式を評価し，式が真 (true) であるときに一連の動作が実行される．結果が空でなく，非ゼロの要素 (論理値または実数値) のみが含まれる場合に，式は true．それ以外の場合はfalse.

else ifではなくelseifとくっつける必要があることに注意

```
n = 100;
if n > 10
    disp('n > 10')
elseif n > 5
    disp('10 > n > 5')
else
    disp('5 > n')
end   % "n > 10“
```

# 1-2. MATLAB/Octaveにおける while文とif文の使い方(2/2)

While文

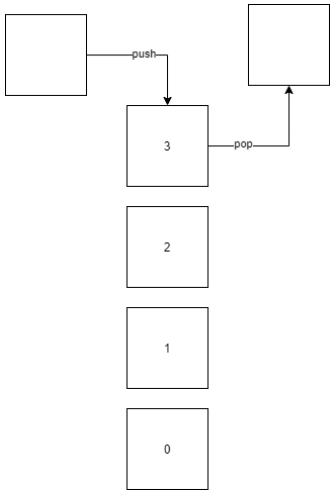
while 文は，if 文と同じように，条件式を書く．条件を満たすと end まで実行される，という点も同じ．ただし，while は end まで辿り着くと，条件式の判定行にジャンプする．

- lim = 1000;
- x = 2;
- while x <= lim
- 　x = x * 2;
- end
- x　% x = 1024

# 1-3.スタックやキューの動作例(1/3)

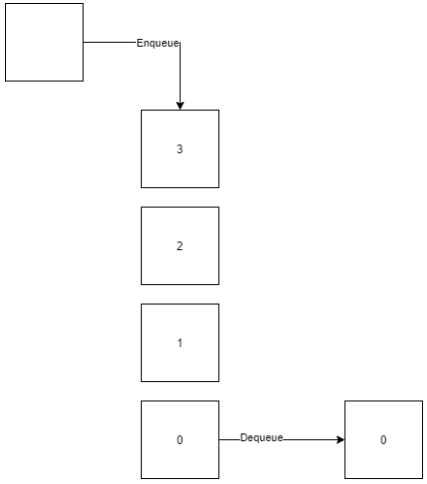
スタック

要素の挿入と削除がリストの先頭だけで行われる LIFO のデータ構造．LIFO (Last In, First Out) とは「最後に入ったものが最初に出てゆく」という意味．




キュー

キューはリストの一方の端で挿入が行われ反対の端で削除が行われる FIFO の構造．FIFO (First In, First Out) とは「最初に入ったものが最初にでてゆく」という意味．



# 1-3.スタックやキューの動作例(2/3)

**MATLABでのスタック動作ソースコード**

```
% (1) Initialize
stack_data = [];
% push
stack_data = [1 stack_data];
stack_data = [2 stack_data];
stack_data = [3 stack_data];
stack_data = [4 stack_data];
stack_data
% pop
ret = stack_data(end);
stack_data(end) = [];
ret
stack_data
```

**実行結果**

```
>> stack
stack_data =
   4   3   2   1
ret =  1
stack_data =
   4   3   2
```

# 1-3.スタックやキューの動作例(3/3)

**MATLABでのキュー動作ソースコード**

```
% (1) Initialize
q_data = [];
% push
q_data = [1 q_data];
q_data = [2 q_data];
q_data = [3 q_data];
q_data = [4 q_data];
q_data
% pop
ret = q_data(1);
q_data(1) = [];
ret
q_data
```

**実行結果**

```
>> queue
q_data =
  4   3   2   1
ret =  4
q_data =
  3   2   1
```

# 1-4.スタックとキューの共通点や差異について(機能面)

共通点

- スタックもキューも以下のような機能を持つデータ構造のこと
  - 要素 x をデータ構造に追加する
  - データ構造から要素を取り出す

差異

- スタックはデータ構造に入っている要素のうち、最後に push した要素を取り出す
- キューデータ構造に入っている要素のうち、最初に push した要素を取り出す

# 1-4.スタックとキューの共通点や差異について(実装面)

共通点

- どちらも要素を取り出す，要素を追加するという実装を行う必要がある

差異

- スタックは要素を取り出す際は一番最後に追加されたものの位置を記憶しておき，取り出し，要素数を1減らす
- キューは要素を取り出す際に一番最初に追加されたものを取り出すため，先頭の要素を記憶しておき，取り出し，要素数を1減らす

# 1-5.グラフを表現(1/2)

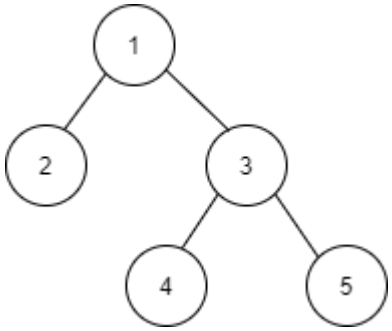
グラフを表現するには隣接行列を表現すれば良い.

