平成31年度　知能情報工学実験演習2

離散構造モデリング

第1回演習資料

坂本比呂志　平田耕一　芳野拓也

情報工学部　知能情報工学科

1. この実験演習の目的と進め方

実験の準備

1. 実験の資料を下記のURLの講義資料からダウンロードする

   http://www.donald.ai.kyutech.ac.jp/~hiroshi

   または 「坂本比呂志」で検索して講義資料から入手する

2. ダウンロードしたzipファイルを解凍する(“unzip filename.zip”)

3. プログラムの実行許可(“chmod u+x ex.sh”)

3. ex.shを実行する（./ex.sh リターン）

4. 下記アドレスに出席確認メールを送る（次ページを参照）
   提出用アドレス： experiment@donald.ai.kyutech.ac.jp

【メール送信時の件名の約束】

プログラムで自動チェックを行っているのでメールの件名は下記の書式とすること
メールの送信先： experiment@donald.ai.kyutech.ac.jp

- 出席確認メール → attend○_△
- 課題提出メール → kadai□_△
- 質問メール → question_△

ただし、○=実験の回（1~3）、△=学籍番号、□=課題番号(1~6)

【正しい件名の例】
attend1_13231017
kadai4_13231017
kadai2_13231017再提出　　※末尾に追加した文字列は無視されるのでOK
question_13231017（2）　　※これも同様にOK

【誤りの例】
attend_13231017　　※実験の回の書き忘れ
attend_99999999　　※存在しない学籍番号
kadai_13231017　　※課題番号の書き忘れ
queston_13231017　　※単語の綴り間違い

評価について

合計3回（3週）の演習

1回あたり2つの課題に取り組む(課題1~6まで)

- 奇数番目の課題は基本課題で必須（1,3,5の提出が合格の最低ライン）
- 偶数番目の課題は発展課題でオプション（出来具合により加点する）

課題の提出について

- プログラム，実行結果をまとめて電子ファイルをメールで提出

  (詳細は各課題の指示に従う)

- 奇数番目の課題は原則として時間内に提出すること
- 偶数番目の課題は3週目の終わりまでに提出すること
- 課題提出が遅れそうなときはTAに相談すること

課題提出について(補足)

課題の提出について

・プログラム，実行結果をまとめて電子ファイルをメールで提出

1.課題提出後に受理した旨の返信メールがTAから届く(数日中)

2.返信メールが来ない場合，課題は受理していないのでTAに至急連絡

3.以上の手続きを怠った場合，追加レポートの受理は認めない

上の連絡を含め個別にTAから連絡が来る可能性があるので，

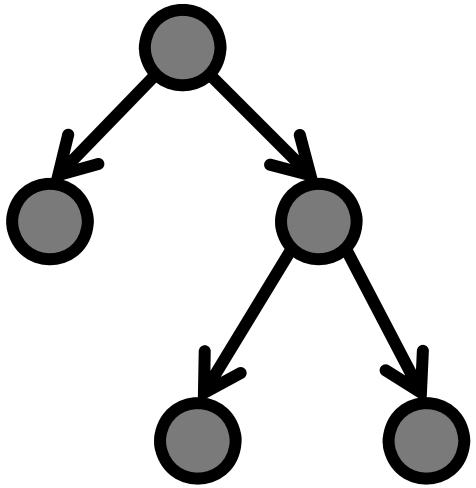
自分が課題提出に使ったメールアドレスは実験期間中，こまめに確認

※他人のレポートをコピーすることは禁止する．判明した場合は不可とする．

この演習のねらい:
離散構造データに習熟する

離散構造データのいろいろ

- 1次元データ:文字列(整数の列も文字列と同義)
  - DNA配列:ACGTACCAGTATA...
  - Twitterのつぶやき
  - HTML/XMLデータ
- 2次元データ:木構造(平面に埋め込み可能)
- 3次元データ:グラフ構造



この演習では1次元データ(文字列)に関する問題に取り組み,入出力の管理,メモリの動的確保,ポインタについての理解を深める

プログラミング言語はC言語を用いる

この演習で取り組む課題:
データ圧縮プログラムの作成

可逆データ圧縮:

データxについて，f(x)=y, size(x) > size(y) となるような関数f()を計算するプログラム

run-length圧縮:

