

平成 19 年度 秋期
ソフトウェア開発技術者
午後Ⅱ 問題

試験時間 15:30 ~ 16:30 (1 時間)

注意事項

1. 試験開始の合図があるまで、問題冊子を開いて中を見てはいけません。
2. この注意事項は、問題冊子の裏表紙に続きます。必ず読んでください。
3. 答案用紙への受験番号などの記入は、試験開始の合図があってから始めてください。
4. 問題は、次の表に従って解答してください。

問題番号	問 1
選択方法	必須

5. 答案用紙の記入に当たっては、次の指示に従ってください。
 - (1) HB の黒鉛筆又はシャープペンシルを使用してください。
 - (2) 受験番号欄に、受験番号を記入してください。正しく記入されていない場合は、採点されません。
 - (3) 生年月日欄に、受験票に印字されているとおりの生年月日を記入してください。正しく記入されていない場合は、採点されないことがあります。
 - (4) 解答は、問題番号ごとに指定された枠内に記入してください。
 - (5) 解答は、丁寧な字ではっきりと書いてください。読みにくい場合は、減点の対象になります。

注意事項は問題冊子の裏表紙に続きます。
こちら側から裏返して、必ず読んでください。

問1 B木に関する次の記述を読んで、設問1～4に答えよ。

B木は、外部記憶装置にデータを格納するために考えられた、木構造のデータ構造である。ここでは、データを識別するキーを正の整数として、キーだけをB木に格納する場合を考える。ただし、キーの重複はないものとする。

B木の各ノードは図1に示す構造をしており、木全体として次の(1)～(8)の性質をもっている。ただし、一つのノードにはキーを n 個まで格納できるものとする。また、 n は偶数で、 $M=n/2$ とする。

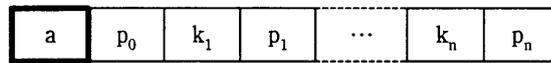


図1 ノードの構造

- (1) $k_1 \sim k_n$ にはキーが格納される。また、ノード内に格納されているキーの個数が a に格納される。
- (2) ノード内では、キーは昇順に並んでいる。つまり、 $k_1 < k_2 < \dots < k_a$ である。
- (3) $p_0 \sim p_n$ には子ノードへのポインタが格納される。
- (4) p_0 が指す部分木内のキーはすべて k_1 より小さく、 $1 \leq i < a$ の p_i が指す部分木内のキーは、すべて k_i より大きくかつ k_{i+1} より小さい。また、 p_a が指す部分木内のキーは、すべて k_a より大きい。
- (5) 子ノードが存在しない場合、該当するポインタはnilである。
- (6) 根のノードには $1 \sim n$ 個のキーが、根以外のノードには $M \sim n$ 個のキーが格納されている。つまり、根以外のノードでは、キーは格納できる個数の半数以上格納されている。
- (7) 葉のノード以外では、ノード内のキーの個数に1を加えた個数のポインタが格納されている。
- (8) 根から葉のノードまでの経路の長さ(深さ)は、どの葉のノードも同じである。

