

問 5 ヒープに関する次の記述を読んで、設問 1～5 に答えよ。

次の条件を満たす二分木を考える。

条件 1: 根から最も遠い葉の深さを d としたとき、葉の深さがすべて d であるか、 d と $d-1$ の 2 種類に限られていて、深さ d の葉が左側から詰められている。

条件 1 を満たす二分木が、次の条件 2 を満たすとき、これをヒープと呼ぶ。

条件 2: 葉を除くすべてのノードの値が、その子のノードの値より大きいか、又は等しい。

ヒープの例を図 1 に示す。各ノードの左上の数値はノードの番号を、ノード内の数値はノードの値を、それぞれ示す。ノードの番号は、次の規則で付ける。

- ・根には番号 1 を付ける。
 - ・番号 k のノードの左の子には番号 $2k$ 、右の子には番号 $2k+1$ を付ける。
- 以下、番号 n のノードを、ノードという。

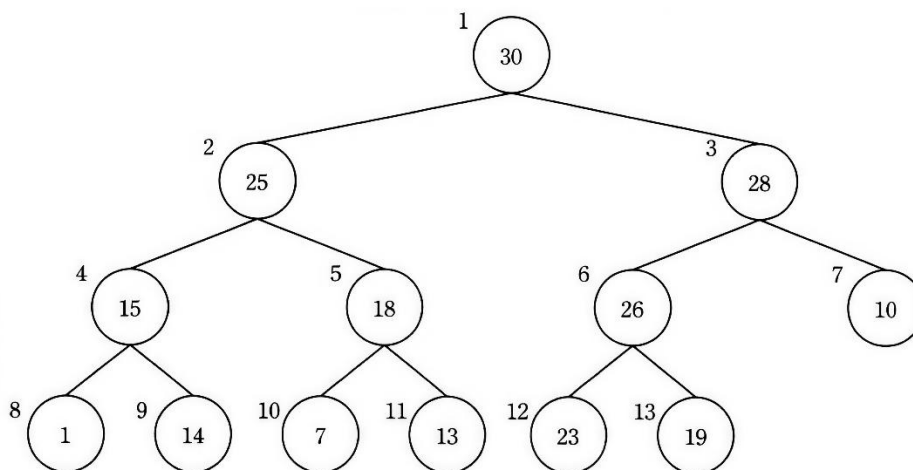


図 1 ヒープの例

[heap_correct のアルゴリズム]

heap_correct は、条件 1 を満たす二分木の中のノード i が条件 2 を満たさない場合に、親ノードの値が子ノードの値より大きいか等しくなるように親子の値を入れ替える操作を、ノード i を根とする部分木が条件 2 を満たすようになるまで、ノード i から葉へ向かって繰り返す処理である。このとき、ノード i の子ノードを根とする部分木は、あらかじめ条件 2 を満たしているものとする。 heap_correct の実行例を図 2, 3 に示す。

