1　アルゴリズム演習 6

1.1　二分探索

二分探索は、データがソートされて小さい順 (または大きい順) に並んでいるときに有効な探索方です。

次の様なデータからデータの 90 を二分探索する場合を考えてみましょう。

```
配列 a[]
添え字　: 値
 0      :  5 <-low
 1      :  8
 2      : 15
 3      : 21
 4      : 33
 5      : 36
 6      : 41 <-x
 7      : 55
 8      : 76
 9      : 79
10      : 80
11      : 85
12      : 90 <-upper
```

検索範囲の下限を low、上限を $upper$ とし、$x = (low + upper)/2$ の位置のデータとキー(探すデータを比較します。もし、キーの方が大きければ、キーの位置は x より上にあるはずなので low を $x + 1$ にします。逆に、キーの方が小さければ、キーの位置は x より下にあるはずなので $upper$ を $x - 1$ にします。これを、$low <= upper$ の間くり返します。

「C 言語」(河西朝雄著　ナツメ社)154 頁

1.2　2bun.c

```
/*
  2bun.c
  2 分探索
  C 言語 155 頁
```

```
*/

#include <stdio.h>

#define N 13

int main()
{
  int a[] = {5, 8, 15, 21, 33, 36, 41, 55, 76, 79, 80, 85, 90};
  int x;
  int key;
  int low = 0;
  int upper = N - 1;

  printf("data ? ");
  scanf("%d", &key);

  while(low <= upper){
    x = (low + upper) / 2;
    if(a[x] == key){
      printf("%d番 : %d\n", x + 1, a[x]);
      break;
    }
    if(a[x] < key){
      low = x + 1;
    }else{
      upper = x - 1;
    }
  }
  return 0;
}
```

1.3 periodic.c

```
/*
 periodic.c
*/
#include <stdio.h>
#include <stdlib.h>
```

```
#define N 20

typedef struct{
  int  num;
  char name[20];
}gen;

int binary_search(gen a[],int key){
  int left=0,right=N-1,center;
  for(;;){
    center=(left+right)/2;
    if(a[center].num==key) return center;
    if(left>=right) return -1;
    if(key > a[center].num){
      left = center + 1;
    }else{
      right = center - 1;
    }
  }
  return -1;
}

int main(void){
  int  gennum,i=0,key;
  char buf[256];
  FILE *fp;
  gen genso[N];
  fp=fopen("genso.txt","r");
  if(fp==NULL){
    puts("File_error.");
    exit(1);
  }
  while((fgets(buf,256,fp))!=NULL){
    sscanf(buf,"%d %s",&genso[i].num,&genso[i].name);
    printf("%s\n", buf);
    i++;
  }
  puts("１～２０番まで元素番号の元素を検索できます。");
  printf("番号を入力してください："); scanf("%d",&key);
  gennum=binary_search(genso,key);
```

```
  if(gennum==-1){
    puts("検索対象はありません。");
    return;
  }
  printf("%d番は%sです。\n",genso[gennum].num,genso[gennum].name);
  fclose(fp);
  return 0;
}
```

1.4 genso.txt

```
1 水素
2 ヘリウム
3 リチウム
4 ベリリウム
5 ホウ素
6 炭素
7 窒素
8 酸素
9 フッ素
10 ネオン
11 ナトリウム
12 マグネシウム
13 アルミニウム
14 ケイ素
15 リン
16 硫黄
17 塩素
18 アルゴン
19 カリウム
20 カルシウム
```

2 AIによる大規模データ処理入門

人工知能の分野では、生物の進化にヒントを得た計算手法が研究されています。こうした計算手法は、進化的計算あるいは進化的手法と呼ばれています。その代表例は、ホランド (J.H.Holland) が 1970 年代に提唱した遺伝的アルゴリズム (genetic algorithm. GA)