同一文字の連続(run)をその長さ(length)で置き換える圧縮法

文字コードの読み替えによって，runは
次のように置き換え可能

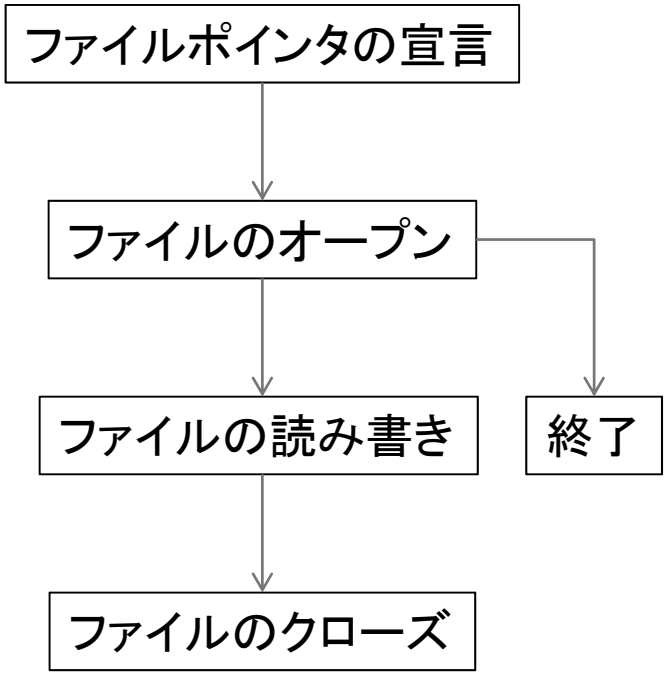
AAAAAAAAA = A9
AAAAAAAAABBBBB = A9B5
AAAAAAAAABBBBBCCCCCC = A9B5C6

汎用的なrun-length圧縮: 長いrunがないデータでもうまく圧縮したい

2. 基本のおさらい

ファイルの入出力

ファイルの読み込みの流れ



ファイルから一行ずつ読み込んでそれを配列に格納して標準出力に表示するプログラムのサンプル

```
#define MAXSTR 100000000 // 文字列の最大長さ
#define MAXLEN 100000 //ファイルに書き込まれた1行の最大長
//func.hに記載

//領域の確保
string = (char *)malloc(sizeof(char)*MAXSTR);
tmp_string = (char *)malloc(sizeof(char)*MAXLEN);

// fgetsで入力ファイルの文字列を一行ずつ読み込み配列stringに格納
while (fgets(tmp_string, MAXLEN, file_in)) {
strcat(string,tmp_string);
}
```

コマンドライン引数

C言語では以下のようにコマンドラインから引数を渡せる

int main(int argc, char *argv[ ])

・argcは引数の個数(プログラム自身を含む)

・argv[ ]は引数の文字列を格納した配列へのポインタを表す

実行ファイルa.outを作成し，./a.out file1 file2 file3などと実行した場合は以下のように引数に関する情報が格納される

argc
4

argv[0]	argv[1]	argv[2]	argv[3]
a.out¥0	file1¥0	file2¥0	file3¥0

入出力のサンプルプログラム (sample.c)

入力ファイルから一行ずつ読み込んで出力ファイルに書きだすプログラムのサンプル（引数のエラー処理を含む）

```
char *string_array;
int size;    //配列のサイズ
FILE *file_in, *file_out;

  // 引数のチェックと入出力ファイルのオープン
  if (argc != 3){
    printf("different number of arguments¥n");
    exit(1);
  }

  if ((file_in=fopen (argv[1],"r")) == NULL ){
    printf("input file not opened¥n");
    exit(1);
  }
// 右のプログラムに続く
```

```
if ((file_out=fopen (argv[2],"w")) == NULL){
  printf("output file not opened¥n");
  exit(1);
}

// fgetsで入力から一行ずつ読み込みながら
// file_outに書き込む
while (fgets(string_array, MAXLEN, file_in)) {
  fputs(string_array, file_out);
}

free(string_array);
fclose(file_in);
fclose(file_out);
```

プログラムの分割コンパイルとmake

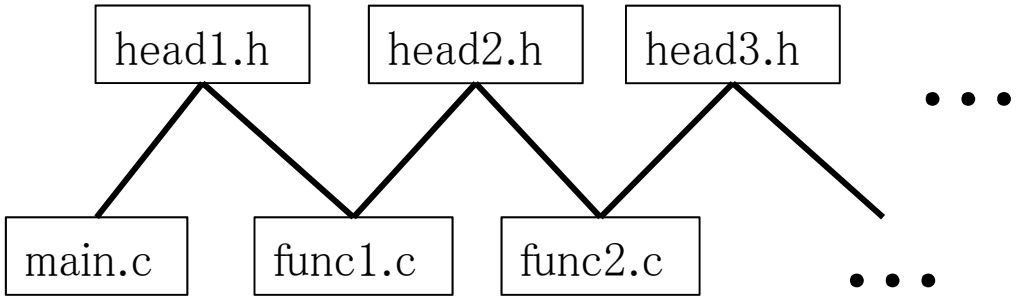
(Cに限らず)プログラムはソースコードを別々に分割して開発することで効率がよくなる

