アルゴリズムとデータ構造II

講義3：

グラフ、定義、表現

https://elms.u-aizu.ac.jp

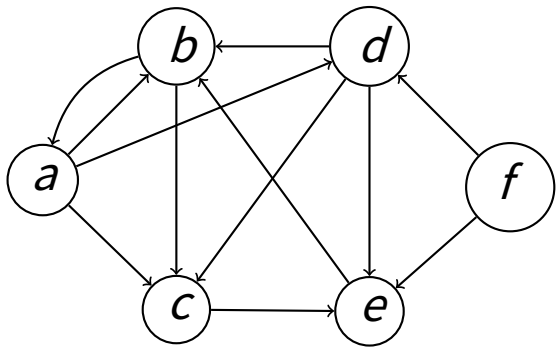
有向グラフ

私 グラフ *G*（*V*、*E*）頂点（ノード）のセットで構成されます

*V*とエッジのセット *E*。 エッジが 順序対

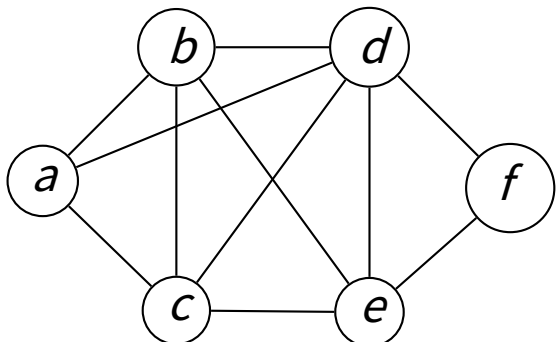
（（*v*、*w*）頂点の場合、グラフは次のようになります。 指示 （エッジは

アークとも呼ばれます）



無向グラフ

私 エッジが 順序付けられていないペア （セット）異なる
頂点（（v、w））グラフは
無向。



グラフ表現

私 有向グラフ G（V、E）、if（v、w）のエッジです E
次に頂点と言います w に隣接しています v。エッジ
とも言います（v、w）から v に w。隣接する頂点
の数 v それは（アウト）次数 の v。

私 無向グラフで G（V、E）、（w、v）および（v、w）です
同じエッジ。w に隣接しています v if（v、w）にあります E。
頂点の次数は、それに隣接する頂点の数です。私たちはエッジを言います（v、w）に事件があります v。

道

私 A 道 グラフには、フォームの一連のエッジがあります

（（v_1、v_2）、（v_2、v_3）、。。。。、（v_{n-1}、v_n）。パスはからです v_1 に v_n 長さの $n-1.1$。

私 特別な場合として、単一の頂点はのパスを示します

それ自体からそれ自体までの長さ0。

シンプルなサイクル

私 パスは シンプル 上のすべてのエッジとすべての頂点の場合
おそらく最初と最後の頂点を除いて、パスは区別されます。

私 A サイクル 少なくとも1の長さの単純なパスであり、
同じ頂点で開始および終了します。無向グラフでは、サイクルの長さは少なくとも3でなければなりません。

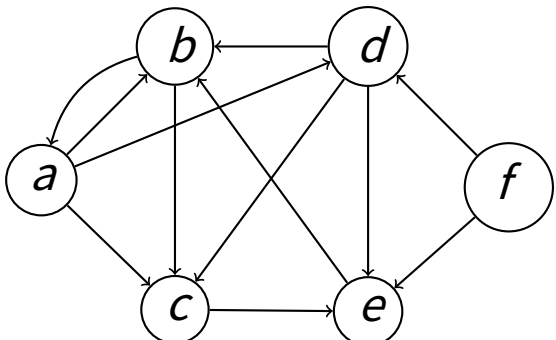
隣接行列

私 グラフの一般的な表現の1つ

G（V、E）それは 隣接行列、$|V| \times |V|$ マトリックス A 0と1の、ここで $A[i, j] = 1$ i ff頂点からのエッジがあります 私 頂点へ j。 隣接行列には $O(|V|^2)$ グラフのメモリスペース G（V、E）。

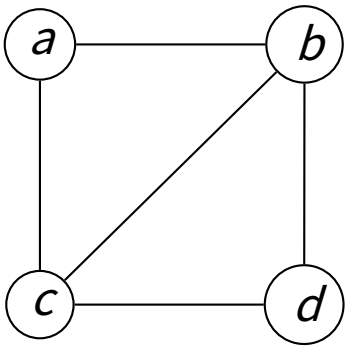
グラフと隣接行列

	a	*b*	*c*	*d*	*e*	*f*
a	0	1	1	1	0	0
b	1	0	1	0	0	0
c	0	0	0	0	1	0
d	0	1	1	0	1	0
e	0	1	0	0	0	0
f	0	0	0	1	1	0



