「オブジェクト指向プログラミング特論」

2020 年度期末レポート課題

締切:2020/8/18

1　Eulr 閉路の列挙

頂点を弧で結んだものがグラフである。弧に向きの無いグラフを無向グラフという。頂点の集合を V、弧の集合を A とし、グラフを $G = (V, A)$ と表す。頂点 $v \in V$ に接続する弧の集合を δv とする。弧 $a \in A$ の両端の頂点の集合を ∂a とする。

無向グラフに対して、全ての弧を一度ずつ経由し、始点に戻る閉路（閉じた弧の列）を Euler 閉路という。Euler 閉路を列挙する再帰的なアルゴリズムを Algorithm 1 に示す。

Algorithm 1 Euler 閉路を列挙する再帰アルゴリズム (r は始点)

```
 1: procedure ENUMERATE(v, A_Euler)
 2:     if v = r ∧ |A_Euler| = |L| then
 3:         見つかった Euler 閉路を保存
 4:     else
 5:         for all a ∈ δv do                          ▷ v に接続するすべての弧
 6:             if a ∉ A_Euler then
 7:                 A'_Euler = A_Euler ∪ {a}           ▷ a を追加したリストを新たに生成
 8:                 w = ∂a \ {v}                       ▷ a の v と反対側の頂点
 9:                 ENUMERATE(w, A'_Euler)
10:             end if
11:         end for
12:     end if
13: end procedure
```

再帰的なアルゴリズムは、簡潔に記述できるという利点がある一方で、実際の動作が分かりにくく、停止条件が正しくない場合には、暴走の危険がある。そこで、Euler 閉路を

列挙する非再帰的アルゴリズムを考える。対象となるグラフを図 1 に示す。

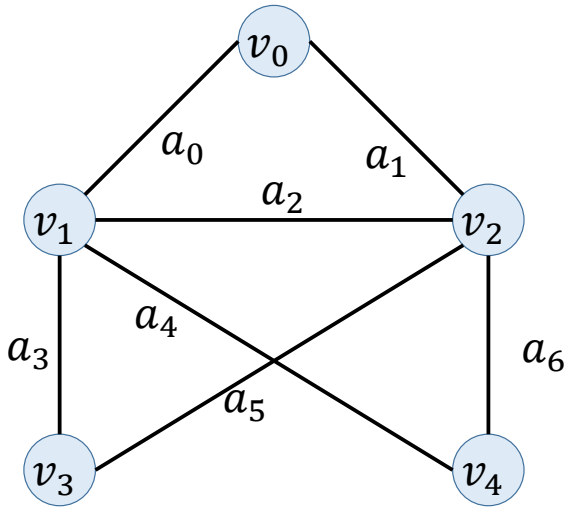

図 1　グラフの例

再帰的アルゴリズムを非再帰的に書き直す際によく使う方法は、再帰によって待たせているタスクを管理する方法である。今回は、始点から開始した経路を待ち行列に入れて、管理することにする。また、Algorithm 1 は、深さ優先で列挙をしているが、幅優先、つまり始点からの距離を少しずつ長くすることで列挙することにする。

始点からの弧の列 A_{Euler} と、その経路で到達した頂点 w の組 (A_{Euler}, v) を待ち行列 Q で管理する。グラフ (図 1) に対して、v_0 から開始すると、Q の初期値として以下のように設定する。

$$Q = \langle([a_0], v_1), ([a_1], v_2)\rangle \tag{1.1}$$

次に、Q から先頭要素を取り出し、頂点 v_1 から次の頂点へと移動する経路を見つけ、それを Q の末尾に追加する。

$$Q = \langle([a_1], v_2), ([a_0, a_2], v_2), ([a_0, a_3], v_3), ([a_0, a_4], v_4)\rangle \tag{1.2}$$

同様に先頭の $([a_1], v_2)$ を取り出し、次の頂点へと移動する経路を見つけ、それを Q の末

尾に追加する。

$$Q = \langle([a_0, a_2], v_2), ([a_0, a_3], v_3), ([a_0, a_4], v_4), \\ ([a_1, a_2], v_1), ([a_1, a_5], v_3), ([a_1, a_6], v_4)\rangle \tag{1.3}$$

以上を繰り返すと、やがて経路として Euler 閉路が出現する。Euler 閉路となったものは待ち行列に戻さず、他のリストへ保存する。全ての Euler 閉路が見つかると、待ち行列が空になる。アルゴリズムとしてものを Algorithm 2 に示す。

Algorithm 2 Euler 閉路を列挙する非再帰アルゴリズム (r は始点)

```
1:  while |Q| > 0 do
2:      (A_Euler, v) = Q.poll()                    ▷ Q から先頭を取り出す
3:      if v = r ∧ |A_Euler| = |L| then
4:          見つかった Euler 閉路を保存
5:      else
6:          for all a ∈ δv do                      ▷ v に接続するすべての弧
7:              if a ∉ A_Euler then
8:                  A'_Euler = A_Euler ∪ {a}       ▷ a を追加したリストを新たに生成
9:                  w = ∂a \ {v}                   ▷ a の v と反対側の頂点
10:                 (A'_Euler, w) を Q の末尾に追加
11:             end if
12:         end for
13:     end if
14: end while
```

2 プログラムの作成と実行

講義時間中に示した再帰的に Euler 閉路を列挙するクラス `Euler` を参考に、Algorithm 2 によって非再帰的に Euler 閉路を列挙するクラス `EulerNonRecuresive` を作成しなさい。コンストラクタでは、対象となるグラフと始点を引数としなさい。また、探索をするメソッドは `public List<List<Arc>> startEnumerate()` とする。

Algorithm 2 に現れる始点からの経路と到達した頂点の組 (P, v) に相当するクラスを定義しなさい。このクラスは、情報を保持するのが目的であることから、コンストラクタ以外のメソッドを持たなくてもよい。

実行にあたっては、再帰的アルゴリズムと結果を比較しなさい。また、本レポート課題中のグラフとは異なるグラフを 2 つ以上試しなさい。

3　期末レポートについて

期末レポート課題は、以下の通りです。期日 17 時までに、Teams より PDF にて提出すること。

- Word や LaTeX などを利用して電子的に組版すること。提出するファイル名は学籍番号.pdf とすること。
- レポート課題と対応するプログラム等を含むこと。
- 動作例を作成すること。
- 単に、プログラムだけ提出した場合には、不合格とする。
- プログラムでは、わかりやすい変数名とメソッド名を用いるとともに、適切にコメントを付し、可読性を高めること。
- プログラムの全体に関する説明をつけること。
- 正しい日本語で作成すること。
- 他人のレポートを写したと判断した場合には、不合格とする。
- 書籍や Web ページ等を参考としている場合には、必ず出典を明示すること。

4　レポート採点基準

C:プログラムを作成しているが、十分な説明が行われていない。または、適切にクラス構成が行われていない。
B:クラスが適切に設計され、プログラムに十分な工夫がある。
A:B に加え、クラス設計、処理の流れがレポートで適切に説明されている。
S:A の基準を満たし、特に顕著な工夫や記述がある。