【目次】

7JFC1121 佐藤圭一
Version.1.001 [99/01/21 22:24]

§１：目的

　各種アプリケーションを設計する段階において、集合の中から目的のデータを探し出すプロセスは、かなり重要な位置を占めている。

　今回は、たくさんのデータの中から目的のデータを探し出す、「探索」のアルゴリズムの種類、及び用法、特徴などを分析するために、各種探索のアルゴリズムを比較・検討してみた。

7JFC1121 佐藤圭一

§２：プログラム概要

プログラムは、pascal(Turbo Pascal 7.0 For PC/AT Compatible)上で動作するよう設計した。又、DOS(MS-DOS)環境であればコンパイルされたプログラムは動作するであろう。

プログラム開発に使用したマシンは、NEC 社製 PC-9821Cb Model2 及び AMD-K6 266MHz 搭載の自作 PC/AT 互換機である。

実行環境については後述する。

プログラムは、「各種アルゴリズムにて探索をし、Order 数もしくは実行時間を標準出力に返す」という動作をする。

分割コンパイル等はしていないが、PASCAL の手続き(Procedure)機能を用いて、プログラム内部をいくつかのブロックに区切ってある。ブロックの内容は、データ生成部・時間取得部・線形探索部・Hash テーブル定義部・Hash 探索部・二分探索部・二分木探索部・二分木探索データ生成部・ソート(並び替え)部・メインルーチンとした。

7JFC1121 佐藤圭一

§３：アルゴリズム概要

基本的に、アルゴリズムは「アルゴリズムとデータ構造(岩波書店刊、1989 年)に基づいたアルゴリズムを使用している。但し、Hash に関しては、今回の探索元があまり大きくない為、単純な Hash テーブルを採用した。

§３．１：線形探索

探索方法としては最も単純な探索方法である。

頭から順番に比較処理を行い、最後まで該当項目が無かった場合、「無し」となる。

§３．２：Hash 法

要素の数だけ「入れ物」を用意しておき、「入れ物」の中に要素の番地等を入れておくことで、一回のみの探索で結果が分かる探索方法である。

が、メモリ領域を消費するので、比較的小さい探索に向いている。

§３．３：二分探索木法

根(ROOT)を頭にして、「大きい」「小さい」のラベルを貼った「枝」を各要素に２本づつ生やしていったデータベース(Binary Tree)を利用する探索方法である。

探索・データ生成共に少し時間・メモリ領域を食うが、Hash ほどではないため、中規模～大規模の探索に向いている。

§３．４：二分探索法

整列されたデータを使用し、半分半分・・・と検索対象を割っていく検索方法である。探索にかかる時間はあまり二分探索木と変わらないが、必ず整列されたデータを使わなければならないため、予め整列されたデータを扱う時に威力を発揮する。

7JFC1121 佐藤圭一

§４：シミュレーションの方法

前述の通り、プログラムは pascal を使用、以下の５つの要素について、Order 数、時間を算出した。

１：要素ランダム、検索対象ランダム

２：要素ランダム、検索対象昇順

３：要素ランダム、検索対象降順

４：要素偶数のみ、検索対象偶数のみ（必ず HIT する）

５：要素奇数のみ、検索対象偶数のみ（必ず HIT しない）

そして、検索対象を１０～９０、１００～９００として計算にかけた。

要素は、１～３は１０００固定、４～５については検索対象と同数とした。

プログラムのコンパイルは自作 PC/AT 互換機、MS-Windows98 上にて Turbo Pascal 7.0 を用いた。又、Order 数の算出には CPU の演算速度そのものは関係しないため、コンパイルに使用した PC と同じ PC を使用した。

時間の算出に使用した PC は、以下の条件を満たしているマシンを使用した。

１：1/100 秒単位の時刻を BIOS が扱えること

２：CPU の演算速度が遅いこと

ループさせるとデータ生成等にかかる時間が不明瞭になる為

３：MS-DOS マシンであること

そこで、富士通社製 FMR-CARD を使用した。スペックは以下の通り。

CPU	intel 80C86 5MHz	OS	日本語 MS-DOS ROM Version3.2 L31 B
Memory	640KB		
DHRYSTONE	17.52sec / 30000Loop	P/N	D12B-0880-B202

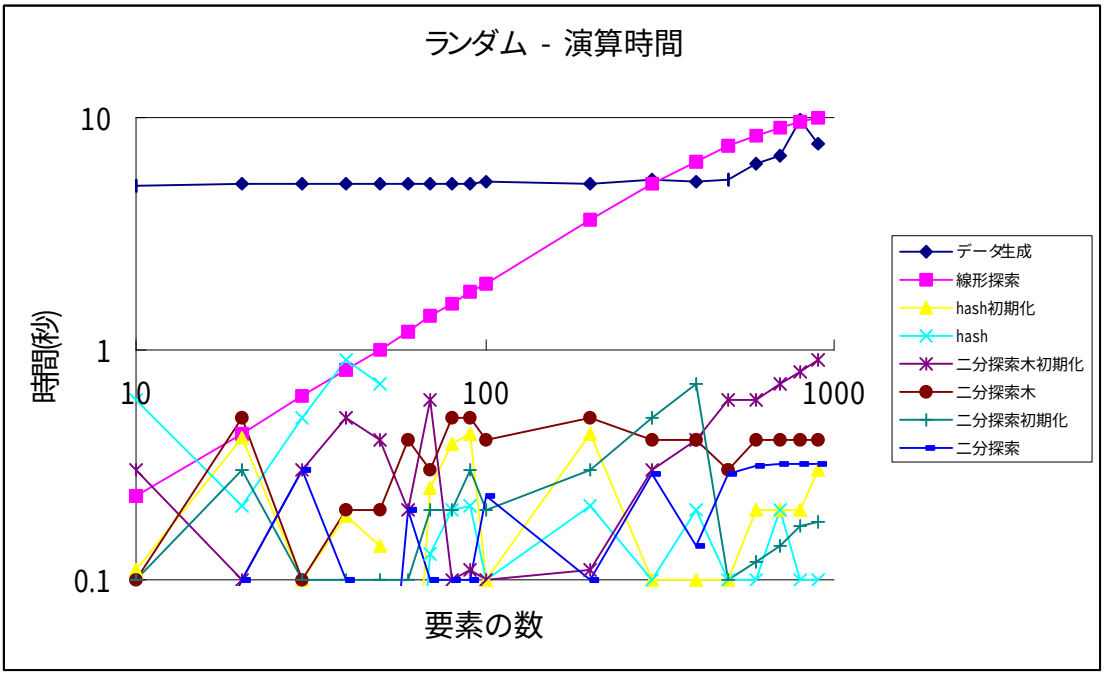
なお、FM-R シリーズ用に最適化されたコンパイラではなく、DOS 汎用のコンパイラを使用した為、性能を生かしきれていない可能性がある。

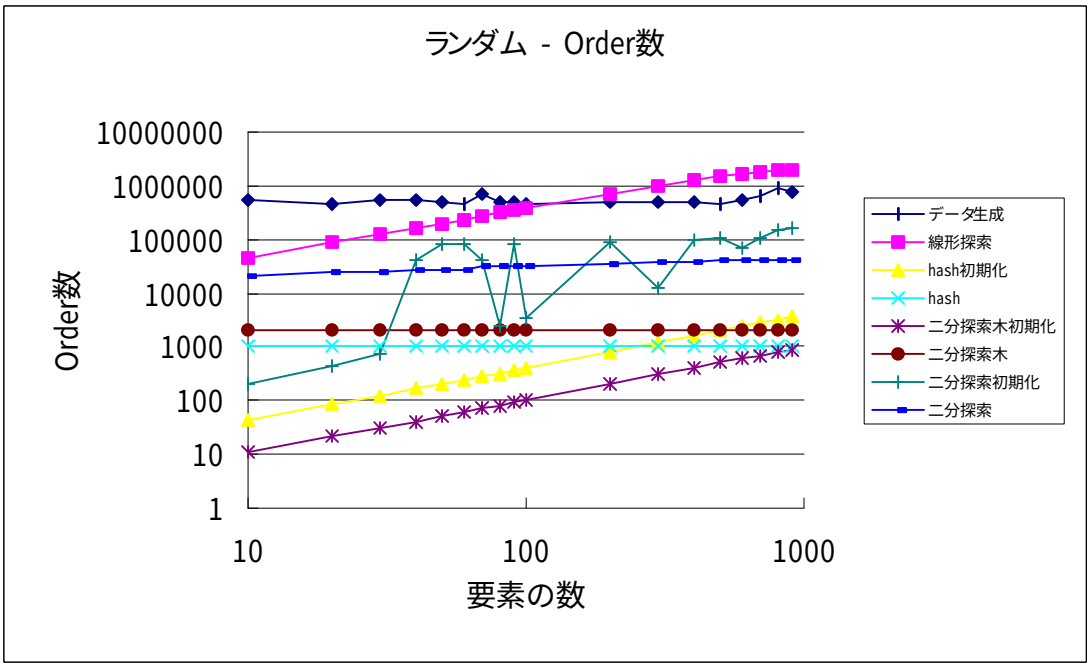
また、データの転送には富士通社製 FM-TOWNS を使用、IC メモリーカードを用いた。