グラフと隣接行列

	あいうえお
a	0 1 1 0
b	1 0 1 1
c	1 1 0 1
d	0 1 1 0



隣接リスト

私 グラフの別の可能な表現は次のとおりです。

隣接リスト。

私 頂点の隣接リストは、すべての頂点のリストです。

それに隣接しています。グラフは|で表すことができますV|

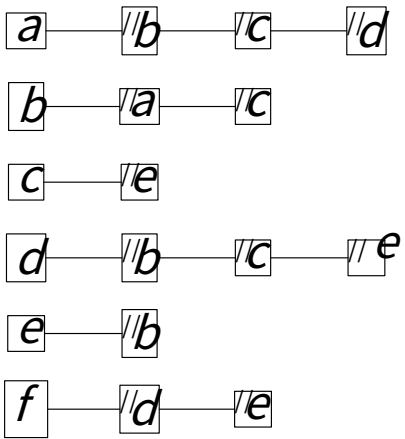
隣接リスト（頂点ごとに1つ）。

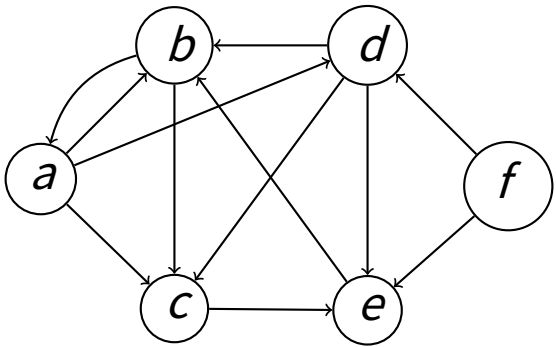
隣接リストの表現

私 隣接リストには $O(|V|+|E|)$ メモリスペース
グラフ用 G（V、E）。隣接リストは、まばらなグラフによく使用されます G（V、E）と$|E|$ $|V|^{2.2}$。

私 注意してください$|E| \leq |V|(|V|-1)$ 有向グラフと
$|E| \leq (|V|(|V|-1))/2$無向グラフの場合。

グラフと隣接リスト





検索プロセス

グラフGの表現が確立されると、グラフのタイプ（有向または無向）は次の検索プロセス、DFSおよびBFSでは重要ではありません。隣接行列またはリストで処理するだけです。

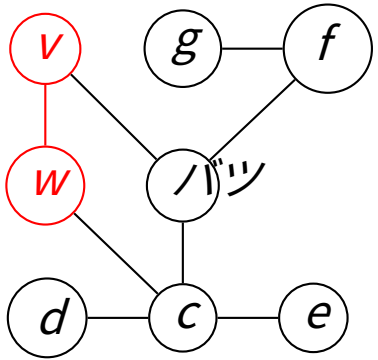
深さ優先探索

私 深さ優先探索（DFS） する自然な方法です
訪問 すべての頂点とすべてのエッジを体系的に
グラフ チェックします。

私 それは持っていま 次のような多くのグラフ問題のアプリケーション
接続性の確認、接続されたコンポーネントまたは
サイクルの検索などをグラフで行います。

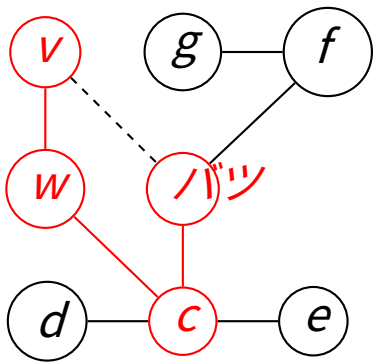
DFSの最初のステップ

私 DFS訪問グラフ G（V、E）次のように。選択する
頂点 v と訪問 v。（v のルートとも呼ばれます
DFS検索ツリー。）　次に、任意のエッジを選択します（v、w）
事件 v そして　訪問 w。