$n=2$ ($M=1$) の場合の B 木の例を図 2 に示す。

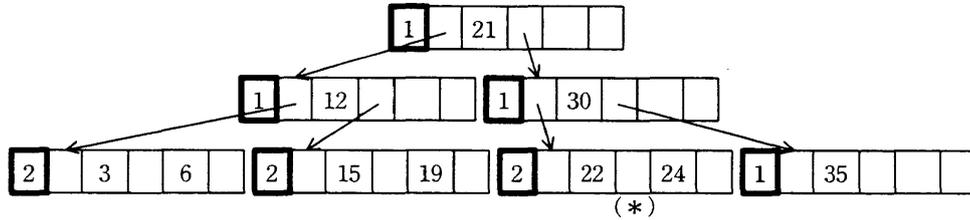


図2 B木の例

B 木に対する操作として、指定されたキーが B 木内にあるかどうかを調べる“探索”と、指定されたキーを B 木に格納する“挿入”の二つの操作を考える。

〔探索〕

指定されたキー x について、根のノードから順に、各ノードに格納されているキー k_i と x を順番に比較しながら、次の動作を行う。

- ・ $x = k_i$ となる k_i が見つければ、目的の値が見つかり探索終了
- ・ $x < k_1$ ならば、ポインタ p_0 をたどって、次のノードを探索
- ・ $k_i < x < k_{i+1}$ ($1 \leq i < a$) ならば、ポインタ p_i をたどって、次のノードを探索
- ・ $k_a < x$ ならば、ポインタ p_a をたどって、次のノードを探索
- ・ たどろうとしたポインタが nil ならば、目的の値は見つからずに探索終了

〔挿入〕

指定されたキー x を挿入する場合、上の〔探索〕と同様にして、キー x が入る可能性があるノードを探索する。探索しているノードでキー x が見つかったときは、何もせずに終了する。一方、葉のノードに到達してもキー x が見つからないときは、到達した葉のノード、又は新たに用意するノードに x を挿入する。すなわち x をノードに挿入する場合、次の二つの可能性が考えられる。

- (1) 到達したノードに新たにキーを挿入する場所がある場合、そのノードの適切な場所に x を挿入する。
- (2) 到達したノードに新たにキーを挿入する場所がない場合、新たにノードを用意し、到達したノードに格納されているキーと x の合計 $n+1$ ($= 2M+1$) 個のキーを昇順

B木のノードをプログラムで扱うために、データ構造 node を定義する。node は図 4 に示すように、node 内に保持しているキーの個数をもつ変数 n_key と、それぞれ 2M, 2M+1 個の配列変数である key, branch から構成される。key にはノード内のキーが、branch にはポインタが入る。初期値として、branch には nil を入れる。p をデータ構造 node を指すポインタとして、node 内の変数は、それぞれ p->n_key, p->key[i] (1 ≤ i ≤ 2M), p->branch[j] (0 ≤ j ≤ 2M) と表記する。

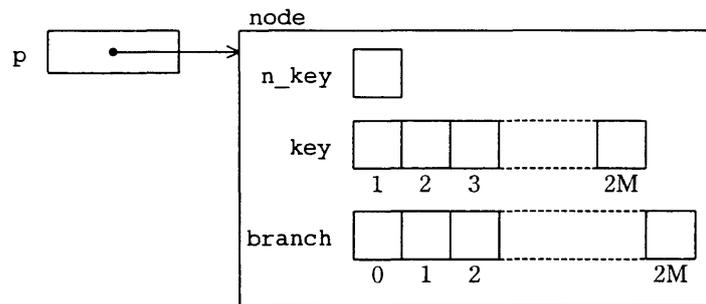


図 4 データ構造 node とポインタ p

まず、探索の操作を行うプログラムを考える。図 5 は探索を行う関数 search のプログラムで、探索する B 木の根のノードへのポインタ p と探索するキーの値 keyvalue を受け取り、B 木内にキーが見つかった場合は TRUE を、見つからなかった場合は FALSE を返す。

```
function search( p, keyvalue )
// p: 根のノードへのポインタ, keyvalue: 探索するキーの値
while( p が  )
    k ← 1
    while( k が p->n_key 以下である かつ  )
        k ← k+1
    endwhile
    if ( k が p->n_key 以下である かつ p->key[k] が keyvalue に等しい )
        return TRUE // キーが見つかった
    endif
    p ← 
endwhile
return FALSE // キーが見つからない
endfunction
```

図 5 関数 search のプログラム

次に、挿入の操作を行うプログラムを考える。ただし、ここではプログラム全体ではなく、〔挿入〕(1)の場合に使用する関数 `insertitem` と、〔挿入〕(2)の場合に使用する関数 `split` について考えるものとする。