7JFC1121 佐藤圭一

§５：結果

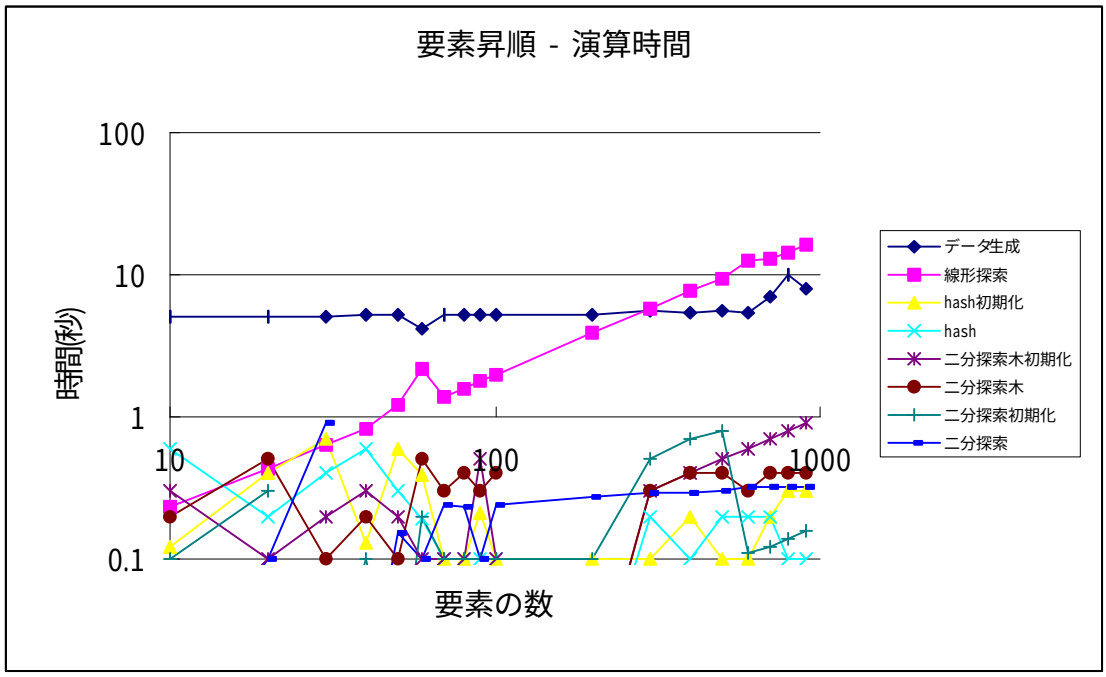
§５．１：ランダムの場合

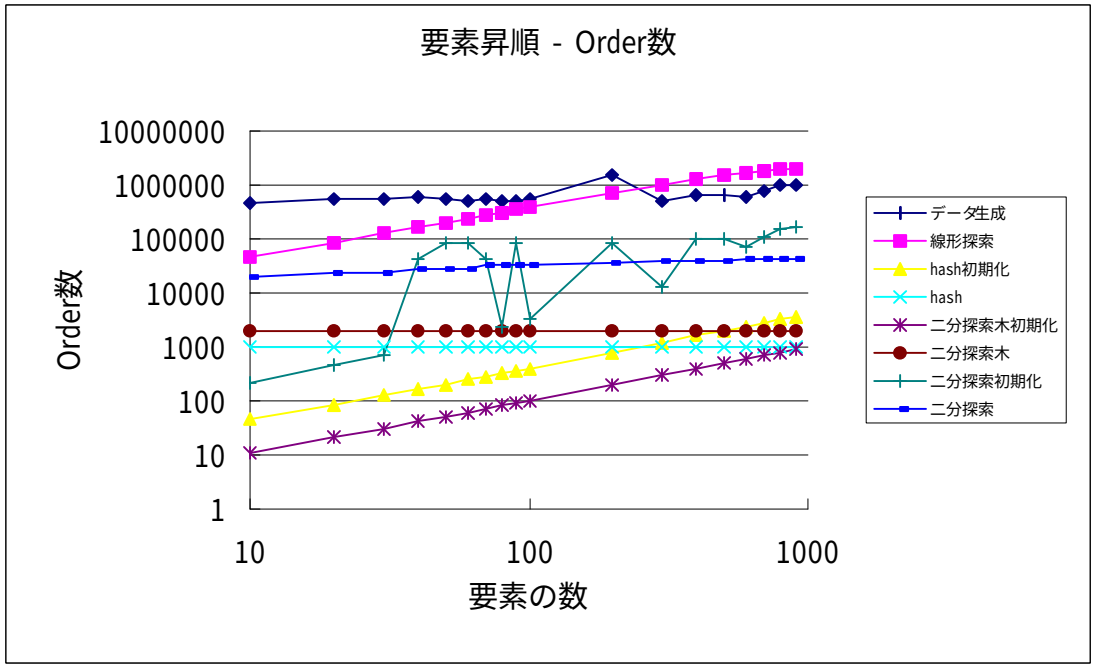




7JFC1121 佐藤圭一
Version.1.001 [99/01/21 22:24]

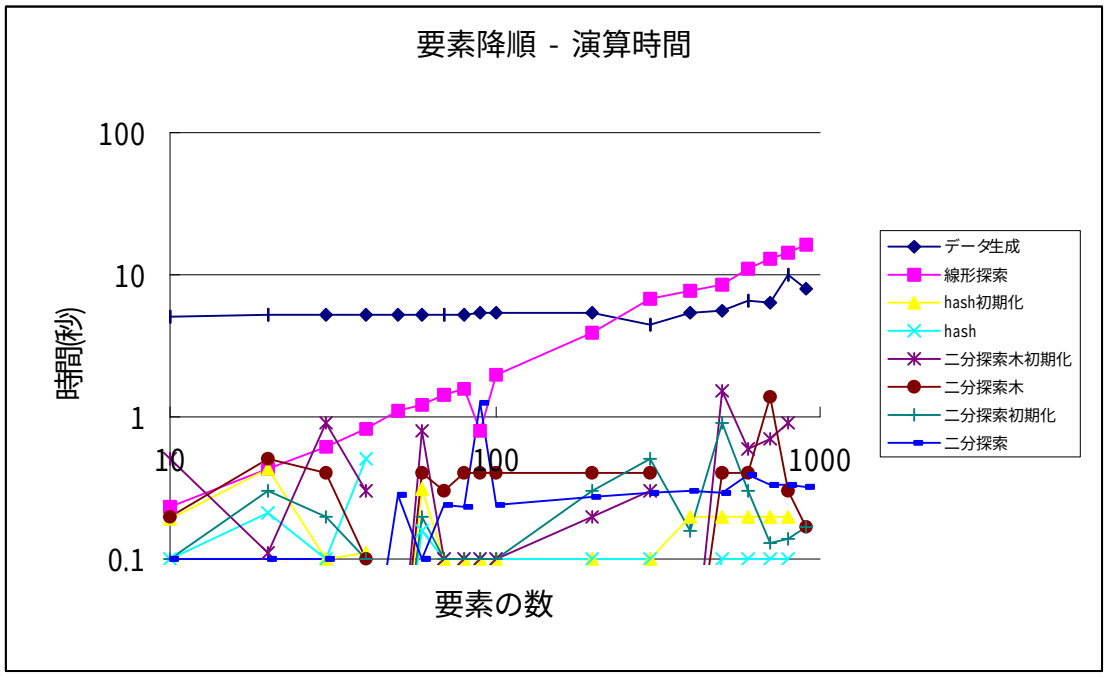
§５．２：要素が昇順に並んでいる場合

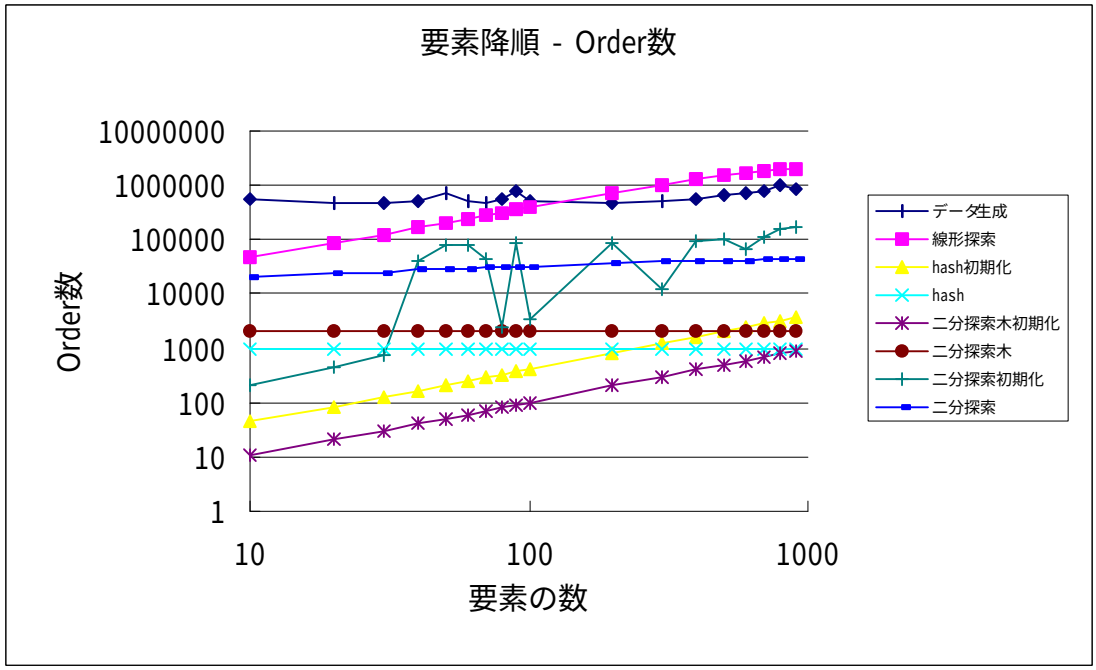




7JFC1121 **佐藤圭一**
Version.1.001 [99/01/21 22:24]