**隣接行列（Adjacency matrix）**とは，「あるノードと別のあるノードが接続しているか否か」という情報を表現した行列．あるノードとあるノードの接続に注目し，それらが接続している場合は1，接続されていない場合は0の要素を持つ.

- 無向グラフでは，各ノード間を互いに行き来できるように1が指定された行列を定義すれば良い
- 有効グラフでは，上から下にのみ遷移しうる木構造であれば目的とは違う方向には進めないように0が指定された行列を定義すれば良い

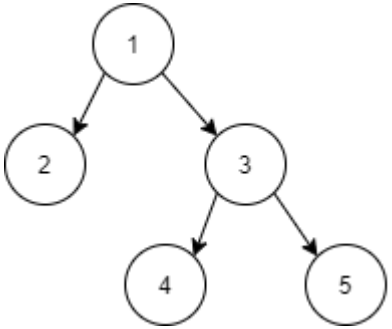
# 1-5.グラフを表現(2/2)

無向グラフの表現例




```
A = [
      0 1 1 0 0; % 1 -> 2 or 3
      1 0 0 0 0; % 2 -> 1
      1 0 0 1 1;
      0 0 1 0 0;
      0 0 1 0 0
    ];
```

有向グラフの表現例




```
A = [
      0 1 1 0 0; % 1 -> 2 or 3
      0 0 0 0 0; % 2 -> NULL
      0 0 0 1 1; % 3 -> 4 or 5
      0 0 0 0 0; % 4 -> NULL
      0 0 0 0 0  % 5 -> NULL
    ];
```

# 1-6.「演算子」や「組み込み関数」の一覧(1/2)

**基本的な算術演算**

- 加算 +
- 減算 -
- 乗算 .*
- 行列乗算　*
- 配列の右除算　./
- 要素単位のべき乗　.^
- 行列のべき乗　^
- 除算後の剰余 (モジュロ演算) mod
- 最も近い小数または整数への丸め round

**関係演算子**

- 等価性の判定 ==
- 以上かどうかの判別 >=
- より大きいかどうかの判別 >
- 以下かどうかの判別 <=
- 未満かどうかの判別 <
- 不等価の判定 ~=
- 配列の等価性を判別 isequal

# 1-6.「演算子」や「組み込み関数」の一覧(2/2)

**組み込み関数の一例**

- length　　最大の配列の次元の長さ
- size　　配列サイズ
- ndims　　配列の次元数
- numel　　配列の要素数
- isempty　　空かどうか
- ismember　　その要素が配列内にあるか

# 問題1のまとめ

MATLAB/Octaveを用いて，次の3つの課題に取り組みなさい．

1. MATLAB/Octaveの基本的な使い方を理解しなさい．
2. スタックとキューの動作を確認し，説明しなさい．
3. 行列を利用して，グラフを表現する方法について，説明しなさい．

まとめ

- MATLABの基本的な使い方を整理しつつ，演算子を用いて，キューとスタックをMATLABで実現した．
- MATLABでは配列の扱いが簡単で，すぐに実装できた
- グラフの表現も行列を用いることで，有向グラフと無向グラフを実装できることを確認した

# 問題2

グラフ探索における“深さ優先探索”と“幅優先探索”を，MATLAB/Octaveで実装しなさい．さらに，その探索方式や実装について，問題1の説明と関連付けて，考察しなさい．

# 問題2の小問題

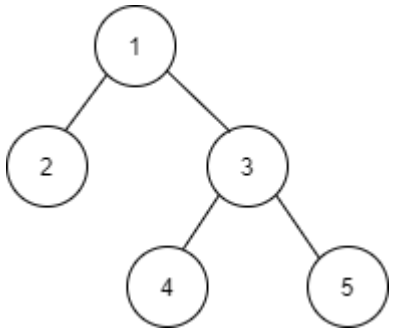
1. 深さ有線探索と幅優先探索のソースコードを，それぞれ示しなさい．
2. 探索対象となるグラフを1つ以上用意し，探索結果を示しなさい．
3. 実装や結果の，共通点や差異に注目して，二つの探索方式の，実装面と機能面について考察せよ．

※実施していないものは青字とした

# 2-1幅優先探索ソースコード

探索をするグラフは以下のグラフである.

