情報工学演習III

4. 簡単な整列アルゴリズムと線形リスト

渡辺宏太郎　伊達　央

補助　リュ・ビョンジュン　山下　美里
ブイ・クワン・ズン

マニュアルを持っているかどうかの確認

問53　各自の所持しているc言語のマニュアル（教科書）を提示せよ．
以後，演習IIIの時には携行する事．

確認印

単純交換ソート（バブルソート）

次に示す数字の並びを昇順に並べ替えることを考える.

3	4	5	2	6	1	7

（1ラウンド目）

3	4	5	2	6	1	7
3	4	5	2	1	6	7
3	4	5	1	2	6	7
3	4	1	5	2	6	7
3	1	4	5	2	6	7

1	3	4	5	2	6	7

比較するが交換しない

比較して交換する

1	3	4	5	2	6	7

液体中の気泡（液体より軽い）が上に上がってゆくイメージからバブルソートという名がついています.

（2ラウンド目）

1	3	4	5	2	6	7
1	3	4	5	2	6	7
1	3	4	2	5	6	7
1	3	2	4	5	6	7
1	2	3	4	5	6	7

バブルソートは隣り合う2つのデータで
左の方が大きければ，入れ替え，
そうでなければ入れ替えを行いません.

1ラウンドごとに左から小さい順に要素が確定しますから，ラウンドが進むにつれ，比較の回数は少なくなります.

（3ラウンド目）

1	2	3	4	5	6	7

つづく・・・

問54　以下のプログラムの関数bubble(int *a, int n)を完成し，前のページの例で正しく動作することを確認せよ．

```
#include <stdio.h>
#define NUMBER 7

void swap(int *a, int *b);
void bubble(int *a, int n);


int main(){

  int i;
  int a[NUMBER];

  for(i=0;i<NUMBER;i++){
    printf("a[%d]:",i);
    scanf("%d", &a[i]);
  }

  bubble(&a[0],NUMBER);

  for(i=0;i<NUMBER;i++){
    printf("%d ",a[i]);
  }
  printf("\n");
  return(0);
}

void swap(int *a, int *b){
  int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;
}

void bubble(int *a, int n){
  int i,j;


}
```

確認印

問55　バブルソートの例において3ラウンド目にはソートが完了している．よって3ラウンド目には1回も値の交換が行われない．各ラウンドの値の交換回数exchを記憶することにより，交換回数が0回ならば，以降のラウンドをスキップしてソート完了とするように関数bubbleを変更してみよ．この関数をbubble2とする．

以下のようにbubbleとbubble2の実行速度を比較してみよ．ただし，配列は次ページのほとんどソート済みの配列を用いる．

time ./a.out

で実行時間を計ることができる．

bubbleを用いた実行

```
 9856 9857 9857 9858 9859 9860 9861 9862 9863 9864 9865 9866 9867 9868 9869 9870
 9871 9872 9873 9874 9875 9876 9877 9878 9879 9880 9881 9882 9883 9884 9885 9886
 9887 9888 9889 9890 9891 9892 9893 9894 9894 9895 9896 9897 9898 9899 9904 9926


real    0m0.174s
user    0m0.163s
sys     0m0.000s
[wata@csimc13 ensyuu3]$
```

bubble2の方が速いことを確認せよ．

bubble2を用いた実行

```
 9840 9841 9842 9843 9844 9845 9846 9847 9848 9849 9850 9851 9852 9853 9854 9855
 9856 9857 9857 9858 9859 9860 9861 9862 9863 9864 9865 9866 9867 9868 9869 9870
 9871 9872 9873 9874 9875 9876 9877 9878 9879 9880 9881 9882 9883 9884 9885 9886
 9887 9888 9889 9890 9891 9892 9893 9894 9894 9895 9896 9897 9898 9899 9904 9926


real    0m0.018s
user    0m0.010s
sys     0m0.000s
[wata@csimc13 ensyuu3]$
```

