知能情報処理

第13回～15回
ゼミ配属の問題への適用
(GAはいろいろな問題に適用できる）

ゼミ配属問題とは

- 100人の学生から，10の研究室の希望調査を行った．どのように振り分ければ，もっとも学生の希望を満たしつつ，配属人数のバランスがとれた配属ができるか．
- 希望調査の結果は、seminar.csvに

学籍番号, 第1希望, 第2希望, 第3希望

1,1,2,3

2,1,2,3

3,1,2,4

4,1,2,8

5,1,3,2

のように格納されている。ゼミの番号は1～10で書かれている(ここでは0～9ではないので注意）。

コード化

学生番号	0	1	2	3	4	5	6	…	99
研究室番号	7	5	1	2	1	6	0	…	2

研究室番号：0～9の整数（1～10ではないので注意。1足した値が実際の研究室番号。）

クラスの数 NCLASS (=10), 学生数 LENGTH (=100)

学生番号 gno[0]～gno[LENGTH -1]
第1希望 kibo1[0]～kibo1[LENGTH -1]
第2希望 kibo2[0]～kibo2[LENGTH -1]
第3希望 kibo3[0]～kibo3[LENGTH -1]
以上をファイルから読み込む

個体群 population[NUM][LENGTH]

適応度 fitness[NUM]

	学生0	学生1	学生2	学生3	・・・	学生 LENGTH-1
個体0	(ゼミ)4	3	0	9	・・・	6
個体1	8	5	2	0	・・・	7
:	:					
個体 NUM-1	7	5	0	3	・・・	6

適応度
530
471
686

Kibo1[ ]	7	4	0	3	・・・	6
Kibo2[ ]	6	7	1	5	・・・	9
Kibo3[ ]	1	3	4	8	・・・	0

人数

	クラス0	クラス1	クラス2	・・・	クラス NCLASS-1
個体0	4	8	2	・・・	3
個体1	12	7	1	・・・	2
個体NUM-1	14	10	0	・・・	1

各個体毎・クラス毎の人数を求める関数. ninzuはint ninzu[NUM][NCLASS]の配列
呼ぶときは

calc_ninzu(population, ninzu);

適応度

- 制約条件の充足により
- **希望の実現**

  一人ずつ、第1希望（10），第2希望（5），第3希望（3），それ以外（0）
- **ハード制約**：

  １．一人は1研究室のみ配属可能

  　　このコード化によれば，自動的に満たされる

  ２．１研究室に15名を超えることは許されない

  　　大きなペナルティ（−100等）
- **ソフト制約**：

  研究室の学生数の上限：10名を超えた部分については，一人超えるたびにペナルティ（=−5としてみる）。

```
int ninzu_penalty1=100;
int ninzu_penalty2=5;
calc_score(ninzu_penalty1,ninzu_penalty2,population,ninzu,gno,kibo1,kibo2,kibo3, fitness);
```

13～15回の問題

- 適応度がなるべく高いゼミ配属表を作成せよ。
- （レポートの内容）
  - プログラムリスト（全体を提出すること）
  - 繰り返し100回目の結果において、適応度が最大の配属に関する
    - 決まった配属先が第1希望の数
    - 決まった配属先が第2希望の数
    - 決まった配属先が第3希望の数
    - 適応度の値
    - 全員の配属先の表
    - コメント、感想

  をレポートにして提出せよ。
- 提出期限
  - 1月5日（火）13:50　その後解説します

課題(第13回）

- 次の部分を作って、14:30までにメール送付せよ。プログラムリストにその場所を示す。ファイル名は、dai13kaime.txtとせよ。
- 個体population[NUM][LENGTH]と、ファイルから読み込んだ
  - 学生番号 gno[LENGTH]
  - 第1希望 kibo1[LENGTH]
  - 第2希望 kibo2[LENGTH]
  - 第3希望 kibo3[LENGTH]

があるとき、次ページに示す部分のプログラムを完成させよ。

- 値 population[best_id][j] に対して、希望順位（1～3, それ以外）
  juni[j]=1,2,3,0となるようにする
- 同じく個体best_id に対して、希望順位1～3が実現出来ている人数
  juni_count[0], juni_count[1],…,juni_count[3]
- 同じく個体best_id に対して、研究室ごとの人数
  zemi_count[0]～zemi_count[9]

12/15はこれを完成させる

```
// 学籍番号、氏名をここに書くこと
//順位の計算

（作ること）

//配属を書き出す
fprintf(fp,"番号,第1希望,第2希望,第3希望,配属,希望順位¥n");
for (i=0;i<LENGTH;i++){
    fprintf(fp,"%d,%d,%d,%d,%d,%d¥n",gno[i],kibo1[i],kibo2[i],kibo3[i],population[best_id][i],juni[i]);
}
//第1希望から第3希望が実現している数を数える
juni_count[0] = 0;
juni_count[1] = 0;
juni_count[2] = 0;
juni_count[3] = 0;

（作ること）

for (i=1;i<=3;i++){
    fprintf(fp,"第%d希望,%d¥n",i,juni_count[i]);
}
fprintf(fp,"それ以外,%d¥n",juni_count[0]);
//研究室ごとのカウント数

（作ること）

fprintf(fp,"研究室,カウント数¥n");
for (i=0; i<NCLASS; i++){
    fprintf(fp,"%d,%d¥n",i,zemi_count[i]);
}
```

- プログラムは、できあがっているものを田中が持っているので、それに入れてコンパイルし、エラーがあった場合はエラーメッセージをメールで返す。
- メール送付先 m_tanaka@center.konan-u.ac.jp （centerのついていないアドレスを使わないこと）
- プログラムのその他の部分もこの科目のHPに置いているが、次回の課題に関係する部分を抜いているので、自分でこの不完全なサンプルをコンパイルすることはできない。
- 学生は、不完全なサンプルにおける、隠してある部分を全部作らなければ、コンパイルは通らない。