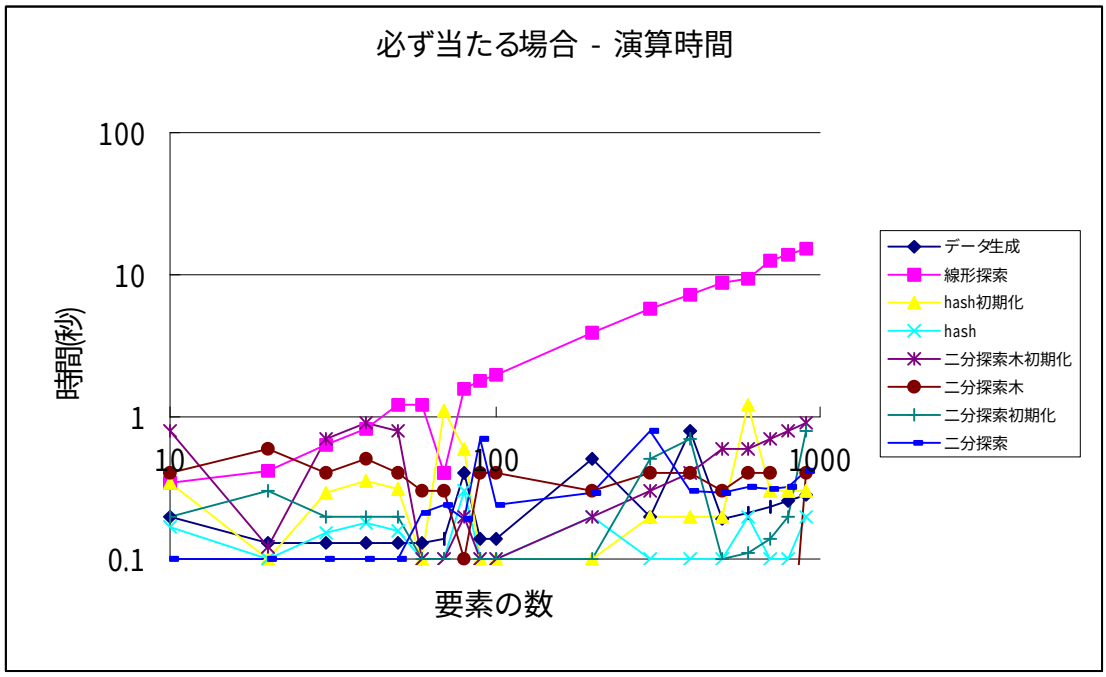
§５．３：要素が降順に並んでいる場合

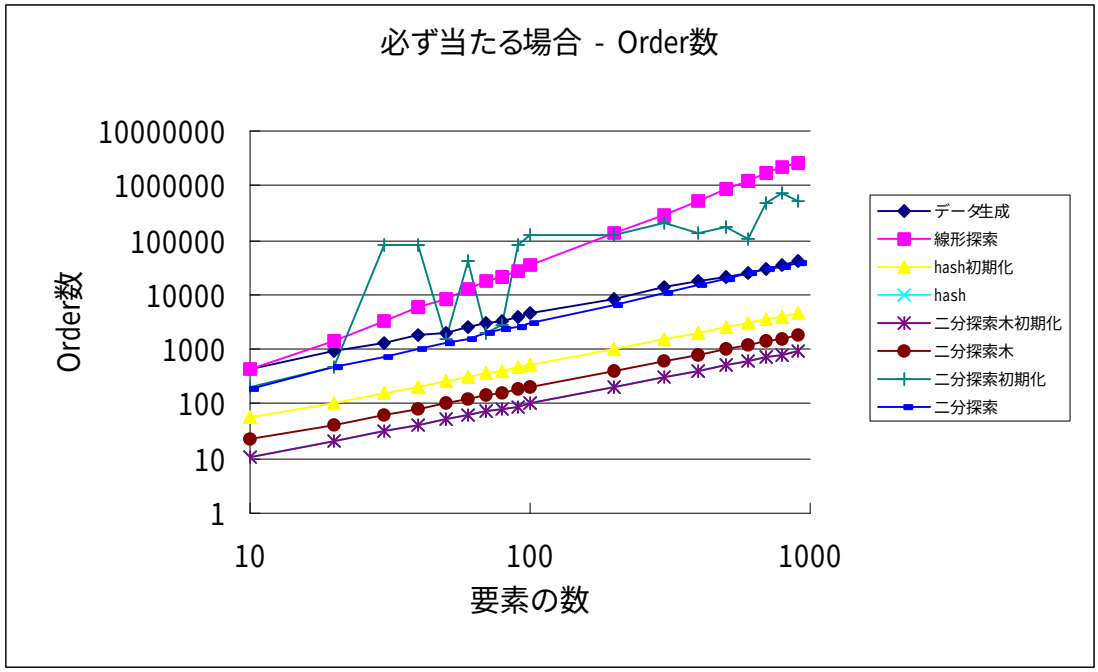




7JFC1121 **佐藤圭一**
Version.1.001 [99/01/21 22:24]

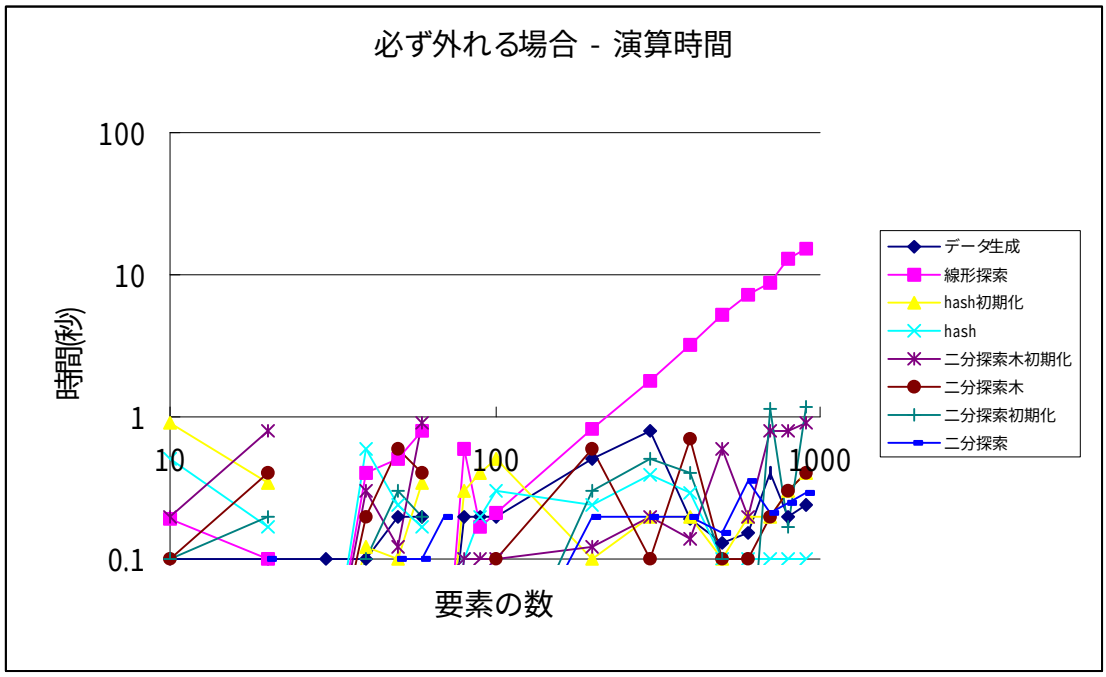
§５．４：検索対象が必ず存在する場合

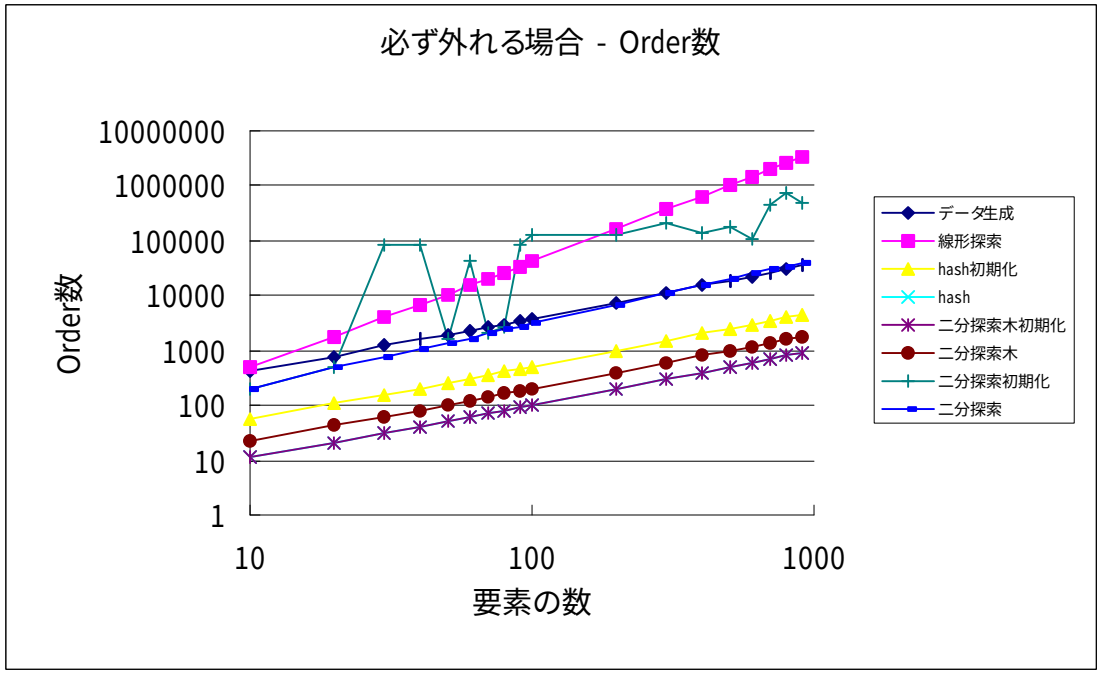




7JFC1121 **佐藤圭一**
Version.1.001 [99/01/21 22:24]

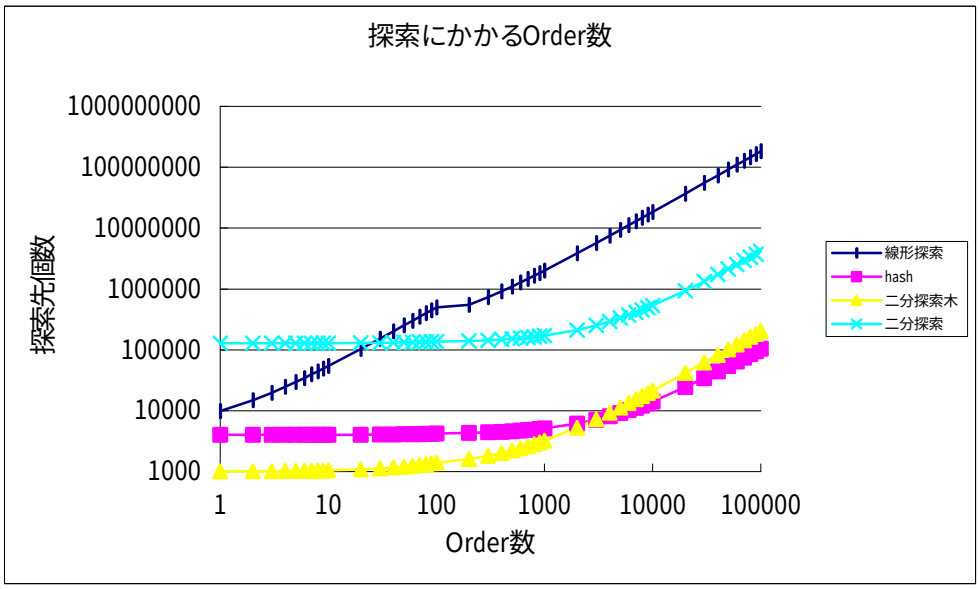
§５．５：検索対象がまったく存在しない場合





7JFC1121 **佐藤圭一**
Version.1.001 [99/01/21 22:24]

§５．６：探索にかかるOrder数



要素の数を一定（１０００個）にし、探索させるデータをループさせて得たグラフである。

初期化と探索の時間を足して算出した。

7JFC1121 佐藤圭一

§６：考察

今回の各種探索方法の比較を、私は次のように分析する。

１：データの並び方は関係ないということ

２：探索するデータの数、探索先のデータの数のバランスにより、最適なアルゴリズムは変わるということ

しかし、このデータを鵜呑みにするのはあまりにも危険である。

今回、比較対象には入れることができなかったが、「使用リソース」という重要な課題も残されている。

また、比較すべき対象が複数の場合や、数字以外（文字などの場合）は、各探索方法を組み合わせて使用するのが最適、というのもありえる。

現に、私は perl 言語にて簡単なデータベースを作った事があるが、その時は「線形探索」と「Hash」を組み合わせて使用した記憶がある。

また、元来 PASCAL がリスト指向の言語ではない為、リスト指向の言語を使用する事により、二分探索木等の探索が高速化する可能性は高い。

要は、最適な探索アルゴリズムを探すには、「やってみるしかない」のが現状のような気がする。

探索アルゴリズムの応用例：

現在のコンピュータに搭載されている CPU は高速・高性能である。が、高度なパイプライン処理や、ステージ予測、更に各種演算ユニットの内蔵等を行ってきた結果、その CPU が持つ力を 100%発揮するには、やはり探索アルゴリズムを応用し、プログラムコードの並び替えを行う必要があるように思う。(もっとも、COMPAQ 社の DEC-ALPHA プロセッサ用のソフトウェア「FX!32」等、コードの最適化を行うプログラムは存在する。)