確認印

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

#define NUMBER 10000

void swap(int *a, int *b);
void bubble(int *a, int n);
void bubble2(int *a, int n);


int main(){

  int i;
  int a[NUMBER];
  srand(11111);

  for(i=0;i<NUMBER-100;i++){
    a[i] = i;
  }
  for(i=NUMBER-100;i<NUMBER;i++){
    a[i] = rand()%10000; /*a[i]に乱数を代入 0から9999まで*/
  }

  bubble2(&a[0],NUMBER);

  for(i=0;i<NUMBER;i++){
    printf("%d ",a[i]);
  }
  printf("\n");
  return(0);
}

void swap(int *a, int *b){
  int tmp;
  tmp = *a;
  *a = *b;
  *b = tmp;
}
```

この配列はほとんどソート済みである．このような配列についてはbubble2は効果がある．

```c
void bubble2(int *a, int n){
  int i,j;
  int exch;

  for(j=1;j<=n-1;j++){
    for(i=n-1;i>=j;i--){

      /*適当な処理を書いてみよ*/
    }

    if(exch==0){
      break;
    }
  }

}
```

もう少し改良

今度はデータが次のように与えられているものとする．

1	2	7	3	5	6	4

1	2	7	3	5	4	6

1	2	7	3	4	5	6

1	2	7	3	4	5	6

1	2	3	7	4	5	6

1	2	3	7	4	5	6

1	2	3	7	4	5	6

この場合，最後の交換が行われるのは，先頭から2番目と3番目の要素になっている．

2ラウンド目以降，先頭から3番目までのまでの要素はソート済みとしてソート対象から外してよい．最後に交換した位置をlastとする．これを利用して，bubble2をさらに改良したbubble3を考えよう．

問56　最後に交換した位置をlastを利用して関数bubbleを変更してみよ．この関数をbubble3とする．

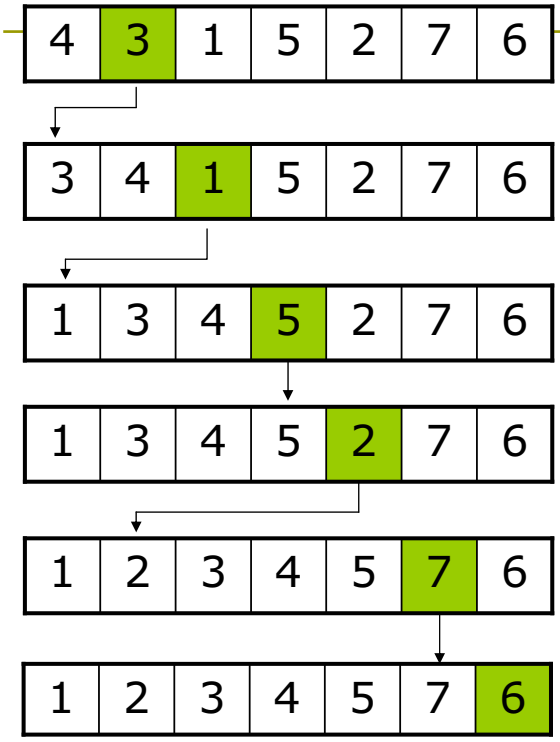
問55と同じデータを利用してbubbleとbubble3の実行速度を比較してみよ．

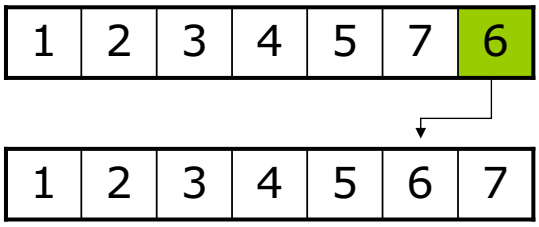
```
void bubble3(int *a, int n){
  int i,k = 1;
  int last;

  while(k < n-1){
    last = n-1;

    /*適当な処理を書いてみよ*/

    k = last;
  }

}
```

確認印

単純挿入ソート

次のような動作をする整列法を単純挿入ソートとよびます．





緑の部分に注目してその要素をそれより前（もしくは同じ位置）の位置に挿入します．

問57　単純挿入ソートを実行する関数void insertion(int *a, int n)を作成せよ．形式は下の形式を穴埋めしたものでも良いし，自分で1から作成したものでも良い．

（1）まず，4,3,1,5,2,7,6の整列が正しく行われることを確認せよ

（2）次に問55と同じデータを利用してbubbleとinsertionの実行速度を比較してみよ