図6に関数 `insertitem` のプログラムを示す。関数 `insertitem` は、対象ノードに新たにキーとポインタを格納する場所がある場合に、挿入したいキーとポインタをノードの `k` 番目の要素として格納する。したがって、ノードの `k` 番目以降のキーとポインタを一つずつ後ろにずらし、引数で渡される挿入するキーの値 `keyvalue` とポインタの値 `newp` をノードの `k` 番目の要素に代入する。

```
function insertitem( p, k, keyvalue, newp )
// p: ノードへのポインタ, k: 挿入場所の配列の添字
// keyvalue: 挿入するキーの値, newp: 挿入するポインタの値
  for ( i を p->n_key から k まで1ずつ減らす )
    p->key[ エ ] ← p->key[ オ ]
    p->branch[ エ ] ← p->branch[ オ ]
  endfor
  p->key[k] ← keyvalue
  p->branch[k] ← newp
  p->n_key ← カ
endfunction
```

図6 関数 `insertitem` のプログラム

図7に関数 `split` のプログラムを示す。関数 `split` は、ポインタ `p` が指すノードに新しくキーを挿入したいが、対象ノードには既に $2M$ 個のキーが格納されており、キーを新たに格納する場所がない場合に、分割を行う関数である。このとき、元々の対象ノードに入っていたキーと新たに挿入するキーを合わせた全部で $2M+1$ 個のキーを昇順で並べたとき、新たに挿入するキーは `k` 番目のキーとなっている状況で呼び出される。関数 `split` は、 $2M+1$ 個のキーを、 $M+1$ 個と M 個の二つの組に分け、前半の $M+1$ 個のキーをポインタ `p` が指すノードに格納し、後半の M 個のキーを新しく生成するポインタ `q` が指すノードに格納する。また、ノード内のポインタもそれに合わせて適切に格納する。そして、親ノードへ昇進するキーの値を大域変数の `parentkeyvalue` に設定し、新たに作成したノードへのポインタ `q` を返り値とする。

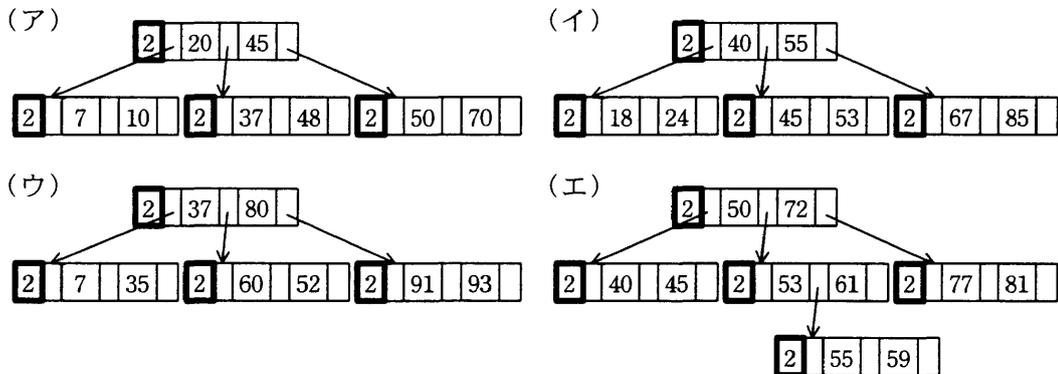
```

function split( p, k, keyvalue, newp )
// p: ノードへのポインタ, k: 昇順でキーを並べたときの挿入するキーの順番
// keyvalue: 挿入するキーの値, newp: 挿入するポインタの値
  if ( k が M+1 以下である )
    m ← M
  else
    m ← M+1
  endif
 新しくノードを作成し, 作成したノードへのポインタを q に代入
  for ( j を m+1 から 2*M まで 1 ずつ増やす ) // 後半部の移動
    q->key[ キ ] ← p->key[j]
    q->branch[ キ ] ← p->branch[j]
  endfor
  q->n_key ← ク
  p->n_key ← ケ
  if ( k が M+1 以下である )
    insertitem( p, k, keyvalue, newp )
  else
    insertitem( q, k-m, keyvalue, newp )
  endif
  parentkeyvalue ← コ // 昇進するキーの値を設定
  q->branch[0] ← p->branch[p->n_key]
  p->n_key ← サ
  return q // 新しく作成したノードへのポインタを返す
endfunction