7JFC1121 佐藤圭一

§７：付録

§７．１：プログラムソース

§７．１．１：探索プログラムの構成

探索プログラムは、標準出力がそのまま CSV ファイルになるように工夫した。
よって、少し分かりにくいかもしれないが、補助バッチファイルの一例を
載せておくので、そちらも参照して欲しい。
また、各種演算の為、プログラムの変数部分などが違う物が多数あることを
白状しておく:-)

§７．１．２：FMR-CARD 演算用探索比較プログラム

```
program search_t; (* 7JFC1121 Keiichi SATO *)

(*
探索アルゴリズム比較プログラム
Made By eucalyptus.(Keiichi SATO) 1998
For FMR-CARD 演算
 *)

uses dos;

(* ＊＊＊＊＊変数宣言＊＊＊＊＊ *)
type ar_int = array[0..10000] of integer;
        tree     = ^node;
        node     = record
           key              : integer;
           left, right : tree;
        end;

var
        p                       : tree;
        look,data,hash : ar_int          ;
        tmp                     : integer;
        temp                    : array[0..10] of integer;
        time                    : array[0..10] of integer;
        time_msec       : array[0..10] of integer;
        num                     : array[0.. 5] of integer;


(* ＊＊＊＊＊手続き宣言＊＊＊＊＊ *)
(* データ生成手続き *)
procedure dataset(var data,hash                                 : ar_int ;
                        data_variation,data_length,flag,temp1 : integer);
   var i,j,k,l,m : integer;

   begin
      for i := 0 to data_variation do
         begin
            hash[i] := 0
         end;

      for i := 0 to data_length do
         begin
            k := 0;
            l := 0;
            repeat
```

```
            l := l + 1;
            Randseed := j + i + l;
            j := Random(data_variation);
               if (flag = 1) then
                  begin
                     j := j * 2;
                  end;
               if (flag = 2) then
                  begin
                     j := j * 2;
                     j := j + 1;
                  end;
            if (hash[j] = 0) then
               begin
                  data[i] := j;
                  hash[j] := 1;
                  k       := 1
               end
         until k = 1
      end
   end;


(*  時間取得・演算手続き  *)
procedure get_sec (var sec1,sec1_msec,sec2,sec2_msec : integer);
var hour,min,sec,sec100 : word;
      sec_temp,sec_temp2    : integer;

   begin
      gettime(hour,min,sec,sec100);
      sec_temp2 := hour * 3600;
      sec_temp    := min * 60;
      sec_temp2 := sec_temp + sec_temp2;
      sec_temp2 := sec + sec_temp2;
      sec2 := sec_temp2 - sec1;
      sec1 := sec_temp2;
      sec_temp := sec100 + 100;
      sec_temp2 := sec_temp - sec1_msec;
      if (sec_temp2 > 99) then
         sec2_msec := sec_temp2 - 100
         else
            begin
               sec2 := sec2 - 1;
               sec2_msec := sec_temp2
            end;
      sec1_msec := sec100
   end;


(*  線形探索手続き  *)
procedure l_search(var data,look : ar_int;
                           var data_length,
                           look_length : integer);
   var i,j,p,a : integer;

   begin
      for p := 0 to look_length do
         begin
            i := 0;
            j := 0;
            repeat
               if data[i] = look[p] then
                  j := 1;
               i := i + 1;
            until (j = 1) or (i > data_length)
```

```
        end
    end;


(*  二分探索手続き  *)
procedure d_search(var data,look    : ar_int;
                               var data_length,
                               look_length       : integer
                              );
  var l,h,p,m : integer;

  begin
     for p := 0 to look_length do
        begin
           l := 0;
           h := data_length;
           while l <= h do
              begin
                 m := (l+h) div 2;
                 if look[p] <= data[m] then
                    h := m - 1;
                 if look[p] >= data[m] then
                    l := m + 1
              end
        end
    end;


(* Hash 法初期化手続き  *)
procedure hash_init(var hash,data             : ar_int;
                                  var data_variation,
                                        data_length         : integer);
  var i,j : integer;

  begin
     for i := 0 to data_variation do
        begin
           hash[i] := 0
        end;
     for i := 0 to data_length do
        begin
           j := data[i];
           hash[j] := 1
        end
  end;


(* Hash 法探索手続き  *)
procedure hash_search(var hash,look     : ar_int;
                                        data_length,
                                        look_length : integer);
var i,p : integer;

  begin
     for p := 0 to look_length do
        begin
           i := hash[p]
        end
    end;


(*  二分探索木挿入手続き  /  教科書仕様通り  *)
procedure btree_insert(var x : integer;
                                         p : tree);
  begin
```

```
    if p = nil then
      begin
        new(p);
        with p^ do
          begin
            key   := x;
            left  := nil;
            right := nil;
          end;
      end
    else
      if x = p^.key then
        begin end
      else
        if x < p^.key
          then btree_insert(x,p^.left)
        else
          btree_insert(x,p^.right);
  end;


(*  二分探索木手続き  *)
procedure btree(target : integer;
                p      : tree     );

  begin
    if p = nil then
      begin end
      else
        if   target = p^.key then
          begin end
        else
          if target < p^.key
            then
              btree(target,p^.left)
            else
              btree(target,p^.right);
  end;


(*  二分探索木連続挿入手続き  *)
procedure btree_insert_array(var data        : ar_int;
                             var p           : tree;
                             var data_length : integer);
  var i : integer;

  begin
    for i := 0 to data_length do
      btree_insert(data[i],p)
  end;


(*  二分探索木連続探索手続き  *)
procedure btree_array(var look        : ar_int;
                      var look_length : integer;
                      var p           : tree);
  var i : integer;

  begin
    for i := 0 to look_length do
        btree(look[i],p)
  end;


(*  クイックソート手続き  *)
```

```
procedure quick_sort(var data                : ar_int     ;
                         left,right          :integer );
  var i,j,k,w,some,pivot : integer;

  begin
    some   := (left + right ) div 2;
    pivot := data[some];
    i        := left;
    j        := right;

    repeat
      begin
        while data[i] < pivot do
          i := i+1;

        while data[j] > pivot do
          j := j-1;

        if (i <= j) then
          begin
            w          := data[i];
            data[j] := w;
            i := i+1;
            j := j-1
          end
      end;
    until i > j;
    if (j > left) then
      quick_sort(data,left,j);

    if (i < right) then
      quick_sort(data,i,right)

  end;


(*  降順クイックソート手続き  *)
procedure quick_sort_rev(var data              : ar_int     ;
                         left,right          :integer );
  var i,j,k,w,some,pivot : integer;

  begin
    some   := (left + right ) div 2;
    pivot := data[some];
    i        := left;
    j        := right;
    repeat
      begin

        while data[i] > pivot do
          i := i+1;

        while data[j] < pivot do
          j := j-1;

        if (i <= j) then
          begin
            w          := data[i];
            data[j] := w;
            i := i+1;
            j := j-1
          end
      end;
    until i > j;
    if (j > left) then
```

```
        quick_sort_rev(data,left,j);

      if (i < right) then
        quick_sort_rev(data,i,right)

  end;


(*  ＊＊＊＊＊メインルーチン＊＊＊＊＊  *)

begin

(*  各種演算変数の初期化  *)
  for tmp := 0 to 5 do
    num[tmp] := tmp;
  for tmp := 0 to 10 do
    time[tmp] := 0;
  for tmp := 0 to 10 do
    time_msec[tmp] := 0;

(*  処理内容登録  *)

(* CSV 方式での入力用  *)
  readln(temp[1],temp[0],temp[3],temp[2],temp[4],temp[5]);

(*  問答方式での入力用  *)
(*    writeln('探索プログラム');
  write('データの要素数を入力して下さい:');
  readln(temp[1]);
  write('データの最大値を入力して下さい:');
  readln(temp[0]);
  write('探索先の要素数を入力して下さい:');
  readln(temp[3]);
  write('探索先の最大値を入力して下さい:');
  readln(temp[2]);
  writeln('データの仕様を入力して下さい');
  write('[0:昇順  1:降順  2:ランダム  3:ランダム(偶数のみ)]:');
  readln(temp[4]);
  writeln('探索先の仕様を入力して下さい');
  write('[0:ランダム  1:ランダム(偶数のみ) 2:ランダム(奇数のみ)]:');
  readln(temp[5]); *)

(*  データ生成  *)
  get_sec(time[0],time_msec[0],time[1],time_msec[1]);
  if (temp[4] <> 3) then
    dataset(data,hash,temp[0],temp[1],num[0],time_msec[0]);
  if (temp[4] =  3) then
    dataset(data,hash,temp[0],temp[1],num[2],time_msec[0]);
  if (temp[4] = 0) then
    quick_sort(data,num[1],temp[1]);
  if (temp[4] = 1) then
    quick_sort_rev(data,num[1],temp[1]);
  if (temp[5] = 0) then
    dataset(look,hash,temp[2],temp[3],num[0],time_msec[0]);
  if (temp[5] = 1) then
    dataset(look,hash,temp[2],temp[3],num[2],time_msec[0]);
  if (temp[5] = 2) then
    dataset(look,hash,temp[2],temp[3],num[1],time_msec[0]);

(*  各種探索手続きの呼出  *)
  get_sec(time[0],time_msec[0],time[1],time_msec[1]);
  l_search(data,look,temp[1],temp[3]);
  get_sec(time[0],time_msec[0],time[2],time_msec[2]);
  hash_init(hash,data,temp[0],temp[1]);
```

```
    get_sec(time[0],time_msec[0],time[3],time_msec[3]);
    hash_search(data,look,temp[1],temp[3]);
    get_sec(time[0],time_msec[0],time[4],time_msec[4]);
    btree_insert_array(data,p,temp[1]);
    get_sec(time[0],time_msec[0],time[5],time_msec[5]);
    btree_array(look,temp[3],p);
    get_sec(time[0],time_msec[0],time[6],time_msec[6]);
    quick_sort(data,1,temp[1]);
    get_sec(time[0],time_msec[0],time[7],time_msec[7]);
    d_search(data,look,temp[1],temp[3]);
    get_sec(time[0],time_msec[0],time[8],time_msec[8]);

(*  結果出力  *)

(* CSV 形式での出力  *)
    write(temp[1]);
    for tmp := 1 to 8 do
      write(',',time[tmp],'.',time_msec[tmp]);
    writeln;

(*  問答形式での出力  *)
(*      writeln('＊＊＊結果＊＊＊');
    writeln('データ生成              にかかった時間 : ',time[1]);
    writeln('線形探索                にかかった時間 : ',time[2]);
    writeln('Hash の初期化           にかかった時間 : ',time[3]);
    writeln('hash                    にかかった時間 : ',time[4]);
    writeln('二分探索木の初期化にかかった時間 : ',time[5]);
    writeln('二分探索木              にかかった時間 : ',time[6]);
    writeln('二分探索の初期化   にかかった時間 : ',time[7]);
    writeln('二分探索                にかかった時間 : ',time[8]); *)
    end.
```