```
void insertion(int *a, int n){
  int i,j,pos,tmp;

  for(i=1;i<n;i++){
    tmp = a[i];
    j = i-1;
    while(a[j]>a[i]&&j>=0){
      /* posとjをどうするのか */
    }

    if(j != /*適当に埋めよ */){
      for(j=i;;){ /*適当な処理を書く*/
        a[j] = a[j-1];
      }
      a[pos] = /*適当に埋めよ*/
    }

  }

}
```

この部分は，a[i]を挿入する位置を捜している処理

a[i]以前の配列の値が全てa[i]以下だったら何もする必要が無いから

最後にa[i]を挿入する

確認印

構造体リストの整列に関する問題

次のように4つのint型データをメンバにもつ構造体

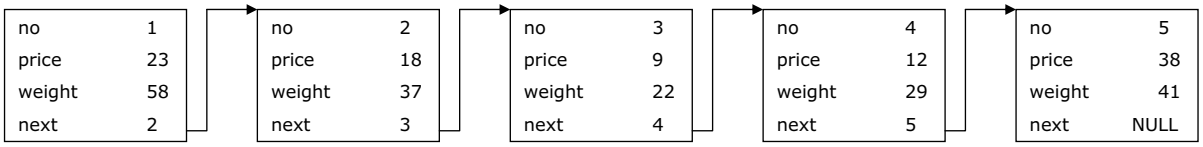
```
struct knap{
int no;
int  price;
int  weight;
struct knap *next;
}
```

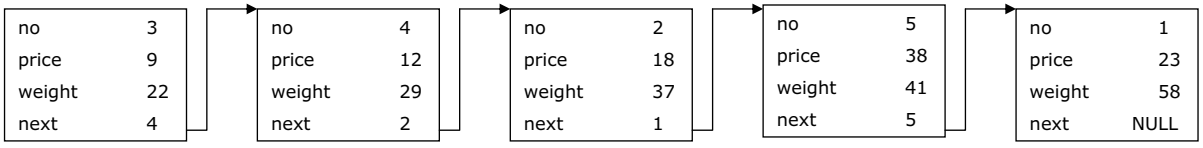
を考えます．意味としては，ある1つの商品の商品番，価値，重さを表します．この構造体のリストを作成する関数が，

struct knap *insert(int no, int pr, int w, struct knap *p, struct knap *init);

です．これは，問44のstruct test *insert(char a, struct test *p, struct test *init);

と使い方は同じです．下のように構造体のリストが与えられたとき，weightの小さい順にリストを並べ替えることを考えます．





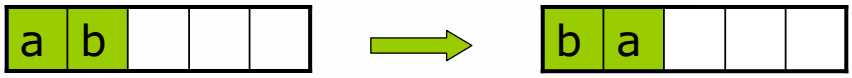
問58　テンプレート中（ホームページにある）の
struct knap *swap_(struct knap *front, struct knap *a, struct knap *b);
struct knap *bubble3(struct knap *front);
を適当に変更して1ページ前の結果が得られるようにせよ.

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

struct knap{
  int no;
  int price;
  int weight;
  struct knap *next;
  };

struct knap *insert(int no, int pr, int pw, struct knap *p, struct knap *init);
struct knap *swap_(struct knap *front, struct knap *a, struct knap *b);
struct knap *bubble3(struct knap *front);


int main(void){
  struct knap *front,*p;

  front = NULL;
  front = insert(1, 23, 58, NULL, front);
  front = insert(2, 18, 37, front, front);
  front = insert(3, 9, 22, front->next, front);
  front = insert(4, 12, 29, front->next->next, front);
  front = insert(5, 38, 41, front->next->next->next, front);

  /* 先頭からデータの表示を行う　*/
  p= front;
  while(p != NULL){
    printf("%d %d %d\n",p->no,p->price, p->weight);
    p = p->next;
  }
  printf("\n");

  front = bubble3(front);

  /* 先頭からデータの表示を行う　*/
  p= front;
  while(p != NULL){
    printf("%d %d %d\n",p->no,p->price, p->weight);
    p = p->next;
  }

  return(0);
}
```

```
[wata@csimc13 ensyuu3]$ gcc ex58.c
[wata@csimc13 ensyuu3]$ ./a.out
1 23 58
2 18 37
3 9 22
4 12 29
5 38 41

3 9 22
4 12 29
2 18 37
5 38 41
1 23 58
[wata@csimc13 ensyuu3]$
```