Cの場合は，以下のように分割して作成し，まとめてコンパイルする

(1) main**関数**

(2) その他の関数

(3) 関数のプロトタイプ宣言や構造体定義(ヘッダーファイル)




```
//func.h
#define MAXSTR 100000000 // 文字列の最大長
#define MAXLEN 100000 //ファイルに書き込まれ
int runlength(unsigned char *string, int size);
int main(int argc, char *argv[]);
```

gcc -c main.c → main.o, …
gcc -o main main.o func1.o func2.o …

毎回これをコマンドラインで実行するのは大変

makeによるバッチ処理

・makeコマンドはMakefileという名前のファイルで定義されているバッチ処理を実行する

・Makefileのバッチ処理は以下の命令を並べたものからなる

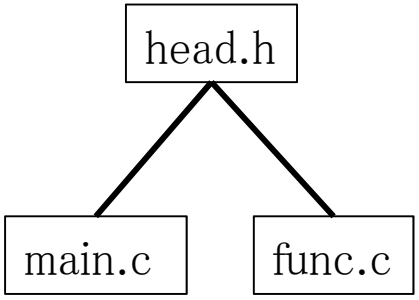
1行目　目的のファイル：それを生成するためのソースファイル

2行目　タブ　生成方法の記述

・図のソースファイルでは以下のようにMakefileを記述する

```
all: main

main: main.o func.o
        gcc -o main main.o func.o
main.o: main.c head.h
        gcc -c main.c
func.o: func.c head.h
        gcc -c func
```



図

3. 基本的なrun-length圧縮

ランレングス(run-length)圧縮

符号化

・同じ文字の連続をラン(RUN)という（赤い部分は長さ6のRUN）

例：bbbbbbbbabbbabaaaaaa

・ランレングス(Run-Length)圧縮：文字列のランをその長さで表現する圧縮法

・例の文字aの長さ6のランはa6と置き換える

復号化

・例えば　a3b1c2a3　が与えられたとき，最初の文字をそのランの文字，次の文字をそのランの長さを表す整数とみなして復号する

・補足：1 Byte 文字にはそれぞれ7 bitで表現される整数が割り振られており，これをASCII コートと呼ぶ（次の表を参照）

・環境によっては先頭に0を詰めて8bitで表現することが多い．（本実験でも8bitと考える）

ASCII コード表

10進数	2進数	割り当て文字
0	00000000	NUL
…	…	…
65	01000001	A
66	01000010	B
67	01000011	C
…	…	…
97	01100001	a
98	01100010	b
…	…	…
127	01111111	DEL

ランレングス圧縮プログラムの概要

・入力文字列Sが与えられる(Sは文字列)

・Sを先頭から一文字ずつ読みながら以下の処理を行う

（実際には1行ずつ読み込みながら）

その文字が例えば'A'のとき

AのRUNの長さを数えて，それをn(A)とする

整数n(A)に対応するASCII文字をXとする(例えばn(A)=67ならばX='C')

処理したRUNに対する符号として2文字AXを出力する

・以上の処理をSを読み終わるまで続ける

[例] AA…AのようにAが67回連続していればこの部分を2文字ACで符号化する．
（ASCIIコード表により文字Cが表す整数は67であり，ACでAの67回の連続を表現）

[注意] この符号化では，偶数番目(0番から始まる)の文字がその文字自身であり，
続く奇数番目の文字がその文字の連続回数を表す．（繰り返しが1の場合も含む）

課題1

DNA配列テキスト(Test1.txt)を入力として読み込みながらランレングス圧縮を行い結果をファイルに書きだすプログラムを作成せよ.

圧縮文字列を復号プログラムによって圧縮前のテキストに正しく復号できることを確認すること. デコーダは”make decode”とコマンド入力すると作成される.

また, 圧縮結果をオリジナルのテキストと比較してその圧縮率(圧縮後テキストのバイト数 / 元テキストのバイト数)を計算し, 十分に小さくなっていることを確認せよ.

※課題全体(1～6)についての注意事項

main.c, func.h および Text1~6.txt は変更してはならない.

課題1の提出方法

提出するもの

- Makefile
- func.h
- main.c
- check.c
- runlength.c
- Test1.txt（オリジナルデータ）
- Test1.rle（Test1.txtを圧縮したファイル）

これらをフォルダ“kadai1_学籍番号”にまとめて、ZIPで圧縮

(フォルダの圧縮コマンド)

zip -r kadai1_学籍番号.zip kadai1_学籍番号

(アドレス) experiment@donald.ai.kyutech.ac.jp

(件名) kadai1_学籍番号

課題2

課題1で作成したプログラムは，runの長さが十分長くなると（256以上）圧縮データを復号できなくなる．この問題を解決しTest2.txtを圧縮せよ．

※課題2を提出すれば課題1を提出する必要はないものとする．

課題2の提出方法

提出するもの

- Makefile
- func.h
- main.c
- check.c
- runlength.c
- Test2.txt（オリジナルデータ）
- Test2.rle（Test2.txtを圧縮したファイル）

これらをフォルダ“kadai2_学籍番号”にまとめて、ZIPで圧縮

(アドレス) experiment@donald.ai.kyutech.ac.jp

(件名) kadai2_学籍番号