アルゴリズムとデータ構造　第一回レポート課題

2019/05/03

US162039：梶田悠

2019

目次

[レポートの課題内容 2](#_Toc8496197)

[線形リスト 2](#_Toc8496198)

[処理手順 2](#_Toc8496199)

[コード 3](#_Toc8496200)

[実行結果 3](#_Toc8496201)

[考察 4](#_Toc8496202)

[木構造 4](#_Toc8496203)

[処理手順 4](#_Toc8496204)

[コード 4](#_Toc8496205)

[実行結果 4](#_Toc8496206)

[考察 5](#_Toc8496207)

# レポートの課題内容

１行が「数値,文字列」で構成されるファイル（data\_long.txt）を読み込み，ファイルの全てのデータを線形リストと木構造（２分探索木）に格納するプログラムを作成し、各データ構造の構築時間を比較せよ。 また、構築された各データ構造に対して「数値」で探索し、対応する「文字列」を表示する機能を実装し、「数値」で１０００回、探索する場合の探索時間を比較せよ。

コードは[github](https://github.com/HarukaKajita/DataStructure_report01)にアップロードしてあるので、レポートのコードが読みづらい場合は適宜参照して下さい。

# 線形リスト

### 処理手順

　データ構造構築の処理手順を以下に示す。

1. コマンドライン引数でデータが記述されたファイルを入力。
2. データを一行ずつ読み込み、セルにキーと値を格納。
3. 全データを読むまで２を繰り返す。この時一つ前のセルの参照を各セルが保持することで線形リストとしてのデータ構造を構築している。（末尾はNULL）

探索処理の手順を以下に示す。

forループで0~999までの整数をキーとして持つデータを探索するプログラムになっているが、以下はそのfor文内での処理の手順である。

1. リストの先頭要素とキーの値を引数に、渡されたキーを持つデータを探し該当するデータがあればその要素のポインタを返すsearch\_cell()関数を呼び出す。
2. １の返り値がNULLの場合、探索がリストの末尾の要素まで到達した事を示すので、該当データーが存在しない旨の文を出力。
3. １の返り値がNULLでなかった場合は、該当するデータが存在した事になるのでデータのキーと値を出力。

## コード

コード 1 list\_cell.cpp

|  |
| --- |
| // 線形探索（配列の動的確保）  #include <iostream>  #include <fstream>  #include <stdlib.h>  #include <string.h>  //時間計測用  #include <chrono>  using namespace std;  // セルを表わす構造体の定義  struct cell  {  int key; // キー  char data[256]; // キーに対応する値（文字列）  struct cell\* next; // 次のデータへのポインタ  };  // 機能：リストの要素生成  // 引数: int key キー  // char \*data 対応するデータとなる文字列  // 戻値: 生成された要素へのポインタ．  struct cell\* make\_cell(int key, const char\* data)  {  struct cell\* mk\_cell;  // cellの領域を確保する  mk\_cell = new struct cell;  /\* 文字列 data を cell->data へコピーする \*/  strcpy\_s(mk\_cell->data, data);  // 整数へ変換  mk\_cell->key = key;  // nextを初期化  mk\_cell->next = NULL;  return mk\_cell;  }  // 機能：リストを検索  // 引数：head -- リストの先頭要素のアドレス  // key -- 検索したいキー  // 戻値：キーが存在した要素のポインタ．ない場合は NULL  struct cell\* search\_cell(struct cell\* head, int key)  {  struct cell\* buf;  buf = head;  while (buf != NULL)  {  if (buf->key == key)  {  return buf;  }  buf = buf->next;  }  return NULL;  }  int main(int argc, char\* argv[])  {  if (argc == 1) // 引数がない場合  {  cout << "input file name\n";  return -1;  }  // ファイルオープン  ifstream fp(argv[1]);  // リストのヘッドを定義し初期化  struct cell\* head;  head = NULL;  // データのリストへの格納  cout << "Loading " << argv[1] << endl;  string line;  //構築時間の計測開始  chrono::system\_clock::time\_point start, end;  start = chrono::system\_clock::now();  while (fp >> line)  {  // ファイルからキーと対応する値（ファイル名）を取得  int index = line.find(","); // 区切り文字までの文字数  string key\_str = line.substr(0, index);  string value = line.substr(index + 1);  // キーを整数変換 （本当は危険）  int key = atoi(key\_str.data());  if (value.length() > 255)  {  cout << "length of value is too long\n";  return -1;  }  // リスト要素の生成  struct cell\* e\_cell;  e\_cell = make\_cell(key, value.data());  // 生成した要素をリストの先頭に追加  e\_cell->next = head;  // 先頭アドレスを更新  head = e\_cell;  }  //構築時間の計測終了  end = chrono::system\_clock::now();  double elapse = chrono::duration\_cast<chrono::microseconds>(end - start).count();  cout << "構築時間：" << elapse << " (micro sec)" << endl;  //0~1000の整数をkeyにして1000回検索し、探索時間を計測  start = chrono::system\_clock::now();  for (int i = 0; i < 1000; i++)  {  int ky = i;  // 配列 data から値がkyである要素番号を探索  struct cell\* s\_cell;  s\_cell = search\_cell(head, ky);  if (s\_cell == NULL)  {  cout << ky << "に対応する値は存在していません。" << endl;  }  else  {  cout << s\_cell->key << " -> " << s\_cell->data << endl;  }  }  //探索時間の計測終了  end = chrono::system\_clock::now();  elapse = chrono::duration\_cast<chrono::microseconds>(end - start).count();  cout << "探索時間：" << elapse << " (micro sec)" << endl;  fp.close();  return 0;  } |

## 実行結果

実行結果 　list\_cell.cppの実行結果の抜粋

|  |
| --- |
| Loading data\_long.txt  構築時間：782510 (micro sec)  //--中略--  探索時間：2.16433e+07 (micro sec) |

# 木構造

## 処理手順

データ構造構築の処理手順を以下に示す。

1. ああああ
2. ああああ

探索処理の手順を以下に示す。

forループで0~999までの整数をキーとして持つデータを探索するプログラムになっているが、以下はそのfor文内での処理の手順である。

1. ままままま
2. みみみいみｍ

## コード

コード 2　mk\_tree.cpp

|  |
| --- |
| class Storage {  private:  Food foods[300]; |

## 実行結果

実行結果 　mk\_tree.cppの実行結果の抜粋

|  |
| --- |
| Loading data\_long.txt  構築時間：2.09494e+06 (micro sec)  //--中略--  探索時間：463457 (micro sec) |

# 考察