7JFC1121 **佐藤圭一**

§７．１．３：Order 演算プログラム

```
program search_d; (* 7JFC1121 Keiichi SATO *)

(*
探索アルゴリズム比較プログラム
Made By eucalyptus.(Keiichi SATO) 1998
Order 演算  For planetree
 *)

uses dos;

(*  ＊＊＊＊＊変数宣言＊＊＊＊＊  *)
type ar_int = array[0..10010] of integer;
        tree     = ^node;
        node     = record
           key              : integer;
           left, right : tree;
        end;

var
        order                : array[0..8]of longint;
        p                    : tree;
        look,data,hash : ar_int          ;
        tmp                  : integer;
        temp                 : array[0..10] of integer;
        num                  : array[0.. 5] of integer;


(*  ＊＊＊＊＊手続き宣言＊＊＊＊＊  *)
(*  データ生成手続き  *)
procedure dataset(var data,hash                          : ar_int ;
                          data_variation,data_length,flag,temp : integer;
                          var order                              : longint    );
   var i,j,k,l,m : integer;

   begin
      for i := 0 to data_variation do
         begin
            hash[i] := 0
;order := order + 1;
order := order + 1;
         end;

      for i := 0 to data_length do
         begin
            k := 0;
            l := 0;
order := order + 1;
order := order + 1;
             repeat
                l := l + 1;
order := order + 1;
               Randseed := j + i + l + temp;
order := order + 1;
               j := Random(data_variation);
;order := order + 1;
                  if (flag = 1) then
                     begin
order := order + 1;
                        j := j * 2 + 1;
                     end;
order := order + 1;
                  if (flag = 2) then
                     begin
order := order + 1;
                        j := j * 2;
                     end;
order := order + 1;
               if (hash[j] = 0) then
                  begin
order := order + 1;
```

```
                    data[i] := j;
order := order + 1;
                    hash[j] := 1;
order := order + 1;
                    k           := 1;
                  end
;order := order + 1;
          until k = 1
        end
  end;


(*  線形探索手続き  *)
procedure l_search(var data,look : ar_int;
                           var data_length,
                           look_length : integer;
                           var order     : longint );
  var i,j,p,a : integer;

  begin
    for p := 0 to look_length do
      begin
;order := order + 1;
          i := 0;
;order := order + 1;
          j := 0;
;order := order + 1;
          repeat
;order := order + 1;
            if data[i] = look[p] then
begin order := order + 1;
              j := 1 end;
;order := order + 1;
            i := i + 1;
;order := order + 1;
;order := order + 1;
          until (j = 1) or (i > data_length)
        end
   end;


(*  二分探索手続き  *)
procedure d_search(var data,look    : ar_int;
                           var data_length,
                           look_length       : integer;
                           var order          : longint
                          );
  var l,h,p,m : integer;

  begin
    for p := 0 to look_length do
      begin
;order := order + 1;
;order := order + 1;
          l := 0;
;order := order + 1;
          h := data_length;
;order := order + 1;
          while l <= h do
            begin
;order := order + 1;
              m := (l+h) div 2;
;order := order + 1;
              if look[p] <= data[m] then
begin order := order + 1;
                h := m - 1 end;
;order := order + 1;
              if look[p] >= data[m] then
begin order := order + 1;
                l := m + 1 end;
            end
        end
```


7JFC1121 佐藤圭一

```
    end;


(* Hash 法初期化手続き  *)
procedure hash_init(var hash,data            : ar_int;
                                var data_variation,
                                      data_length        : integer;
                                var order                : longint    );
   var i,j : integer;

   begin
      for i := 0 to data_variation do
         begin
            hash[i] := 0
;order := order + 1;
         end;
      for i := 0 to data_length do
         begin
;order := order + 1;
;order := order + 1;
            j := data[i];
;order := order + 1;
            hash[j] := 1
         end
   end;


(* Hash 法探索手続き  *)
procedure hash_search(var hash,look     : ar_int;
                                        data_length,
                                        look_length : integer;
                                        var order     : longint);
var i,p : integer;

   begin
      for p := 0 to look_length do
         begin
;order := order + 1;
            i := hash[p]
         end
     end;


(*  二分探索木挿入手続き  /  教科書仕様通り  *)
procedure btree_insert(var x : integer;
                                          p : tree;
                                          order : longint);
   begin
;order := order + 1;
      if p = nil then
         begin
;order := order + 1;
            new(p);
;order := order + 1;
            with p^ do
               begin
;order := order + 1;
                  key     := x;
;order := order + 1;
                  left    := nil;
;order := order + 1;
                  right := nil;
               end;
         end
      else
;order := order + 1;
         if x = p^.key then
            else
               begin
                  order := order + 1;
                  if x < p^.key then
                     begin
```

```
                    order := order + 1;
                    btree_insert(x,p^.left,order);
                  end
               else
                  begin
                    order := order + 1;
                    btree_insert(x,p^.right,order);
                  end
            end
  end;


(*  二分探索木手続き  *)
procedure btree(target : integer;
                    p          : tree;
                    var order : longint);

  begin
    if p = nil then
       begin
       ;order := order + 1;
       end
       else
          begin
;order := order + 1;
          if   target = p^.key then
             begin
             ;order := order + 1;
             end
           else
             begin
;order := order + 1;
               if target < p^.key then
                  begin
;order := order + 1;
                     btree(target,p^.left,order)
                  end
               else
                  begin
;order := order + 1;
                    btree(target,p^.right,order);
                  end
          end end
  end;


(*  二分探索木連続挿入手続き  *)
procedure btree_insert_array(var data             : ar_int;
                                    var p                 : tree;
                                    var data_length : integer;
                                    var order           : longint);
  var i : integer;

  begin
    for i := 0 to data_length do
       begin
;order := order + 1;
          btree_insert(data[i],p,order)
       end
  end;


(*  二分探索木連続探索手続き  *)
procedure btree_array(var look             : ar_int;
                           var look_length : integer;
                           var p                 : tree;
                           var order           : longint);
  var i : integer;

  begin
    for i := 0 to look_length do
      begin
```

```
;order := order + 1;
            btree(look[i],p,order)
        end
   end;


{クイックソート手続き}

procedure quick_sort(var data          : ar_int   ;
                                left,right :integer   ;
                                var order            : longint);
   var i,j,k,w,some,pivot : integer;

   begin
      some   := (left + right ) div 2;
      pivot := data[some];
      i         := left;
      j         := right;
      order := order + 6;


      repeat
         begin

            while data[i] < pivot do
               begin
                  i := i+1;
                  order := order + 2
               end;

            while data[j] > pivot do
               begin
                 j := j-1;
                  order := order + 2
               end;

            if (i <= j) then
               begin
                  for k := 0 to 2 do
                     begin
                        w                  := data[i];
                        order := order + 2
                     end;
                  i := i+1;
                  j := j-1;
                  order := order + 2
               end;
         order := order + 1
         end;
      order := order + 1;
      until i > j;
      order := order + 1;
      if (j > left) then
         begin
            order := order + 1;
            quick_sort(data,left,j,order)
         end;

      order := order + 1;
      if (i < right) then
         begin
            order := order + 1;
            quick_sort(data,i,right,order)
         end

   end;


{降順クイックソート手続き}

procedure quick_sort_rev(var data          : ar_int   ;
                                          left,right :integer   ;
```

```
                                var order            : longint);
  var i,j,k,w,some,pivot : integer;

  begin
    some   := (left + right ) div 2;
    pivot := data[some];
    i         := left;
    j         := right;
    order := order + 6;


    repeat
      begin

        while data[i] > pivot do
          begin
            i := i+1;
            order := order + 2
          end;

        while data[j] < pivot do
          begin
            j := j-1;
            order := order + 2
          end;

        if (i <= j) then
          begin
            for k := 0 to 2 do
              begin
                w                  := data[i];
                order := order + 2
              end;
            i := i+1;
            j := j-1;
            order := order + 2
          end;
      order := order + 1
      end;
    order := order + 1;
    until i > j;
    order := order + 1;
    if (j > left) then
      begin
        order := order + 1;
        quick_sort_rev(data,left,j,order)
      end;

    order := order + 1;
    if (i < right) then
      begin
        order := order + 1;
        quick_sort_rev(data,i,right,order)
      end

  end;

(*  ＊＊＊＊＊メインルーチン＊＊＊＊＊  *)

begin

(*  各種演算変数の初期化  *)
  for tmp := 0 to 5 do
    num[tmp] := tmp;
  for tmp := 0 to 10 do
    order[tmp] := 0;

(*  処理内容登録  *)

(* CSV 方式での入力用  *)
 readln(temp[1],temp[0],temp[3],temp[2],temp[4],temp[5]);
```

```
(*  問答方式での入力用  *)
(*    writeln('探索プログラム');
   write('データの要素数を入力して下さい:');
   readln(temp[1]);
   write('データの最大値を入力して下さい:');
   readln(temp[0]);
   write('探索先の要素数を入力して下さい:');
   readln(temp[3]);
   write('探索先の最大値を入力して下さい:');
   readln(temp[2]);
   writeln('データの仕様を入力して下さい。');
   write('[0:昇順  1:降順  2:ランダム  3:ランダム(偶数のみ)]:');
   readln(temp[4]);
   writeln('探索先の仕様を入力して下さい。');
   write('[0:ランダム  1:ランダム(偶数のみ) 2:ランダム(奇数のみ)]:');
   readln(temp[5]); *)

(*  データ生成  *)
   if (temp[4] <> 3) then
      dataset(data,hash,temp[0],temp[1],num[0],temp[0],order[0]);
   if (temp[4] =  3) then
      dataset(data,hash,temp[0],temp[1],num[2],order[0],order[0]);
   if (temp[4] = 0) then
      quick_sort(data,num[1],temp[1],order[0]);
   if (temp[4] = 1) then
      quick_sort_rev(data,num[1],temp[1],order[0]);
   if (temp[5] = 0) then
      dataset(look,hash,temp[2],temp[3],num[0],order[0],order[0]);
   if (temp[5] = 1) then
      dataset(look,hash,temp[2],temp[3],num[2],order[0],order[0]);
   if (temp[5] = 2) then
      dataset(look,hash,temp[2],temp[3],num[1],order[0],order[0]);

(*  各種探索手続きの呼出  *)
   l_search(data,look,temp[1],temp[3],order[1]);
   hash_init(hash,data,temp[0],temp[1],order[2]);
   hash_search(data,look,temp[1],temp[3],order[3]);
   btree_insert_array(data,p,temp[1],order[4]);
   btree_array(look,temp[3],p,order[5]);
   quick_sort(data,1,temp[1],order[6]);
   d_search(data,look,temp[1],temp[3],order[7]);

(*  結果出力  *)

(* CSV 形式での出力  *)
write(temp[1],',',order[0],',',order[1],',',order[2],',',order[3],',');
writeln(order[4],',',order[5],',',order[6],',',order[7]);

(*  問答形式での出力  *)
(*    writeln('＊＊＊結果＊＊＊');
   writeln('データ生成              にかかった手順 : ',order[0]);
   writeln('線形探索                にかかった手順 : ',order[1]);
   writeln('Hash の初期化           にかかった手順 : ',order[2]);
   writeln('hash                    にかかった手順 : ',order[3]);
   writeln('二分探索木の初期化にかかった手順 : ',order[4]);
   writeln('二分探索木              にかかった手順 : ',order[5]);
   writeln('二分探索の初期化   にかかった手順 : ',order[6]);
   writeln('二分探索                にかかった手順 : ',order[7]); *)
   end.
```