です。遺伝的アルゴリズムはさまざまなバリエーションがあります。
AI による大規模データ処理入門　165 頁　小高　知宏著　オーム社

2.1 遺伝的アルゴリズム ga.c

```
/************************************/
/*        遺伝的アルゴリズム          */
/*            ga.c                  */
/************************************/

#include <stdio.h>
#include <stdlib.h>

/*    記号定数                      */
#define LIMITL 256 /*配列最大領域*/
#define BUFSIZE 256 /*入力バッファサイズ*/
#define DIM 5 /*入力数*/
#define LINES 32 /*遺伝子座の数*/
#define POOLSIZE 30 /*プールサイズ*/
#define LOOPLIMIT 50 /*繰り返し世代数*/
#define SEED 32767 /*乱数のシード*/
#define MRATE 0.03 /*突然変異率*/

/*    関数のプロトタイプの宣言      */
int initldata(int d[][DIM+1]);
            /*学習用データの初期化*/
int binrand() ;/* 2 値乱数 */
double frand(double fmax) ;/*実数乱数*/
void initpool(int pool[][LINES]) ;
            /*遺伝子プールの初期化*/
void evalpool(int pool[][LINES],
      double poolval[],int ldata[][DIM+1],
      int no_of_ldata) ;/*プールの評価値の計算*/
double eval(int gene[],
         int ldata[][DIM+1],int no_of_ldata) ;
                     /*遺伝子の評価値計算   */
void printpool(int p[][LINES],double pv[]) ;
                     /*  プールの状態表示   */
void mating(int p[][LINES],double pv[],
                           int dp[][LINES])  ;
```

```
                        /*  交叉 */
void mutation(int dp[][LINES]);/*突然変異*/
int reverse(int val) ; /*0/1 の逆転*/
void evaldpool(int dpool[][LINES],
      double dpoolval[],int ldata[][DIM+1],
      int no_of_ldata) ;/*dpool の評価値の計算*/
void selection(int p[][LINES],double pv[],
       int dp[][LINES],double dpv[]) ;/*選択*/
double average(double pv[],int n) ;
                        /*   平均値の計算     */
int nrand(int n) ; /* n 未満の整数乱数の生成*/
void singlemating(int p[][LINES],
  double pv[],int dp[][LINES],int point,
  double sumofpoolval) ;/*  交叉を 1 回実施 */
void mating(int p[][LINES],double pv[],
        int dp[][LINES]) ; /*交叉と増殖*/
int roulette(double pv[],double sumofpoolval) ;
                    /*  ルーレット選択     */


/**********************/
/*     main() 関数     */
/**********************/
int main()
{
 int ldata[LIMITL][DIM+1] ;/*学習用データ*/
 int no_of_ldata ;/*学習用データの個数*/
 int pool[POOLSIZE][LINES] ;/*遺伝子プール*/
 double poolval[POOLSIZE] ;/*遺伝子の評価値*/
 int dpool[POOLSIZE*2][LINES] ;/*増殖中のプール*/
 double dpoolval[POOLSIZE*2] ;/*増殖中の評価値*/
 int generation ;/*世代数*/

 /*乱数の初期化*/
 srand(SEED) ;

 /*学習用データの初期化*/
 no_of_ldata=initldata(ldata) ;

 /*遺伝子プールの初期化*/
```

```
 initpool(pool) ;
 evalpool(pool,poolval,
          ldata,no_of_ldata) ;/*評価値の計算*/
 printpool(pool,poolval) ;/*プールの状態表示*/
 printf("\n") ;

 /*遺伝的アルゴリズムの繰り返し*/
 for(generation=0;generation<LOOPLIMIT;++generation)
 {
   /*交叉と増殖*/
   mating(pool,poolval,dpool) ;

   /*突然変異*/
   mutation(dpool) ;

   /*選択*/
   evaldpool(dpool,dpoolval,ldata,no_of_ldata) ;
   selection(pool,poolval,dpool,dpoolval) ;

  /*処理後のプールの状態表示*/
   printf("%d  %lf",generation+1,poolval[0]) ;
