アルゴリズムとデータ構造

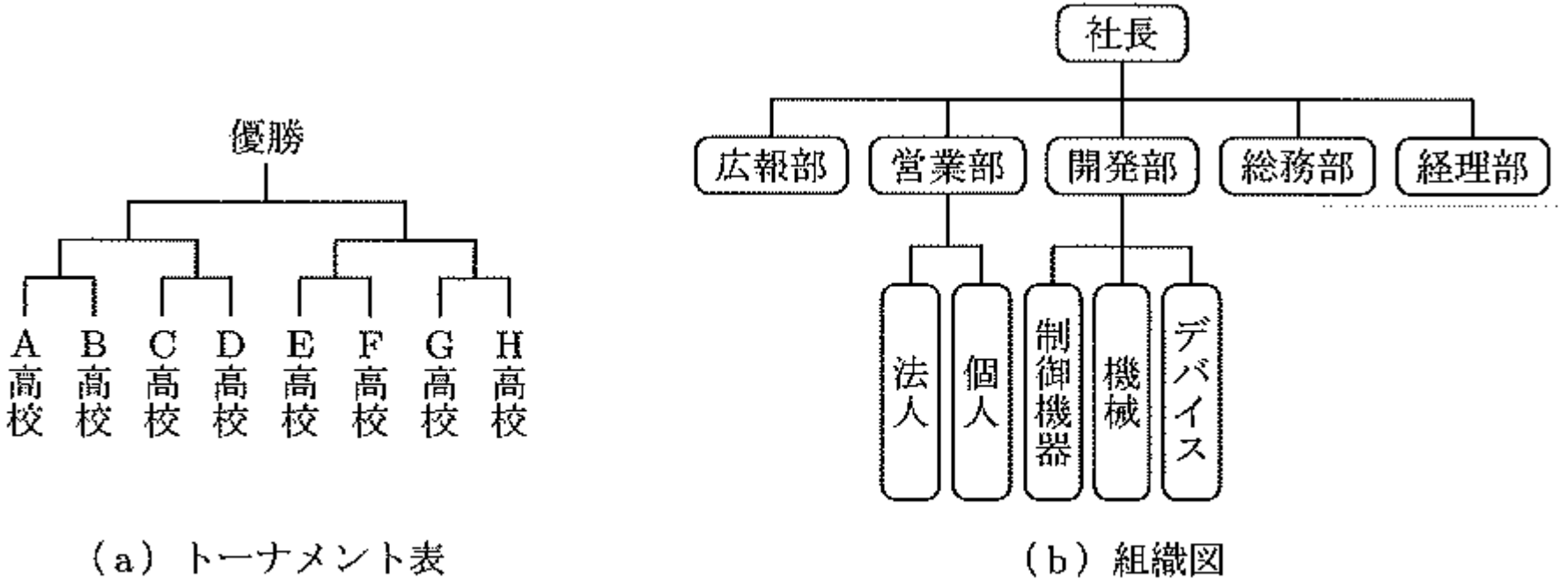
第4週目

担当　情報システム部門　徳光政弘
2025年5月7日

今日の内容

- 完全木
- 再帰
- 二分探索木

木の概念(復習)