7JFC1121 **佐藤圭一**

§７．１．４：探索にかかるOrder数比較プログラム

```
program search_q; (* 7JFC1121 Keiichi SATO *)

(*
探索アルゴリズム比較プログラム
Made By eucalyptus.(Keiichi SATO) 1998
Order 比較  For planetree
 *)

uses dos;

(*  ＊＊＊＊＊変数宣言＊＊＊＊＊  *)
type ar_int = array[0..10010] of integer;
        tree     = ^node;
        node     = record
           key              : integer;
           left, right : tree;
        end;

var
        order                : array[0..8]of longint;
        tmp2,tmp3            : longint;
        p                    : tree;
        look,data,hash : ar_int          ;
        tmp                  : integer;
        temp                 : array[0..10] of integer;
        num                  : array[0.. 5] of integer;


(*  ＊＊＊＊＊手続き宣言＊＊＊＊＊  *)
・・・search_dと全く一緒なので省略・・・

(*  ＊＊＊＊＊メインルーチン＊＊＊＊＊  *)

begin

(*  各種演算変数の初期化  *)
   for tmp := 0 to 5 do
     num[tmp] := tmp;
   for tmp := 0 to 10 do
     order[tmp] := 0;

(*  処理内容登録  *)

(* CSV 方式での入力用  *)
  readln(temp[1],temp[0],temp[3],temp[2],temp[4],temp[5],tmp3,temp[8]);

(*  問答方式での入力用  *)
(*    writeln('探索プログラム');
   write('データの要素数を入力して下さい:');
   readln(temp[1]);
   write('データの最大値を入力して下さい:');
   readln(temp[0]);
   write('探索先の要素数を入力して下さい:');
   readln(temp[3]);
   write('探索先の最大値を入力して下さい:');
   readln(temp[2]);
   writeln('データの仕様を入力して下さい ');
   write('[0:昇順  1:降順  2:ランダム  3:ランダム(偶数のみ)]:');
   readln(temp[4]);
   writeln('探索先の仕様を入力して下さい ');
   write('[0:ランダム  1:ランダム(偶数のみ) 2:ランダム(奇数のみ)]:');
   readln(temp[5]); *)

(*  データ生成  *)
   if (temp[4] <> 3) then
      dataset(data,hash,temp[0],temp[1],num[0],temp[0],order[0]);
   if (temp[4] =  3) then
      dataset(data,hash,temp[0],temp[1],num[2],order[0],order[0]);
   if (temp[4] = 0) then
      quick_sort(data,num[1],temp[1],order[0]);
```

```
   if (temp[4] = 1) then
      quick_sort_rev(data,num[1],temp[1],order[0]);
   if (temp[5] = 0) then
      dataset(look,hash,temp[2],temp[3],num[0],order[0],order[0]);
   if (temp[5] = 1) then
      dataset(look,hash,temp[2],temp[3],num[2],order[0],order[0]);
   if (temp[5] = 2) then
      dataset(look,hash,temp[2],temp[3],num[1],order[0],order[0]);

(*  各種探索手続きの呼出  *)
   if (order[8] <> 1) then
   begin
   for tmp2 := 0 to tmp3 do
   l_search(data,look,temp[1],temp[3],order[1]);
   end;
   hash_init(hash,data,temp[0],temp[1],order[2]);
   for tmp2 := 0 to tmp3 do
   hash_search(data,look,temp[1],temp[3],order[3]);
   btree_insert_array(data,p,temp[1],order[4]);
   for tmp2 := 0 to tmp3 do
   btree_array(look,temp[3],p,order[5]);
   quick_sort(data,1,temp[1],order[6]);
   for tmp2 := 0 to tmp3 do
   d_search(data,look,temp[1],temp[3],order[7]);

(*  結果出力  *)

(* CSV 形式での出力  *)
tmp3 := tmp3 * temp[3];
write(tmp3,',',order[0],',',order[1],',',order[2],',',order[3],',');
writeln(order[4],',',order[5],',',order[6],',',order[7]);

(*  問答形式での出力  *)
(*    writeln('＊＊＊結果＊＊＊');
   writeln('データ生成              にかかった手順 : ',order[0]);
   writeln('線形探索                にかかった手順 : ',order[1]);
   writeln('Hash の初期化           にかかった手順 : ',order[2]);
   writeln('hash                    にかかった手順 : ',order[3]);
   writeln('二分探索木の初期化にかかった手順 : ',order[4]);
   writeln('二分探索木              にかかった手順 : ',order[5]);
   writeln('二分探索の初期化   にかかった手順 : ',order[6]);
   writeln('二分探索                にかかった手順 : ',order[7]); *)
   end.
```

§７．１．５：補助バッチファイル For FMR-CARD

FMR-CARD は、DOS のバージョンが 3 の上に、command /c オプションが無い。

よって、下記のバッチファイルを走らせ、$を>に、#を<に対応させる事により、計算処理用バッチファイルを生成した。

```
[c_bat.bat]
echo off
cls
rem bmpl data.bmp
echo echo ランダム >log_fm.bat
echo echo ランダム $ out3.csv>>log_fm.bat
echo  echo  要素数,データ生成,線形探索,hash  初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
$$out3.csv>>log_fm.bat
call log3_exe 10 11 1
call log3_exe 20 21 1
call log3_exe 30 31 1
call log3_exe 40 41 1
call log3_exe 50 51 1
call log3_exe 60 61 1
call log3_exe 70 71 1
call log3_exe 80 81 1
call log3_exe 90 91 1 1
call log3_exe 100 101 1
call log3_exe 200 201 1
call log3_exe 300 301 1
call log3_exe 400 401 1
call log3_exe 500 501 1
call log3_exe 600 601 1
call log3_exe 700 701 1
call log3_exe 800 801 1
call log3_exe 900 901 1
echo echo  要素昇順  >>log_fm.bat
echo echo  要素昇順   $$out3.csv>>log_fm.bat
echo  echo  要素数,データ生成,線形探索,hash  初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
$$out3.csv>>log_fm.bat
call log3_exe 10 11 2
call log3_exe 20 21 2
call log3_exe 30 31 2
call log3_exe 40 41 2
call log3_exe 50 51 2
call log3_exe 60 61 2
call log3_exe 70 71 2
call log3_exe 80 81 2
call log3_exe 90 91 2
call log3_exe 100 101 2
call log3_exe 200 201 2
call log3_exe 300 301 2
call log3_exe 400 401 2
call log3_exe 500 501 2
call log3_exe 600 601 2
call log3_exe 700 701 2
call log3_exe 800 801 2
call log3_exe 900 901 2
echo echo  要素降順  >>log_fm.bat
echo echo  要素降順   $$out3.csv>>log_fm.bat
echo  echo  要素数,データ生成,線形探索,hash  初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
$$out3.csv>>log_fm.bat
call log3_exe 10 11 3
call log3_exe 20 21 3
call log3_exe 30 31 3
call log3_exe 40 41 3
call log3_exe 50 51 3
call log3_exe 60 61 3
call log3_exe 70 71 3
call log3_exe 80 81 3
call log3_exe 90 91 3
call log3_exe 100 101 3
```

```
call log3_exe 200 201 3
call log3_exe 300 301 3
call log3_exe 400 401 3
call log3_exe 500 501 3
call log3_exe 600 601 3
call log3_exe 700 701 3
call log3_exe 800 801 3
call log3_exe 900 901 3
echo echo 必ず当選>>log_fm.bat
echo echo 必ず当たる $$out3.csv>>log_fm.bat
echo echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
$$out3.csv>>log_fm.bat
call log3_exe 10 22 4
call log3_exe 20 42 4
call log3_exe 30 62 4
call log3_exe 40 82 4
call log3_exe 50 102 4
call log3_exe 60 122 4
call log3_exe 70 142 4
call log3_exe 80 162 4
call log3_exe 90 182 4
call log3_exe 100 202 4
call log3_exe 200 402 4
call log3_exe 300 602 4
call log3_exe 400 802 4
call log3_exe 500 1002 4
call log3_exe 600 1202 4
call log3_exe 700 1402 4
call log3_exe 800 1602 4
call log3_exe 900 1802 4
echo echo 必ず落選>>log_fm.bat
echo echo 必ず外れる $$out3.csv>>log_fm.bat
echo echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
$$out3.csv>>log_fm.bat
call log3_exe 10 22 5
call log3_exe 20 42 5
call log3_exe 30 62 5
call log3_exe 40 82 5
call log3_exe 50 102 5
call log3_exe 60 122 5
call log3_exe 70 142 5
call log3_exe 80 162 5
call log3_exe 90 182 5
call log3_exe 100 202 5
call log3_exe 200 402 5
call log3_exe 300 602 5
call log3_exe 400 802 5
call log3_exe 500 1002 5
call log3_exe 600 1202 5
call log3_exe 700 1402 5
call log3_exe 800 1602 5
call log3_exe 900 1802 5

[log3_exe.bat]
goto %3

:1
echo echo %1 %2 1000 1001 2 0 $temp.txt >>log_fm.bat
echo echo %1 / 900 >>log_fm.bat
echo search_t # temp.txt $$out3.csv >>log_fm.bat
goto end

:2
echo echo %1 %2 1000 1001 0 0 $temp.txt >>log_fm.bat
echo echo %1 / 900 >>log_fm.bat
echo search_t # temp.txt $$out3.csv >>log_fm.bat
goto end

:3
echo echo %1 %2 1000 1001 1 0 $temp.txt >>log_fm.bat
echo echo %1 / 900 >>log_fm.bat
echo search_t # temp.txt $$out3.csv >>log_fm.bat
```

```
goto end

:4
echo echo %1 %2 1000 2003 3 1 $temp.txt >>log_fm.bat
echo echo %1 / 900 >>log_fm.bat
echo search_t # temp.txt $$out3.csv >>log_fm.bat
goto end

:5
echo echo %1 %2 %1 %2 3 2 $temp.txt >>log_fm.bat
echo echo %1 / 900 >>log_fm.bat
echo search_t # temp.txt $$out3.csv >>log_fm.bat
goto end

:end
```