   printf(" %lf\n",average(poolval,POOLSIZE)) ;
   printpool(pool,poolval) ;/*プールの状態表示*/
   printf("\n") ;
 }

 return 0 ;
}

/**********************/
/*  average() 関数     */
/*   平均値の計算      */
/**********************/
double average(double pv[],int n)
{
 int i ;
 double sum=0 ;

 /*pv[] の平均値を計算*/
 for(i=0;i<n;++i)
```

```
  sum+=pv[i] ;

 return sum/n ;
}

/**********************/
/*  selection() 関数   */
/*      選択           */
/**********************/
void selection(int p[][LINES],double pv[],
  int dp[][LINES],double dpv[])
{
 int i,j,k ;
 double maxval ;
 int maxpos ;
 /*p[][] へ dp[][] の上位を代入*/
 for(i=0;i<POOLSIZE;++i){
  maxval=dpv[0] ;
  maxpos=0 ;
  for(j=0;j<POOLSIZE*2;++j){
   if(dpv[j]>maxval){
    maxval=dpv[j] ;
    maxpos=j ;
   }
  }
  /*評価最高の遺伝子を p へ移動*/
  for(k=0;k<LINES;++k) p[i][k]=dp[maxpos][k] ;
  pv[i]=maxval ;
  dpv[maxpos]=0 ;/*最低値に書き換え*/
 }
}

/**********************/
/*  mutation() 関数    */
/*    突然変異         */
/**********************/
void mutation(int dp[][LINES])
{
 int i,j ;
```

```
 for(i=0;i<POOLSIZE*2;++i)
  for(j=0;j<LINES;++j)
   if(frand(1.0)<MRATE)
     dp[i][j]=reverse(dp[i][j]) ;
}

/**********************/
/*   reverse() 関数    */
/*     0/1 の逆転      */
/**********************/
int reverse(int val)
{
 if(val==0) return 1 ;
 return 0 ;
}

/**********************/
/*  roulette() 関数    */
/*  ルーレット選択     */
/**********************/
int roulette(double pv[],double sumofpoolval)
{
 double sum=0 ;/*評価値の部分和*/
 int i=0 ;/*遺伝子の番号*/
 double limit ;/*ルーレットの当たりの値*/

 /*ルーレットの当たりの値を設定*/
 limit=frand(sumofpoolval) ;
 /*ルーレットを回す*/
 while(sum<limit)
  sum+=pv[i++] ;
 --i ;/*行き過ぎているので１引く*/

 return i ;
}

/**********************/
/*singlemating() 関数  */
/*  交叉を 1 回実施     */
/**********************/
```

```
void singlemating(int p[][LINES],
  double pv[],int dp[][LINES],int point,
  double sumofpoolval)
{
 int i ;
 int a,b ;/*ルーレットで選ぶ親の番号*/
 int crosspoint ;/*交叉点*/

 /*ルーレットを用いて親を選ぶ*/
 a=roulette(pv,sumofpoolval) ;
 while((b=roulette(pv,sumofpoolval))==a) ;

 /*交叉点を決定*/
 crosspoint=nrand(LINES) ;

 /*交叉により子遺伝子を作成*/
 for(i=0;i<LINES;++i){
  if(i<crosspoint){
   dp[2*point][i]=p[a][i] ;
   dp[2*point+1][i]=p[b][i] ;
  }
  else{
   dp[2*point][i]=p[b][i] ;
   dp[2*point+1][i]=p[a][i] ;
  }
 }
}

/**********************/
/*  mating() 関数      */
/*    交叉と増殖       */
/**********************/
void mating(int p[][LINES],double pv[],
            int dp[][LINES])
{
 int i ;
 double sumofpoolval=0 ;/*評価値の総和*/

 /*評価値の総和の計算*/
 for(i=0;i<POOLSIZE;++i)
```

```
  sumofpoolval+=pv[i] ;

 for(i=0;i<POOLSIZE;++i){
  /*親を選んで交叉を 1 回実行*/
  singlemating(p,pv,dp,i,sumofpoolval) ;
 }
}

/**********************/
/*  printpool() 関数   */
/*  プールの状態表示   */
/**********************/
void printpool(int p[][LINES],double pv[])
{
 int i,j ;

 for(i=0;i<POOLSIZE;++i){
  for(j=0;j<LINES;++j)