（a）トーナメント表　　（b）組織図

図 3.1　一般生活における木

木の用語

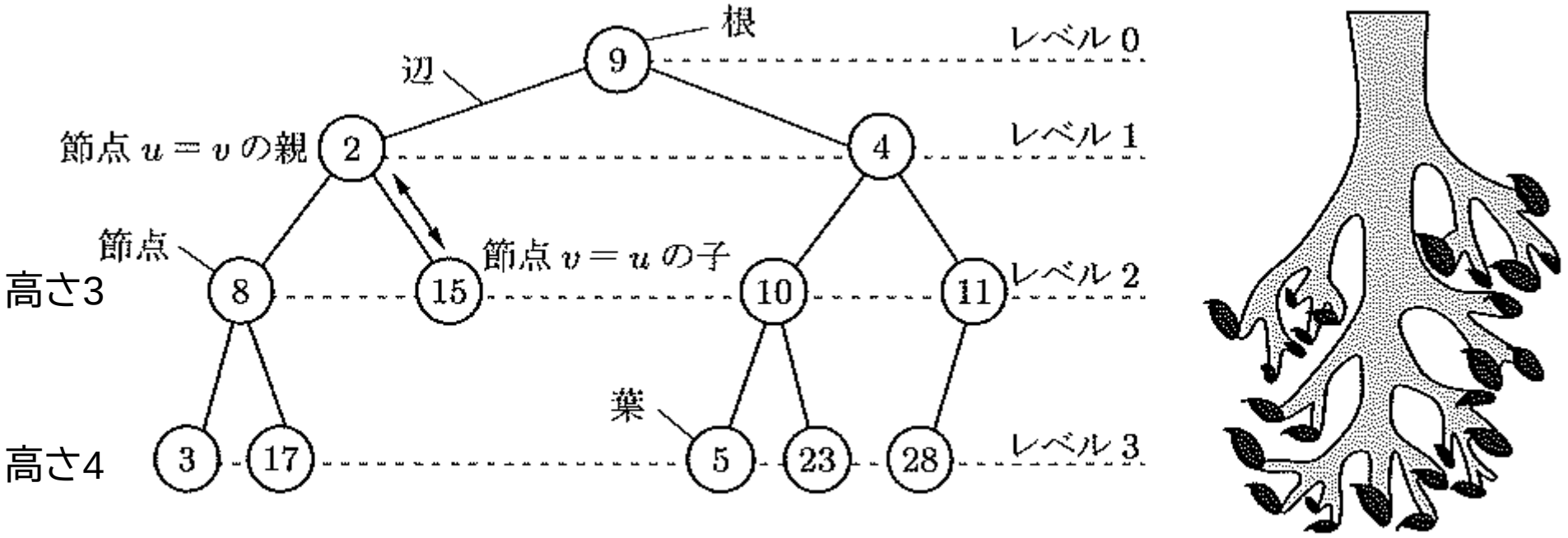


図 3.2　アルゴリズムにおける木と実際の木

木の高さは「レベル＋1」と定義する。レベル3の木の高さは4となる。

本や資料によって用語の定義が微妙に異なることもあるので注意

完全2分木

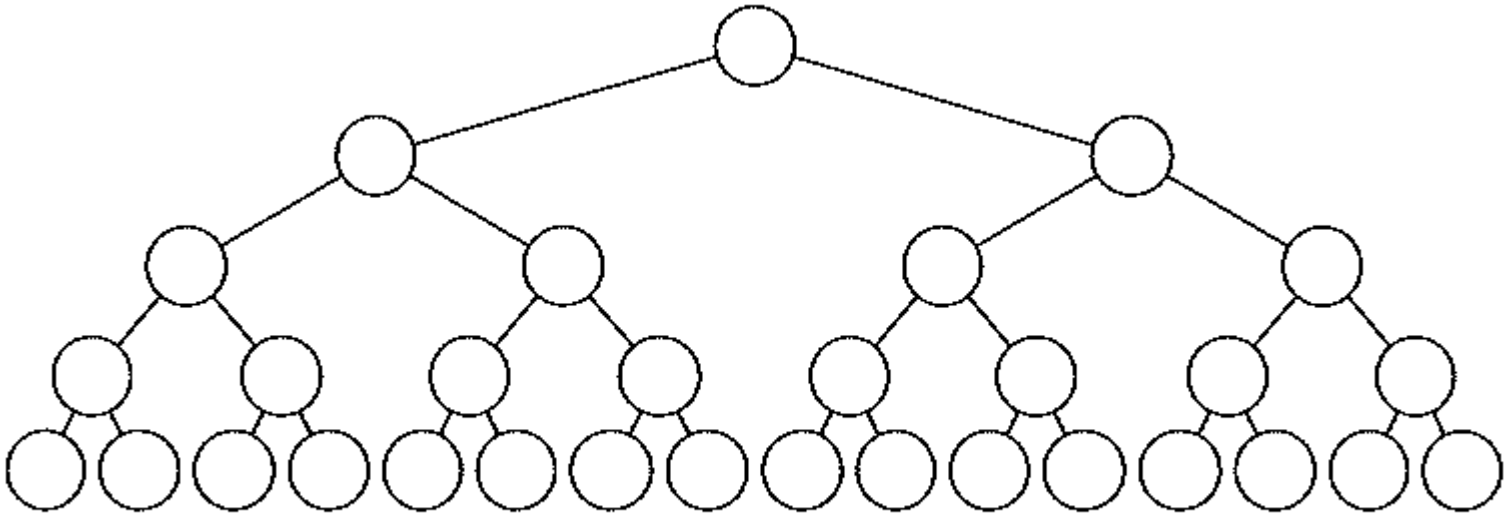


図 3.3　完全 2 分木

木の性質

●性質 3.1

完全 2 分木の葉の数は，その完全 2 分木の高さを h とすると，$2^{h-1} = O(2^h)$ である．