§７．１．６：Order 数生成用バッチファイル

[log2.bat]
```
@echo off
echo ランダム > out2.csv
echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索 >>out2.csv
call log2_exe 10 11 1
call log2_exe 20 21 1
call log2_exe 30 31 1
call log2_exe 40 41 1
call log2_exe 50 51 1
call log2_exe 60 61 1
call log2_exe 70 71 1
call log2_exe 80 81 1
call log2_exe 90 91 1 1
call log2_exe 100 101 1
call log2_exe 200 201 1
call log2_exe 300 301 1
call log2_exe 400 401 1
call log2_exe 500 501 1
call log2_exe 600 601 1
call log2_exe 700 701 1
call log2_exe 800 801 1
call log2_exe 900 901 1
echo 要素昇順 >> out2.csv
echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索 >>out2.csv
call log2_exe 10 11 2
call log2_exe 20 21 2
call log2_exe 30 31 2
call log2_exe 40 41 2
call log2_exe 50 51 2
call log2_exe 60 61 2
call log2_exe 70 71 2
call log2_exe 80 81 2
call log2_exe 90 91 2
call log2_exe 100 101 2
call log2_exe 200 201 2
call log2_exe 300 301 2
call log2_exe 400 401 2
call log2_exe 500 501 2
call log2_exe 600 601 2
call log2_exe 700 701 2
call log2_exe 800 801 2
call log2_exe 900 901 2
echo 要素降順 >> out2.csv
echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索 >>out2.csv
call log2_exe 10 11 3
call log2_exe 20 21 3
call log2_exe 30 31 3
call log2_exe 40 41 3
call log2_exe 50 51 3
call log2_exe 60 61 3
call log2_exe 70 71 3
call log2_exe 80 81 3
call log2_exe 90 91 3
call log2_exe 100 101 3
call log2_exe 200 201 3
call log2_exe 300 301 3
call log2_exe 400 401 3
call log2_exe 500 501 3
call log2_exe 600 601 3
call log2_exe 700 701 3
call log2_exe 800 801 3
call log2_exe 900 901 3
echo 必ず当たる >> out2.csv
echo 要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索 >>out2.csv
call log2_exe 10 22 4
call log2_exe 20 42 4
call log2_exe 30 62 4
call log2_exe 40 82 4
```

```
call log2_exe 50 102 4
call log2_exe 60 122 4
call log2_exe 70 142 4
call log2_exe 80 162 4
call log2_exe 90 182 4
call log2_exe 100 202 4
call log2_exe 200 402 4
call log2_exe 300 602 4
call log2_exe 400 802 4
call log2_exe 500 1002 4
call log2_exe 600 1202 4
call log2_exe 700 1402 4
call log2_exe 800 1602 4
call log2_exe 900 1802 4
echo  必ず外れる >> out2.csv
echo  要素数,データ生成,線形探索,hash 初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索 >>out2.csv
call log2_exe 10 22 5
call log2_exe 20 42 5
call log2_exe 30 62 5
call log2_exe 40 82 5
call log2_exe 50 102 5
call log2_exe 60 122 5
call log2_exe 70 142 5
call log2_exe 80 162 5
call log2_exe 90 182 5
call log2_exe 100 202 5
call log2_exe 200 402 5
call log2_exe 300 602 5
call log2_exe 400 802 5
call log2_exe 500 1002 5
call log2_exe 600 1202 5
call log2_exe 700 1402 5
call log2_exe 800 1602 5
call log2_exe 900 1802 5

[log2_exe.bat]
goto %3

:1
echo %1 %2 1000 1001 2 0 >temp.txt
echo ITEM : %1 SEARCHING...
search_c < temp.txt >>out2.csv
goto end

:2
echo %1 %2 1000 1001 0 0 >temp.txt
echo ITEM : %1 SEARCHING...
search_c < temp.txt >>out2.csv
goto end

:3
echo %1 %2 1000 1001 1 0 >temp.txt
echo ITEM : %1 SEARCHING...
search_c < temp.txt >>out2.csv
goto end

:4
echo %1 %2 1000 2003 3 1 >temp.txt
echo ITEM : %1 SEARCHING...
search_c < temp.txt >>out2.csv
goto end

:5
echo %1 %2 %1 %2 3 2 >temp.txt
echo ITEM : %1 SEARCHING...
search_c < temp.txt >>out2.csv
goto end

:end
```

§７．１．７：Order 数比較用バッチファイル

実際はこれと似たようなバッチファイルがもう一個あり、データを貼り合わせてグラフを生成した。

```
[log4.bat]
@echo off
echo  検索対象の個数,データ生成,線形探索,hash  初期化,hash,二分探索木初期化,二分探索木,二分探索初期化,二分探索
>out4.csv
call log4_exp
call log4_exp 0
call log4_exp 00
call log4_exp 000
call log4_exp 0000 1
call log4_exp 00000 1
call log4_exp 000000 1
call log4_exp 0000000 1
call log4_exp 00000000 1

[log4_exp.bat]
@echo off
call log4_exe 1%1 %2
call log4_exe 2%1 %2
call log4_exe 3%1 %2
call log4_exe 4%1 %2
call log4_exe 5%1 %2
call log4_exe 6%1 %2
call log4_exe 7%1 %2
call log4_exe 8%1 %2
call log4_exe 9%1 %2

[log4_exe.bat]
echo 1000 1001 1 1000 2 0 %1 %2>temp.txt
echo ITEM : %1 SEARCHING...
search_q < temp.txt >>out4.csv
```


7JFC1121 佐藤圭一

§７．２：演算結果

CSV 形式で得られた演算結果を付録として添付しておく。

§７．２．１：演算時間

数値形式の出力。実際は CSV（カンマ区切り）ファイルである。

ランダム

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	5.12	0.23	0.11	0.6	0.3	0.1	0.1	0
20	5.13	0.43	0.41	0.21	0.1	0.5	0.3	0.1
30	5.13	0.62	0.1	0.5	0.3	0.1	0.1	0.3
40	5.14	0.81	0.19	0.9	0.5	0.2	0.1	0.1
50	5.17	1	0.14	0.7	0.4	0.2	0.1	0
60	5.19	1.19	0	0	0.2	0.4	0.1	0.2
70	5.18	1.38	0.25	0.13	0.6	0.3	0.2	0.1
80	5.2	1.56	0.39	0.2	0.1	0.5	0.2	0.1
90	5.22	1.75	0.43	0.21	0.11	0.5	0.3	0.1
100	5.24	1.92	0.1	0.1	0.1	0.4	0.2	0.23
200	5.22	3.63	0.43	0.21	0.11	0.5	0.3	0.1
300	5.36	5.13	0.1	0.1	0.3	0.4	0.5	0.29
400	5.32	6.42	0.1	0.2	0.4	0.4	0.7	0.14
500	5.42	7.53	0.1	0.1	0.6	0.3	0.1	0.29
600	6.33	8.42	0.2	0.1	0.6	0.4	0.12	0.31
700	6.83	9.12	0.2	0.2	0.7	0.4	0.14	0.32
800	9.71	9.63	0.2	0.1	0.8	0.4	0.17	0.32
900	7.75	9.93	0.3	0.1	0.9	0.4	0.18	0.32

要素昇順

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	5.13	0.23	0.12	0.6	0.3	0.2	0.1	0
20	5.14	0.43	0.4	0.2	0.1	0.5	0.3	0.1
30	5.13	0.63	0.7	0.4	0.2	0.1	0	0.9
40	5.16	0.82	0.13	0.6	0.3	0.2	0.1	0
50	5.18	1.2	0.6	0.3	0.2	0.1	0	0.15
60	4.2	2.21	0.39	0.19	0.1	0.5	0.2	0.1
70	5.2	1.4	0.1	0.1	0.1	0.3	0.1	0.24
80	5.22	1.6	0.1	0.1	0.1	0.4	0.1	0.23
90	5.24	1.8	0.21	0.1	0.5	0.3	0.1	0.1
100	5.26	1.99	0.1	0.1	0.1	0.4	0.1	0.24
200	5.26	3.9	0.1	0	0	0	0.1	0.27
300	5.65	5.68	0.1	0.2	0.3	0.3	0.5	0.29
400	5.39	7.62	0.2	0.1	0.4	0.4	0.7	0.29
500	5.52	9.47	0.1	0.2	0.5	0.4	0.8	0.3
600	5.46	12.5	0.1	0.2	0.6	0.3	0.11	0.32
700	6.98	12.88	0.2	0.2	0.7	0.4	0.12	0.32
800	9.89	14.28	0.3	0.1	0.8	0.4	0.14	0.32
900	7.94	16.38	0.3	0.1	0.9	0.4	0.16	0.32

要素降順

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	5.13	0.23	0.19	0.1	0.5	0.2	0.1	0.1

20	5.23	0.43	0.43	0.21	0.11	0.5	0.3	0.1
30	5.19	0.62	0.1	0.1	0.9	0.4	0.2	0.1
40	5.24	0.83	0.11	0.5	0.3	0.1	0.1	0
50	5.28	1.1	0	0	0	0	0	0.28
60	5.2	1.22	0.31	0.16	0.8	0.4	0.2	0.1
70	5.2	1.41	0.1	0.1	0.1	0.3	0.1	0.24
80	5.31	1.59	0.1	0.1	0.1	0.4	0.1	0.23
90	5.42	0.8	0.1	0.1	0.1	0.4	0.1	1.24
100	5.47	1.99	0.1	0.1	0.1	0.4	0.1	0.24
200	5.44	3.91	0.1	0.1	0.2	0.4	0.3	0.27
300	4.43	6.74	0.1	0.1	0.3	0.4	0.5	0.29
400	5.4	7.6	0.2	0	0	0	0.16	0.3
500	5.52	8.42	0.2	0.1	1.5	0.4	0.9	0.29
600	6.55	10.99	0.2	0.1	0.6	0.4	0.3	0.39
700	6.3	12.89	0.2	0.1	0.7	1.4	0.13	0.33
800	9.99	14.35	0.2	0.1	0.9	0.3	0.14	0.33
900	7.95	16.1	0	0	0	0.17	0.17	0.32

必ず当たる

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	0.2	0.34	0.34	0.17	0.8	0.4	0.2	0.1
20	0.13	0.42	0.1	0.1	0.12	0.6	0.3	0.1
30	0.13	0.63	0.29	0.15	0.7	0.4	0.2	0.1
40	0.13	0.82	0.36	0.18	0.9	0.5	0.2	0.1
50	0.13	1.2	0.31	0.16	0.8	0.4	0.2	0.1
60	0.13	1.21	0.1	0.1	0.1	0.3	0.1	0.21
70	0.14	0.4	1.1	0.1	0.1	0.3	0.1	0.24
80	0.4	1.6	0.6	0.3	0.2	0.1	0	0.19
90	0.14	1.78	0.1	0.1	0.1	0.4	0.1	0.7
100	0.14	1.97	0.1	0.1	0.1	0.4	0.1	0.24
200	0.5	3.94	0.1	0.2	0.2	0.3	0.1	0.29
300	0.2	5.71	0.2	0.1	0.3	0.4	0.5	0.8
400	0.8	7.29	0.2	0.1	0.4	0.4	0.7	0.3
500	0.19	8.75	0.2	0.1	0.6	0.3	0.1	0.29
600	0.21	9.43	1.2	0.2	0.6	0.4	0.11	0.32
700	0.23	12.5	0.3	0.1	0.7	0.4	0.14	0.31
800	0.26	13.62	0.3	0.1	0.8	0	0.2	0.32
900	0.28	15.17	0.3	0.2	0.9	0.4	0.8	0.42

