ることがわかる．また，各節点の値は，その節点が表す再帰を終了するのに必要な時間計算量を表している．したがって，再帰アルゴリズムの時間計算量は，再帰木のすべての節点が表す値の和に等しいことになる．図 3.7 (d)の再帰木の場合は，再帰木の高さは n であり，各レベルの節点数は 1，各節点の値は c であるので，この再帰木で表される再帰アルゴリズムの時間計算量は $c \times n = O(n)$ である．

再帰木を用いた再帰アルゴリズムの計算量の求め方(2)　再帰木をさらに理解するために，配列 A[0], A[1], ..., A[n-1] の和を求める別の再帰アルゴリズムについて考える．アルゴリズム 3.4 は，配列 A[0], A[1], ..., A[n-1] の和が，その配列の半分ずつの和を足したものに等しいという以下の式に基づいた再帰アルゴリズムである．

$$\sum_{i=0}^{n-1} \texttt{A[i]} = \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} \texttt{A[i]} + \sum_{i=\lfloor \frac{n-1}{2} \rfloor + 1}^{n-1} \texttt{A[i]}$$

アルゴリズム 3.4　和の計算を行う再帰アルゴリズム(その 2)

```
recursive_sum2(A[0],A[1],...,A[n-1]) {
  if (入力の引数がA[k]という1つの配列要素のみである) { return A[k]; }
  else {
    配列A を半分ずつの以下の 2 つの配列に分割する;
      A1={A[0],A[1],...,A[(n-1)/2]}
      A2={A[(n-1)/2+1],A[(n-1)/2+2],...,A[n-1]}
    x=recursive_sum2(A1);
    y=recursive_sum2(A2);
    return x+y;
  }
}
```

なお，簡単のため，アルゴリズム 3.4 では n は 2 のべき乗の数であると仮定している．

このアルゴリズム 3.4 の計算量を考える．n 個の和を求めるのに必要なアルゴリズムの時間計算量を $T(n)$ とおくと，このアルゴリズム 3.4 は，定数個の演算(if 文や加算など)と時間計算量が $T\left(\frac{n}{2}\right)$ となる 2 つの再帰呼び出しから構成されている．したがって，定数個の演算の時間計算量を定数 c とおくと，$T(n)$ について以下の式が成り立つ．

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + c & (n \geq 2 \text{ の場合}) \\ c & (n = 1 \text{ の場合}) \end{cases}$$

上記の漸化式に基づいた再帰木を図 3.8 に示す．まず，式に基づき図(a)の木が得られ，ここから漸化式を 1 回展開することにより図(b)の木が得られる．そして，この木を再帰が終了するまで展開することにより図(c)の再帰木が得られる．

ここで，この再帰木の高さを h とおいて，再帰途中の入力サイズに着目して h の値を求める．再帰を始める前の入力サイズが n であり，再帰木のレベルが 1 つ増えるごとに入力サイズは $\frac{1}{2}$ になる．したがって，再帰木のレベル k における入力サイズは

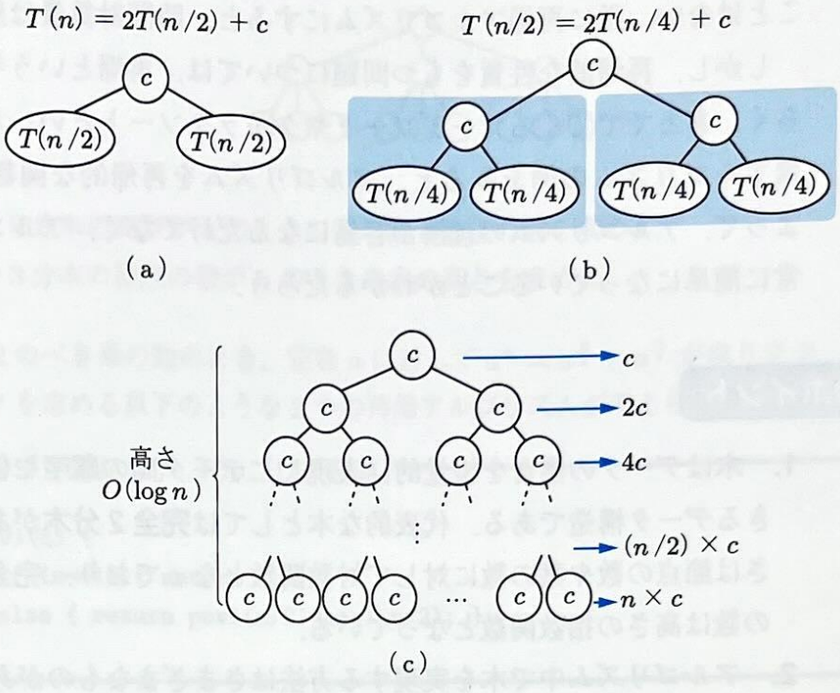

図 3.8　アルゴリズム 3.4 の再帰木

$\left(\frac{1}{2}\right)^k n$ となる．この入力サイズが 1 になるまで再帰は繰り返されるので，再帰木の高さ h について，以下の式が成り立つ．

$$\left(\frac{1}{2}\right)^{h-1} n \leq 1$$

この式は，アルゴリズム 1.3 の計算量の場合と同じように解くことができ，$h = O(\log n)$ となる．また，この再帰木は完全 2 分木であり，葉の数は入力サイズ n に等しいので，性質 3.2 からも $h = O(\log n)$ であることがわかる．

この h を用いて再帰木により表される時間計算量を求める．図 3.8 (c) に表されているように，再帰木の各レベルに含まれる節点の値の和は c, $2c$, $4c$, …, $\frac{n}{2}c$, nc という初項 c，項比 2 の等比数列となっている．また，再帰木の高さは性質 3.2 より $1+\log_2 n$ であり，この数列の項数は木の高さに等しいので，$1 + \log_2 n$ である．よって，初項 a，公比 r の等比数列の和の公式 $\sum_{i=0}^{n-1} a \cdot r^i = a \cdot \frac{1-r^n}{1-r}$ を用いて，再帰木のすべての節点の値の和であるアルゴリズム 3.4 の時間計算量を求めると，以下の式により $O(n)$ であることがわかる．

$$\sum_{i=0}^{\log_2 n} c \cdot 2^i = c \cdot \frac{1 - 2^{1+\log_2 n}}{1-2} = c(2n-1) = O(n)$$

再帰に関する注意　最後に再帰に関する注意を述べる．本節で述べた再帰アルゴリズムの例は，再帰の概念や再帰木の説明のためのものであり，一般には整数の和のような再帰的な性質がない問題に対して再帰アルゴリズムを用いて解く必要はない．アルゴリズム 3.2 の時間計算量とアルゴリズム 3.3 やアルゴリズム 3.4 の時間計算量を比較するとわかるように，一般に再帰を用いてもアルゴリズムの時間計算量が良くなる