スタートは1，ゴール地点は5である.




```
% Example of Graph A
A = [0 1 1 0 0;
     1 0 0 0 0;
     1 0 0 1 1;
     0 0 1 0 0;
     0 0 1 0 0
];
initial_node = 1;
target_node  = 5;
```

# 2-1幅優先・深さ優先探索ソースコード

```
% 初期状態をオープンリストに入れる
open_list = [ initial_node ];
% クローズドリストを空にする
closed_list = [];

while ~isempty(open_list)

  cur_node = open_list(1)
  open_list(1) = [];
  closed_list = [closed_list cur_node]

  if cur_node == target_node
    disp('I got a Target node!')
    break;
  end

  hyp_nodes = find(A(cur_node, :) > 0)

  % closed_list実装により，展開済みのノードを再びオープンリストに格納しないようにする
  if ~isempty(closed_list)
    hyp_nodes = hyp_nodes(~ismember(hyp_nodes, closed_list));
  end

  open_list = [ open_list hyp_nodes ]

end
```

左に示したのは幅優先探索のソースである.

深さ優先にするには，最後の行の，open_listの末尾に追記している動作を，先頭に追記するように変更を加えれば良い.

# 2-2探索結果

探索に用いたグラフは2-1で示したソースコードで用いたものと同様である.

用いたグラフが余り特徴的でなく幅優先，深さ優先ともに実行結果がおなじになってしまった.

```
>> haba

cur_node =  1
closed_list =  1
hyp_nodes =

   2   3

open_list =

   2   3

cur_node =  2
closed_list =

   1   2

hyp_nodes =  1
open_list =  3
cur_node =  3
closed_list =

   1   2   3

hyp_nodes =

   1   4   5

open_list =

   4   5

cur_node =  4
closed_list =

   1   2   3   4

hyp_nodes =  3
open_list =  5
cur_node =  5
closed_list =

   1   2   3   4   5

I got a Target node!
```

```
>> hukasa

cur_node =  1
closed_list =  1
hyp_nodes =

   2   3

open_list =

   2   3

cur_node =  2
closed_list =

   1   2

hyp_nodes =  1
open_list =  3
cur_node =  3
closed_list =

   1   2   3

hyp_nodes =

   1   4   5

open_list =

   4   5

cur_node =  4
closed_list =

   1   2   3   4

hyp_nodes =  3
open_list =  5
cur_node =  5
closed_list =

   1   2   3   4   5

I got a Target node!
>>
```

# 2-3二つの探索方式の共通点

共通点

- オープンリストとクローズドリストを用いた
- 初期ノードと目標ノードを示した
- 最悪の場合の時間計算量はO(グラフの辺の数)
- 最悪の場合の空間計算量はO(グラフのノード数)

# 2-3二つの探索方式の差異

差異

- 幅優先はオープンリストの配列の末尾に追加する
  - キューを使う
  - 大量の情報から探索する事に向かない
- 深さ優先はオープンリストの配列の先頭に追加する
  - スタックを使う
- メモリに載りきらないような大規模な木を探索する場合、深さ優先探索は探索木のパスの長さが長くなりすぎて探索が終わらないという問題を抱えている

# 問題2のまとめ

グラフ探索における“深さ優先探索”と“幅優先探索”を，
MATLAB/Octaveで実装しなさい．さらに，その探索方式や実装について，
問題1の説明と関連付けて，考察しなさい．

まとめ

- 幅優先と深さ優先をOctaveで実装を行い，その結果のソースコードを示した．
- 探索対象となるグラフを用意し，その探索結果を記載した．
- 両者の探索方法の共通点や差異を，実装面や機能面に着目しその違いについて理解を深めた

# その他

グラフが悪く，結果が一緒になったので別のグラフで実行したい

# AI-1の感想

触れたことのない，Octaveというソフトとインタプリタのような言語を用いたプログラムで最初は混乱した.

しかし，充実した資料のおかげで今回のすべての実験を，時間は過ぎてしまったが完了させることができた.

幅優先探索も深さ優先探索については，両者とも同じ結果になってしまったが，次回とその次のミニレポートまでには異なるグラフを用いて結果を示したい.

次回は，発展課題でA*アルゴリズムなどをすると見たので次回に向けて予習と復習をしておきたい.