   printf("%d ",p[i][j]) ;
  printf(" : %lf\n",pv[i]) ;
 }
}

/**********************/
/*  eval() 関数        */
/*遺伝子の評価値計算   */
/**********************/
double eval(int gene[],
         int ldata[][DIM+1],int no_of_ldata)
{
 int i,j ;
 int pos ;/*遺伝子座の位置*/
 int n ;/*２のべき乗*/
 int hit=0 ;/*正解数*/

 for(i=0;i<no_of_ldata;++i){
  /*遺伝子座の決定*/
  pos=0 ; n=1 ;
  for(j=DIM-1;j>=0;--j){
   pos+=n*ldata[i][j] ;
```

```
   n*=2 ;
  }
  /*学習データとの比較*/
  if(ldata[i][DIM]==gene[pos])  ++hit ;
 }

 return (double)hit/no_of_ldata ;
}

/**********************/
/*  evaldpool() 関数   */
/* dpool の評価値の計算*/
/**********************/
void evaldpool(int dpool[][LINES],
      double dpoolval[],int ldata[][DIM+1],
      int no_of_ldata)
{
 int i ;

 for(i=0;i<POOLSIZE*2;++i){
  /*各遺伝子について評価値を計算*/
  dpoolval[i]=eval(dpool[i],ldata,no_of_ldata) ;
 }
}

/**********************/
/*  evalpool() 関数    */
/*プールの評価値の計算*/
/**********************/
void evalpool(int pool[][LINES],
      double poolval[],int ldata[][DIM+1],
      int no_of_ldata)
{
 int i ;

 for(i=0;i<POOLSIZE;++i){
  /*各遺伝子について評価値を計算*/
  poolval[i]=eval(pool[i],ldata,no_of_ldata) ;
 }
}
```

```
/**********************/
/*  initpool() 関数    */
/*遺伝子プールの初期化*/
/**********************/
void initpool(int pool[][LINES])
{
 int i,j ;

 for(i=0;i<POOLSIZE;++i)
  for(j=0;j<LINES;++j)
   pool[i][j]=binrand() ;
}

/**********************/
/*   frand() 関数      */
/*     実数乱数        */
/**********************/
double frand(double fmax)
{
 return (double)rand()/RAND_MAX*fmax ;
}

/**********************/
/*   nrand() 関数      */
/*n 未満の整数乱数生成 */
/**********************/
int nrand(int n)
{
 unsigned int r ;

 while((r=rand())==RAND_MAX) ;
 return (double)r/RAND_MAX*n ;
}

/**********************/
/*  binrand() 関数     */
/*     2 値乱数        */
/**********************/
int binrand()
```

```
{
 if((double)rand()/RAND_MAX>0.5)
  return  1 ;
 else
  return 0 ;
}

/**********************/
/*  initldata() 関数   */
/*学習用データの初期化*/
/**********************/
int initldata(int d[][DIM+1])
{
 char linebuf[BUFSIZE] ;/*入力バッファ*/
 int i=0 ;/*学習用データの個数*/

 while(fgets(linebuf,BUFSIZE,stdin)!=NULL){
                        /*行の読み取りが可能*/
  if(sscanf(linebuf,"%d %d %d %d %d %d",
            &d[i][0],&d[i][1],&d[i][2],
            &d[i][3],&d[i][4],&d[i][5])<=0)
    /*変換できなければ*/
    break ;/*行の読み飛ばし*/
  ++i ;
 }

 return i ;
}
```

2.2 and.dat

```
0 0 0 0 0 0
0 0 0 0 1 0
0 0 0 1 0 0
0 0 0 1 1 0
0 0 1 0 0 0
0 0 1 0 1 0
0 0 1 1 0 0
0 0 1 1 1 0
```

```
0 1 0 0 0 0
0 1 0 0 1 0
0 1 0 1 0 0
0 1 0 1 1 0
0 1 1 0 0 0
0 1 1 0 1 0
0 1 1 1 0 0
0 1 1 1 1 0
1 0 0 0 0 0
1 0 0 0 1 0
1 0 0 1 0 0
1 0 0 1 1 0
1 0 1 0 0 0
1 0 1 0 1 0
1 0 1 1 0 0
1 0 1 1 1 0
1 1 0 0 0 0
1 1 0 0 1 0
1 1 0 1 0 0
1 1 0 1 1 0
1 1 1 0 0 0
1 1 1 0 1 0
1 1 1 1 0 0
1 1 1 1 1 1
```

2.3 実行方法

```
./ga < and.dat
```