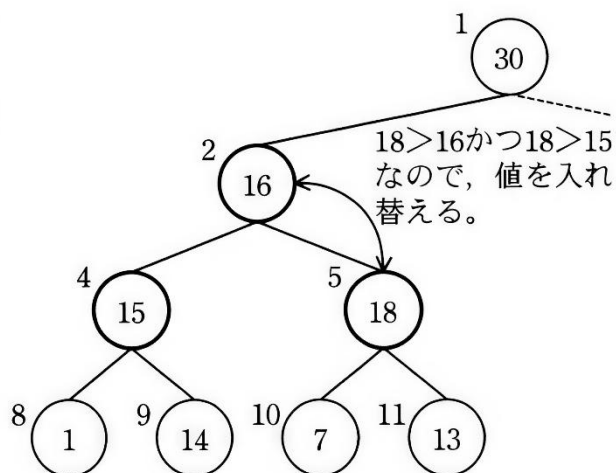


図2 heap_correct 実行例 1

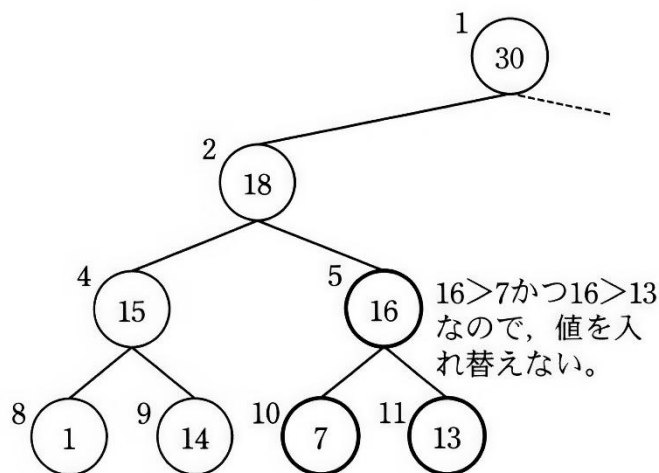


図3 heap_correct 実行例 2

図 2, 3 は, ノード 2 に対して heap_correct を実行した例である。ノード 2 の子であるノード 4 及びノード 5 を根とする部分木は, 条件 2 を満たしている。ノード 2 は, ノード 5 より小さい値をもつので, 条件 2 を満たしていない。heap_correct は, ノード 2 が条件 2 を満たすように, 図 2 の矢印で示すように, ノード 2 の値とノード 5 の値を入れ替える。その結果を図 3 に示す。

図 3 では, ノード 2 と値を入れ替えたノード 5 が, heap_correct の次の処理対象となる。ノード 5 の値は, その左右いずれの子ノードの値よりも大きく, 条件 2 を満たしているので, heap_correct は親子のノードの値を入れ替えずに処理を終える。

図 4 に heap_correct のプログラムを示す。二分木を表すデータ構造には, 十分に大きい 1 次元配列 A を用いる。配列 A の添え字は 1 から始まり, 要素 A[i] はノード i の値を表す。表に, heap_correct で使用する変数とサブルーチンを示す。

表 heap_correct で使用する変数, サブルーチン

変数／サブルーチン	変数の意味／サブルーチンの機能
LENGTH	二分木のノード数を格納する大域変数
exchange(A, x, y)	配列 A の x 番目と y 番目の要素の値を入れ替える。

```
heap_correct(A, i)
  largest ← i
  L ← i * 2
  R ← i * 2 + 1
  if(L が LENGTH 以下である かつ A[L]が  )
    largest ← L
  endif
  if(R が LENGTH 以下である かつ  )
    largest ← R
  endif
  if(largest が i と等しくない)
    exchange(A, i, largest)
    heap_correct(A, largest)
  endif
```

図 4 heap_correct のプログラム

〔heap_make のアルゴリズム〕

heap_correct を用いて、条件 1 を満たす任意の二分木をヒープにするプログラム heap_make を作成する。heap_make は、根からの深さが最も深い葉の中で右端にあるノードの親ノードを始点に、その深さにあるすべてのノードに対し、右から左へ順番に heap_correct を実行する。左端のノードまで処理し終わると、更にその親の深さに移動し、右端から左端へと順番に heap_correct を実行する。これを繰り返して、最後に根に対して heap_correct を実行する。図 5 は heap_make のプログラムである。floor(n)は、数値 n を超えない最大の整数値を表す。

```
heap_make(A)
  n ← floor(LENGTH / 2)
  for(i を n から 1 まで 1 ずつ減らす)
    heap_correct(, )
  endfor
```

図 5 heap_make のプログラム

図 6 の二分木を使って、heap_make の実行例を示す。図 6 の二分木を表す配列を B とする。heap_make(B) を実行すると、LENGTH=15 であるから、ノード 7 から始めて根に至るまで、ノードの番号の降順に heap_correct を実行する。根まで処理を行った結果、根の値は〔オ〕になる。途中で、heap_correct がノード i を処理する際に、ノード i とその左右の子ノードの値を比較し、子の中にノード i の値〔カ〕値をもつものがあれば、heap_correct は再帰的に呼び出される。したがって、図 6 でノード 7 からノード 4 まで処理を終えた時点で、heap_correct は再帰呼出しを含めて〔キ〕回実行される。