DFSの2番目のステップ

私 一般的に、バツ最近訪れたのは
バーテックス。未踏のエッジを選択して検索を続
行します（x、y）に事件 バツ。



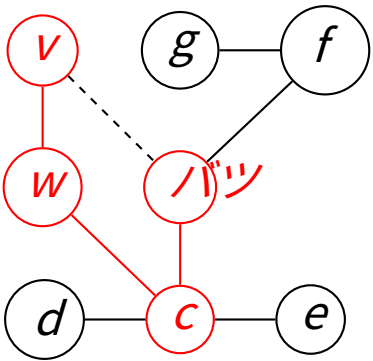
DFSの次のステップ

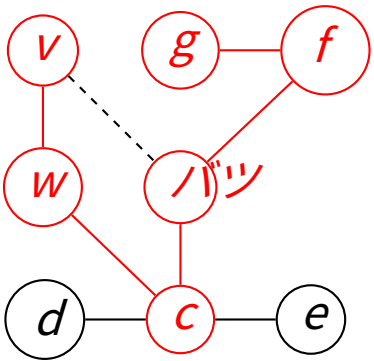
私 場合 y 以前に訪問したことがありますが、別の場所を見つけます

新しいエッジインシデント バツ。 場合 y 以前にされたことはありません

訪問した後、訪問します y そして始めます　新たに検索から

バーテックス y。

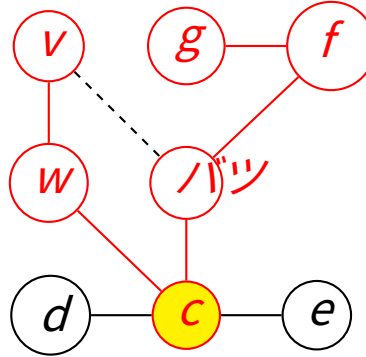




に戻る c

私 すべてのパスで検索を完了した後

から始まる y、 検索はに戻ります バツ、 そこからの頂点 y 最初に到達しました。

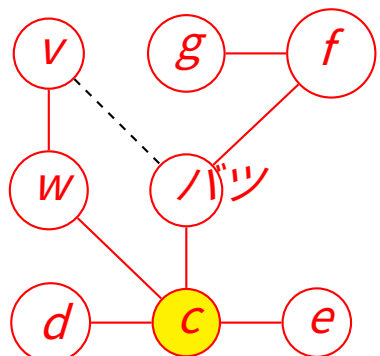


最後に...

私 に入射する未踏のエッジを選択するプロセス

バツ これらのエッジのリストが

疲れ果てた。



DFS

私 このメソッドは呼び出されます 深さ優先探索 私たち以来
可能な限り前方（より深い）方向に検索を続けます。

DFS表現

私 以下は、Cで記述された再帰的プロシージャです。
これは、隣接行列で表されるグラフのDFSを実現します。

私 手順で $val[|V|+1]$は、
頂点にアクセスしました。最初は$val[|V|+1]$はすべてゼロに設定されているため、 $val[k]=0$、頂点を示します k まだ訪問されていません。

DFSコード

私 頂点の場合 *k* それは *私*-訪問した頂点、次に *val [k]* に設定されています

私、1 ≤ *私* ≤|*V*|。

```
DFS（int k）{
   int t;
   val [k] = ++ i;
   for（t = 1; t <= | V |; t ++）
      if（A [k] [t] ! = 0 && val [t] == 0）DFS（t）;
}
```

訪問順または逆順によるノードの番号付け

私 次のような順序（または何らかの方法）にアクセスしてノードに番号を付ける

$v=1$、 $w=2$、 $c=3$、。。。、$g=6$、 $d=7$、 $e=8$または

$v=8$、 $w=7$、 $c=6$、。。。、$g=3$、 $d=2$、 $e=1$

強く接続されたコンポーネントを見つけたり、アーティキュレーションポイントを見つけたりするアプリケーションで役立つことがあります。

幅優先探索

私 別の古典的なグラフ走査アルゴリズムは

私 幅優先探索（BFS）

BFSは次のようにグラフを検索します。

1.1。 頂点を選択します *v* と訪問 *v*。

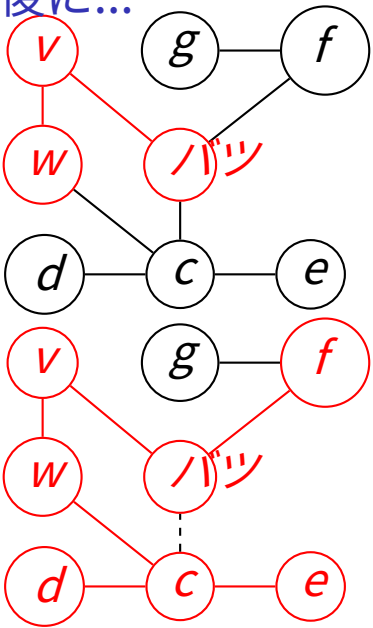
2.2。 すべてのエッジ（*v*、*w*）、 頂点にアクセス *w* そして、置きます *w* 先入れ先出し（FIFO）キューに入れます。