のようになればOK.

```c
struct knap *insert(int no, int pr, int w, struct knap *p, struct knap *init){
  /*ポインタpの指すセルの次にaのセルを挿入，p=NULLならば先頭に挿入*/
  struct knap *q, *r;

  r = (struct knap *)malloc(sizeof(struct knap));
  if(p==NULL){
    q = init;
    init = r;
    init->no = no;
    init->price = pr;
    init->weight = w;
    init->next = q;
  }else{
    q = p->next;
    p->next = r;
    r->no = no;
    r->price = pr;
    r->weight = w;
    r->next = q;
  }
  return(init);
}

struct knap *swap_(struct knap *front, struct knap *a, struct knap *b){
  struct knap *tmp, *fr, *c;
  if(a == front){
    fr = b;
    tmp = b->next;
    b->next = a;
    a->next = tmp;
  }else{
    fr = front; /*この場合はfrontの入れ替わりは起りません*/
    c = front;
    while(/*ここはaの直前のリストを探す処理*/){
      c = /*適当に埋めよ*/
    }
    /*whileを抜けたらcはaの直前のリストのアドレスになっていなければならない*/

    /* a == frontの場合を参考にして適当に埋める*/

  }
  return(fr);
}
```



aが先頭の場合の処理

問題に直接関係するわけではないが，このswap_は連続した2つのリストしか交換できないことに注意.

問58確認印

```
struct knap *bubble3(struct knap *front){
  int i,j,k = 1;
  int last, n = 0;
  struct knap *p;
  p = front;
  while(p->next != NULL){
    /*ここはリストの長さn求める処理.問題ではリストの長さ=5ですが、それが
      分らないとして*/

  /*適当に埋める*/

  }
  n++;

  while(k < n-1){
    last = n-1;

    for(i=n-1;i>=k;i--){
      p = front;
      for(j=0;/*適当に埋める.ここはリストをi-1番目まで進める処理*/){
        p = /*適当に埋める*/
      }

      if(/*i番目のリストのweightよりi-1番目のリストのweightが大きかったら*/){
        front = swap_(front,p,p->next);
        /* swap_は隣合うリストの交換しかできないのでp->nextは冗長なのですが
           意味をはっきりするためにこうしました*/
        last = /*適当に埋める*/;
      }
    }

    k = /*適当に埋める*/;
  }
  return(front);
}
```

（独り言コーナー）

この部分の処理を

p =front + (i-1)

に置きかえられると思っていたら結構重症かもな.

区別できなかったら，表紙のスタッフに聞いてみよう.

線形リストの探索

```
#include <stdio.h>
#include <stdlib.h>
#define NUMBER 100000

struct knap{
  int no;
  int price;
  int weight;
  struct knap *next;
  };

struct knap *insert(int no, int pr, int w, struct knap *p, struct knap *init)
void find_w(struct knap *p, int w);

int main(void){
  struct knap *front,*pos,*p;
  int d1,d2,d3,i;

  srand(11111);
  front = NULL;
  while(scanf("%d%d%d",&d1,&d2,&d3) != EOF){
    if(front == NULL){
      front = insert(d1, d2, d3, NULL, front);
      pos = front;
    }else{
      front = insert(d1, d2, d3, pos, front);
      pos = pos->next;
    }
  }

  /* 先頭からデータの表示を行う*/
  p= front;
  while(p != NULL){
    printf("%d %d %d\n",p->no,p->price, p->weight);
    p = p->next;
  }
  printf("\n");

  for(i=0;i<NUMBER;i++){
    /*100000回検索*/
    find_w(front, rand()%80);
  }

  return(0);
}
```

長さ100の線形リストを作っている.

問59 kp1.txt(ホームページにある)には商品番号，価値，重量の100個のデータが書かれている．以下のmain関数を見ればわかるように問58のinsert関数を使って

./a.out < kp1.txt

とすれば，100個のデータの線形リストを作ることができる.

ここで，与えられた商品にある重量の商品があるかどうか知りたいとする．もし，検索した重量があれば，該当商品全ての

番号，価値，重量

を印字し，無ければ

"No data for weight %d¥n“

(%dには検索した重量が入る)と印字する関数

void find_w(struct knap *p, int w)

を作成せよ.