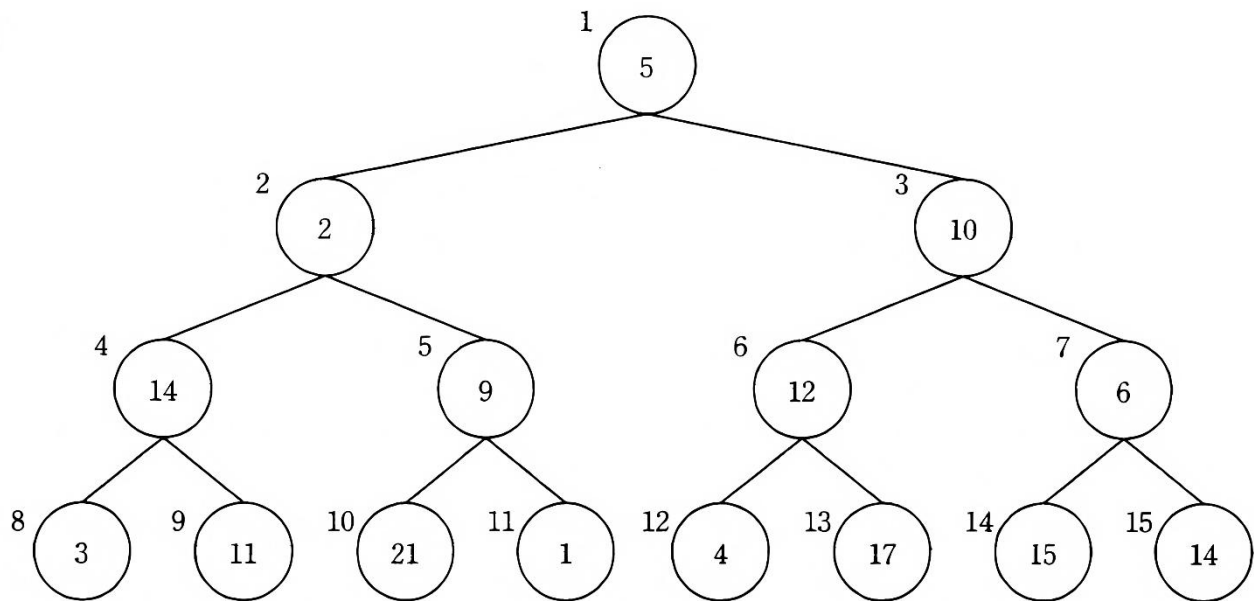


図 6 heap_make の実行例で対象となる二分木 B

設問 1 図 4 中の [ア], [イ] に入れる適切な字句を答えよ。

設問 2 図 5 中の [ウ], [エ] に入れる適切な字句を答えよ。

設問 3 本文中の [オ], [キ] に入れる適切な数値を答えよ。

設問 4 本文中の [カ] に入れる適切な字句を解答群の中から選び、記号で答えよ。

解答群

アと等しい イより大きい ウより小さい

設問 5 heap_correct 及び heap_make の計算量について考える。処理対象となる配列 A の表す二分木のノード数を N とするとき、次の①, ②の計算量として適切な式を、それぞれ解答群の中から選び、記号で答えよ。処理対象となる二分木は、条件 1 を満たしているものとする。

① 二分木の根に対して heap_correct を実行した場合の、heap_correct の計算量。このとき、二分木の根の子ノードを根とする部分木は、条件 2 を満たしているものとする。

② 配列 A の要素が降順にソートされている場合の heap_make(A) の計算量。

解答群

ア $O(1)$ イ $O(\log N)$ ウ $O(N)$ エ $O(N \log N)$ オ $O(N^2)$