必ず外れる

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	0.1	0.19	0.9	0.5	0.2	0.1	0.1	0
20	0.1	0.1	0.34	0.17	0.8	0.4	0.2	0.1
30	0.1	0	0	0	0	0	0	0
40	0.1	0.4	0.12	0.6	0.3	0.2	0.1	0
50	0.2	0.5	0.1	0.24	0.12	0.6	0.3	0.1
60	0.2	0.8	0.34	0.17	0.9	0.4	0.2	0.1
70	0	0	0	0	0	0	0	0.2
80	0.2	0.6	0.3	0.1	0.1	0	0	0
90	0.2	0.17	0.4	0.2	0.1	0	0	0
100	0.2	0.21	0.5	0.3	0.1	0.1	0	0
200	0.5	0.82	0.1	0.24	0.12	0.6	0.3	0.2
300	0.8	1.81	0.2	0.39	0.2	0.1	0.5	0.2
400	0.19	3.22	0.2	0.29	0.14	0.7	0.4	0.2

7JFC1121 **佐藤圭一**

500	0.13	5.3	0.1	0.1	0.6	0.1	0.1	0.15
600	0.15	7.24	0.2	0.1	0.2	0.1	0	0.36
700	0.4	8.84	0.2	0.1	0.8	0.2	1.14	0.21
800	0.2	12.85	0.3	0.1	0.8	0.3	0.17	0.25
900	0.24	15.25	0.4	0.1	0.9	0.4	1.18	0.29

7JFC1121 **佐藤圭一**

§７．２．２：Order 数

ランダム

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	528482	46838	45	1001	11	2002	208	19998
20	448818	86268	85	1001	21	2002	450	23948
30	560416	125298	125	1001	31	2002	717	23989
40	541706	163928	165	1001	41	2002	41124	27910
50	504696	202158	205	1001	51	2002	81238	27846
60	467794	239988	245	1001	61	2002	81704	27934
70	681542	277418	285	1001	71	2002	42252	31753
80	486102	314448	325	1001	81	2002	2440	31794
90	491128	351078	365	1001	91	2002	83107	31844
100	457184	387308	405	1001	101	2002	3308	31741
200	477858	727608	805	1001	201	2002	87589	35542
300	505252	1027908	1205	1001	301	2002	12720	39066
400	493784	1288208	1605	1001	401	2002	97819	39458
500	463056	1508508	2005	1001	501	2002	102887	39741
600	545878	1688808	2405	1001	601	2002	68488	42037
700	644480	1829108	2805	1001	701	2002	110249	42489
800	892626	1929408	3205	1001	801	2002	152446	42801
900	740758	1989708	3605	1001	901	2002	164055	42945

要素昇順

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	465324	46838	45	1001	11	2002	208	19998
20	535116	86268	85	1001	21	2002	450	23948
30	553513	125298	125	1001	31	2002	717	23989
40	586268	163928	165	1001	41	2002	41124	27910
50	552364	202158	205	1001	51	2002	81238	27846
60	510684	239988	245	1001	61	2002	81704	27934
70	560618	277418	285	1001	71	2002	42252	31753
80	485182	314448	325	1001	81	2002	2440	31794
90	513125	351078	365	1001	91	2002	83107	31844
100	546628	387308	405	1001	101	2002	3308	31741
200	1581877	727608	805	1001	201	2002	87589	35542
300	509938	1027908	1205	1001	301	2002	12720	39066
400	632889	1288208	1605	1001	401	2002	97819	39458
500	654911	1508508	2005	1001	501	2002	102887	39741
600	606326	1688808	2405	1001	601	2002	68488	42037
700	792703	1829108	2805	1001	701	2002	110249	42489
800	1025332	1929408	3205	1001	801	2002	152446	42801
900	1009261	1989708	3605	1001	901	2002	164055	42945

要素降順

10	552540	46838	45	1001	11	2002	208	19998
20	471998	86268	85	1001	21	2002	450	23948
30	456772	125298	125	1001	31	2002	717	23989
40	528273	163928	165	1001	41	2002	41124	27910
50	703801	202158	205	1001	51	2002	81238	27846
60	514915	239988	245	1001	61	2002	81704	27934
70	479681	277418	285	1001	71	2002	42252	31753
80	543151	314448	325	1001	81	2002	2440	31794
90	757265	351078	365	1001	91	2002	83107	31844

7JFC1121 佐藤圭一

100	509232	387308	405	1001	101	2002	3308	31741
200	479238	727608	805	1001	201	2002	87589	35542
300	519781	1027908	1205	1001	301	2002	12720	39066
400	576189	1288208	1605	1001	401	2002	97819	39458
500	649415	1508508	2005	1001	501	2002	102887	39741
600	711173	1688808	2405	1001	601	2002	68488	42037
700	781558	1829108	2805	1001	701	2002	110249	42489
800	974651	1929408	3205	1001	801	2002	152446	42801
900	876770	1989708	3605	1001	901	2002	164055	42945

必ず当たる

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	440	433	56	11	11	22	198	193
20	942	1474	106	21	21	42	490	478
30	1346	3384	156	31	31	62	80796	722
40	1869	6185	206	41	41	82	81897	1017
50	2056	8704	256	51	51	102	1591	1290
60	2656	12638	306	61	61	122	41957	1554
70	2983	17287	356	71	71	142	2035	2040
80	3366	21486	406	81	81	162	2703	2317
90	4071	28278	456	91	91	182	82768	2652
100	4650	33591	506	101	101	202	123176	3050
200	8690	133102	1006	201	201	402	126897	6716
300	13437	295788	1506	301	301	602	210662	10955
400	17470	535332	2006	401	401	802	135034	14889
500	21615	843566	2506	501	501	1002	179335	19154
600	25431	1197789	3006	601	601	1202	106969	24694
700	30654	1638927	3506	701	701	1402	461325	28800
800	35506	2131714	4006	801	801	1602	744991	33391
900	42346	2653317	4506	901	901	1802	507393	38181

必ず外れる

要素数	データ生成	線形探索	hash初期化	hash	二分探索木初期化	二分探索木	二分探索初期化	二分探索
10	419	517	56	11	11	22	198	196
20	739	1827	106	21	21	42	490	484
30	1255	3937	156	31	31	62	80796	744
40	1624	6847	206	41	41	82	81897	1028
50	1874	10557	256	51	51	102	1591	1336
60	2229	15067	306	61	61	122	41957	1608
70	2633	20377	356	71	71	142	2035	2056
80	3009	26487	406	81	81	162	2703	2372
90	3413	33397	456	91	91	182	82768	2700
100	3880	41107	506	101	101	202	123176	3132
200	7514	162207	1006	201	201	402	126897	6808
300	11372	363307	1506	301	301	602	210662	11112
400	15594	644407	2006	401	401	802	135034	14956
500	18745	1005507	2506	501	501	1002	179335	19392
600	22302	1446607	3006	601	601	1202	106969	25052
700	26335	1967707	3506	701	701	1402	461325	29336
800	30354	2568807	4006	801	801	1602	744991	33924
900	34933	3249907	4506	901	901	1802	507393	38576

7JFC1121 **佐藤圭一**

§７．１．３：Order 比較表

検索対象の個数	線形探索	hash	二分探索木	二分探索
1	9864	4009	1009	128096
2	14796	4011	1013	128184
3	19728	4013	1017	128272
4	24660	4015	1021	128360
5	29592	4017	1025	128448
6	34524	4019	1029	128536
7	39456	4021	1033	128624
8	44388	4023	1037	128712
9	49320	4025	1041	128800
10	54252	4027	1045	128888
20	103572	4047	1085	129768
30	152892	4067	1125	130648
40	202212	4087	1165	131528
50	251532	4107	1205	132408
60	300852	4127	1245	133288
70	350172	4147	1285	134168
80	399492	4167	1325	135048
90	448812	4187	1365	135928
100	498132	4207	1405	135938
200	548820	4308	1607	139947
300	731760	4409	1809	143956
400	914700	4510	2011	147965
500	1097640	4611	2213	151974
600	1280580	4712	2415	155983
700	1463520	4813	2617	159992
800	1646460	4914	2819	164001
900	1829400	5015	3021	168010
1000	2012340	5116	3223	172019
2000	3841740	6126	5243	212109
3000	5671140	7136	7263	252199
4000	7500540	8146	9283	292289
5000	9329940	9156	11303	332379
6000	11159340	10166	13323	372469
7000	12988740	11176	15343	412559
8000	14818140	12186	17363	452649
9000	16647540	13196	19383	492739
10000	18476940	14206	21403	532829
20000	36770940	24306	41603	933729
30000	55064940	34406	61803	1334629
40000	73358940	44506	82003	1735529
50000	91652940	54606	102203	2136429
60000	1.1E+08	64706	122403	2537329
70000	1.28E+08	74806	142603	2938229
80000	1.47E+08	84906	162803	3339129
90000	1.65E+08	95006	183003	3740029
100000	1.83E+08	105106	203203	4140929

7JFC1121 佐藤圭一

§８：参考文献

アルゴリズムとデータ構造

岩波書店刊行　1993 **年**

著者：岩畑　清

ISBN4-00-010343-1 C3355 ¥3900E

TURBO Pascal プログラミング

森北出版株式会社刊行

著者：黒瀬能筆・松尾俊彦

ISBN4-627-83241-9 C1050 P2163E

初級シスアド厳選４５ポイント速習

学研刊行　合格情報処理 1998 **年１１月号付録**

合格情報処理編集部編

雑誌コード付属せず

§９：感想

とてもではないが、今回は時間が足りなかった。よって、プログラムの検証もロクにしてなく、動作もかなり怪しい。

でも、今回は探索だけではなく、レポートのフォーマット等、勉強になる部分が多かった。より見やすいレポートを書く為に、敢えてプログラムソース及び長ったらしい出力を付録に置いた。

また、Loop 要素を極力排除する為、計算機の選定にもほとほと手を焼いた。

結論としては、NEC 社製及び EPSON 社製の PC-9800 シリーズ互換機は、BIOS レベルで 1/100 秒の秒の演算を行っていないようで、演算処理をあきらめた;-)

レポートの「ヒント」として、Loop 処理を使用する様指示されていたが、「初期化」の要素を組み入れたかった為、敢えて Loop は排除してみた。

本文中でも少し触れたが、私は perl にて簡単なデータベースを昔、作った。そのデータベースは今でも一応利用されている。詳細は「http://www.scoutnet.or.jp/~bsf3/」の活動記録を見て欲しい。

7JFC1121 佐藤圭一