テンプレートのように100000商品を検索し，その実行時間を求めること.

time ./a.out < kp1.txt

とすればよい.

問59続き

正しくプログラムが書けているかどうかの確認のためNUMBER=10として実行してみて下さい．以下のようになればOKです

```
5 49 57
74 95 57
No data for weight 10
No data for weight 1
No data for weight 5
No data for weight 21
No data for weight 79
40 41 71
46 27 71
No data for weight 17
No data for weight 4
9 99 31
20 6 31
45 99 31
[wata@csimc13 ensyuu3]$
```

問59確認印

```
#include <stdio.h>
#include <stdlib.h>
#define NUMBER 1000000

struct knap{
  int no;
  int price;
  int weight;
  struct knap *next;
  };

struct knap *insert(int no, int pr, int w, struct knap *p, struct knap *init);
struct knap *swap_(struct knap *front, struct knap *a, struct knap *b);
struct knap *bubble3(struct knap *front);
void find_w2(struct knap *p, int w);

int main(void){
  struct knap *front,*pos,*p;
  int d1,d2,d3,i;

  srand(11111);
  front = NULL;
  while(scanf("%d%d%d",&d1,&d2,&d3) != EOF){
    if(front == NULL){
      front = insert(d1, d2, d3, NULL, front);
      pos = front;
    }else{
      front = insert(d1, d2, d3, pos, front);
      pos = pos->next;
    }
  }

  /* 先頭からデータの表示を行え */
  p = front;
  while(p != NULL){
    printf("%d %d %d\n",p->no,p->price, p->weight);
    p = p->next;
  }
  printf("\n");

  front = bubble3(front);

  for(i=0;i<NUMBER;i++){
    /*10000回検索*/
    find_w2(front, rand()%80);
  }

  return(0);
}
```

確認印

問60 問59においてbubble3によりリストをweightの小さい順に並べ替えることを考える．このことを利用して問59の関数

void find_w(struct knap *p, int w)

を高速化した関数

void find_w2(struct knap *p, int w)

を考えよ．テンプレートを利用してもよい．

kp1.txt, kp2.txt, kp3.txtの実行時間を問59と比較してみよ．

これ以上速くしようと思ったら，データ構造を2分木で表現して2分探索を行うのが良い．9月にできればと思います．

レポート問題（1）

- 問61　以下のようなような動きをする整列法を単純選択ソートという．配列aの最初から見て行き，最小のものが見つかったら，a[0]と交換する．次はa[1]から見て行き，最小のものが見つかったらa[1]と交換する．以下同様．このようなソートを行う関数

  void selection(int *a, int n)

  を作成せよ．

- 問62　バブルソートにおいて比較・交換の操作を末尾側からではなく先頭側から行うように変更せよ．この場合，最大要素が末尾側に移動することになる．

- 問63　7,1,2,3,4,5,6はほぼソート済みであるにもかかわらず，最大の要素7が先頭にあるため，bubble3でも効率の良い整列をすることができない．というのは，bubble3でも7は各ラウンドで1つしか後ろに移動できないためである．そこで奇数ラウンドでは比較・交換の操作を末尾側から行い，偶数ラウンドでは，比較・交換の操作を先頭側から行うことにすれば，効率の良い整列を行うことができる．このアルゴリズムを双方向バブルソート（シェーカーソート）という．双方向バブルソートを行う関数bubble4を作成せよ．
- 問64　問58のbubble3ではweightで整列することしかできなかったが，第2引数を与えて

  struct knap *bubble3(struct knap *front, char c);

  として商品番号no，価値priceでも整列できるように変更せよ．

- 問65　問58のstruct knap *swap_を隣り合うリスト以外でも交換できるように変更してみよ．

レポート問題（2）

- 問66　問58においてstruct knap 型からメンバーstruct knap * nextを除いて新たに構造体struct knap2を定義する．Struct knap2 kp[5]として配列で実現し，リストの削除追加ができない場合を考える．このとき，問58のstruct knap *bubble3(struct knap *front)をvoid bubble3(struct knap *kp)に変更してみよ．関数swap_が必要なくなるので簡単になる．また，配列で構造体リストを作っているので先頭アドレスを返す必要がなくなる．