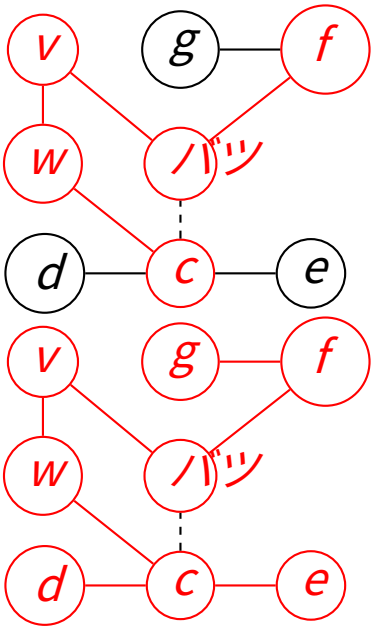
3.3。 後 *w*のすべての人が訪問します（*v*、*w*）、 頂点を選択します *w* FIFOキューから、未訪問のすべてにアクセスします

頂点 バツにとって（*w*、*x*）、 そして、置きます バツキューに入れます。

4.4。 グラフ内のすべての頂点にアクセスするまで、上記のプロセスを繰り返します。

最後に...





BFS

私 BFSは、以前に訪問した頂点のすべての後続を訪問します
それらの後継者のいずれかの後継者を訪問します。

私 これは、訪問するDFSとは対照的です。
その「兄弟」のいずれかを訪問する前に、訪問した頂点の後継者。

DFSとBFS

私 DFSは非常に長くて狭いツリーを作成する傾向がありますが、

BFSは、幅が広く短いツリーを作成する傾向があります。

私 演習問題2は、BFS用のCプログラムを提供します。

グラフの接続性

私 しましょう G（V、E）である 指示 グラフ。2つの頂点uそして vの Gです 強く結びついている からのパスがある場合 u に vとからのパス vに u。

私 A 強連結成分 の Gは 最大 サブグラフ の Gその頂点はすべて互いに強く接続されています。

強く接続されたコンポーネントを見つける

私 強く結びついているものを見つける問題
のコンポーネント G DFSは次のように実行できます。

1.1。 最初のステップは、でDFSを実行することです。 G。

2.2。 次のステップは、のすべてのエッジを反転することです G 作成
逆グラフ G_r、隣接する転置を取る
の行列 G。

3.3。 最後に、DFSオン G_r DFSで指定された最も低い
ラベルを持つ頂点から開始して実行されます
オン G。

強く接続されたコンポーネントの検索2

私 この検索で のすべての頂点にアクセスしない場合 G_r、インクルード

ラベルが最も低い未訪問の頂点が選択され、
そこで検索が再開され、この方法でのすべての頂点
まで続行されます。 G_r訪問されました。

接続済み、k接続済み

私 無向グラフ G と呼ばれる 接続済み もしあれば
の頂点の任意のペア間のパス G。

私 Gは呼ばれます k-いずれかの除去の場合に接続されます $k-1$
頂点は残りのサブグラフを接続したままにします。

私

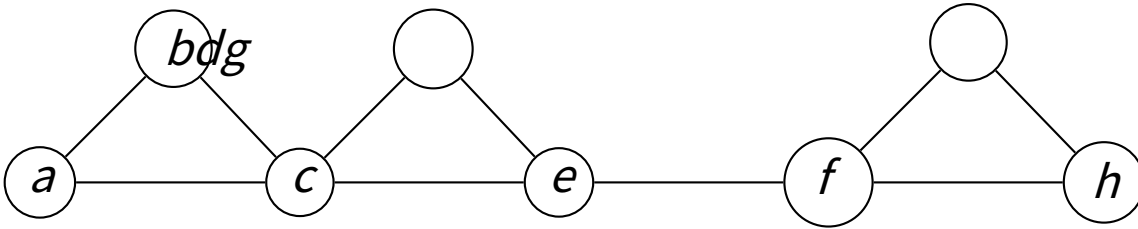
私 1連結（2連結）また、1つの頂点を意味します
2接続されています。
失敗は許容できます。

私 グラフが接続されているが二重接続されていない場合、
アーティキュレーションポイント：削除するとグラフが切断される頂点。

アーティキュレーションポイントの例

私 たとえば、グラフのアーティキュレーションポイントは次のとおりです。

c、*e*、そして *f*。 アーティキュレーションポイントは、 DFS。



アーティキュレーションポイントを見つける

私 たとえば、グラフのアーティキュレーションポイントは次のとおりです。

c、*e*、 そして *f*。 アーティキュレーションポイントは、 DFS。