```

図7 関数 split のプログラム

設問1 次の(ア)～(エ)の図の中でB木として正しいものはどれか, 記号で答えよ。



設問2 M=1である図8のB木に35, 40を順に挿入した後のB木はどうかを示せ。
必要ならば, ノードを追加せよ。

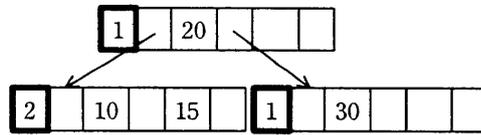


図8 挿入前のB木

設問3 本文中のプログラムに関して, (1), (2)に答えよ。

- (1) 図5のプログラム中の ~ に入れる適切な字句を答えよ。
- (2) 図6, 7のプログラム中の ~ に入れる適切な字句を答えよ。

設問4 木全体のキーの個数が最小となる場合は, 根に1個, 根以外のすべてのノードにキーがそれぞれM個入っている場合である。この場合のB木中のキーの個数の評価に関して, (1)~(3)に答えよ。

- (1) 図9に示すように, 同じ深さごとにノード数とキーの数を考える。図9中の ~ に入れる適切な式を答えよ。

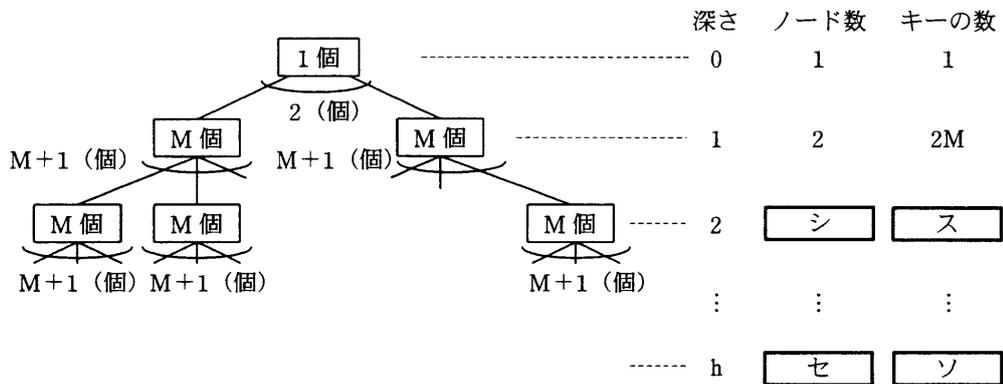


図9 ノードの数とキーの数

- (2) 深さhまでのキーの総数を答えよ。
- (3) 全部のキーの個数を100万個, Mを255とするとB木の深さは, 最大で幾らになるか答えよ。

〔メモ用紙〕

〔メモ用紙〕

〔メモ用紙〕

6. 途中で退室する場合には、手を挙げて監督員に合図し、答案用紙が回収されてから静かに退室してください。

退室可能時間	16:10 ~ 16:20
--------	---------------

7. 問題に関する質問にはお答えできません。文意どおり解釈してください。
8. 問題冊子の余白などは、適宜利用して構いません。
9. 電卓は、使用できません。
10. 試験終了後、この問題冊子は持ち帰ることができます。
11. 答案用紙は、白紙であっても提出してください。
12. 試験時間中にトイレへ行きたくなったり、気分が悪くなったりした場合は、手を挙げて監督員に合図してください。

試験問題に記載されている会社名又は製品名は、それぞれ各社の商標又は登録商標です。
なお、試験問題では、® 及び ™ を明記していません。