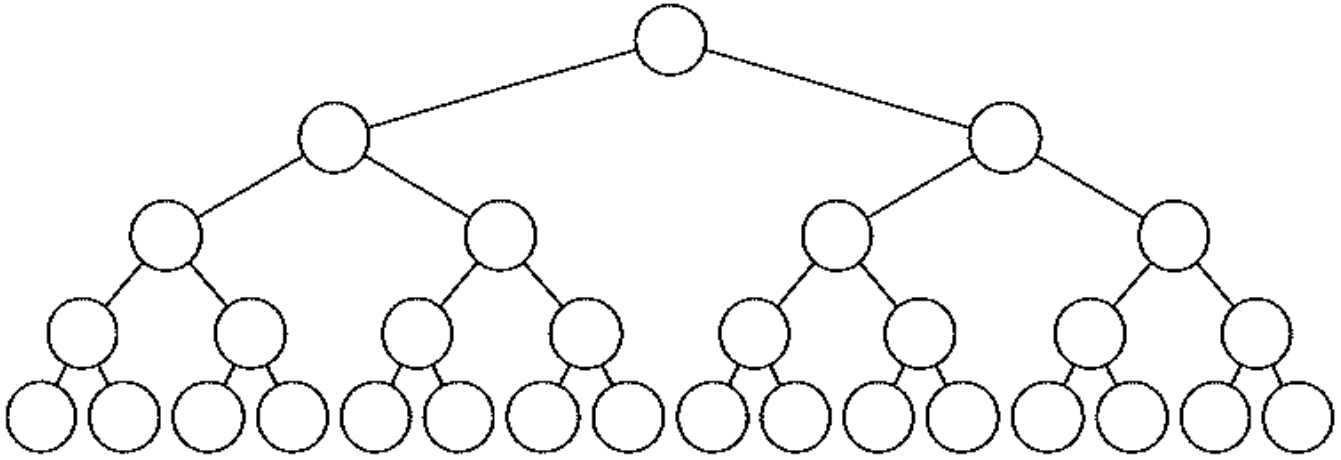


図 3.3　完全 2 分木

木の性質

●性質 3.2

完全2分木の高さは，その完全2分木の葉の数を m とすると，$1+\log_2 m = O(\log m)$ である．

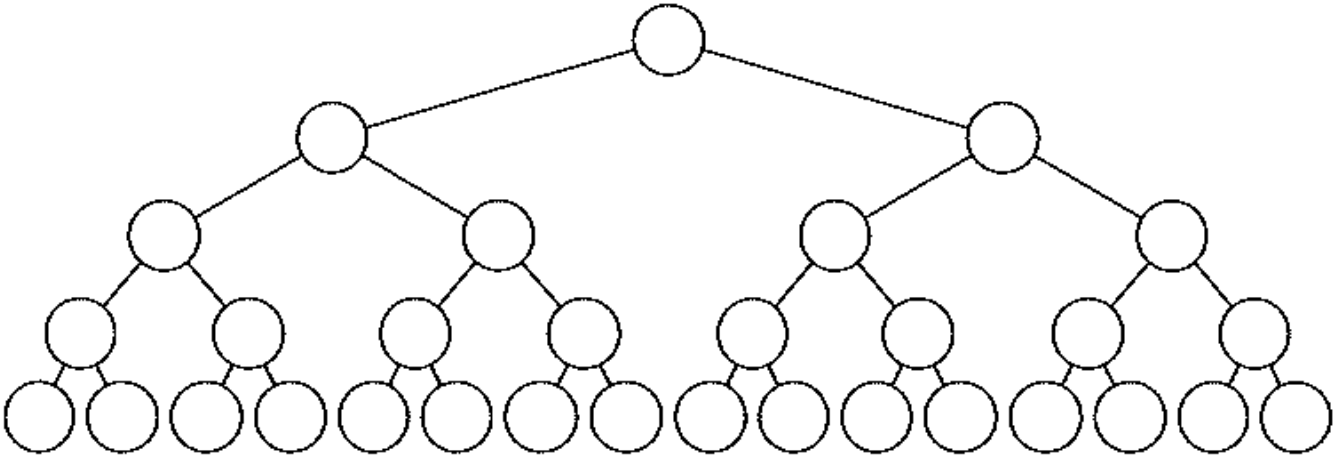


図 3.3　完全2分木

節点の数

$$1 + 2 + 4 + 8 + \cdots + 2^{h-1} = \sum_{k=0}^{h-1} 2^k$$

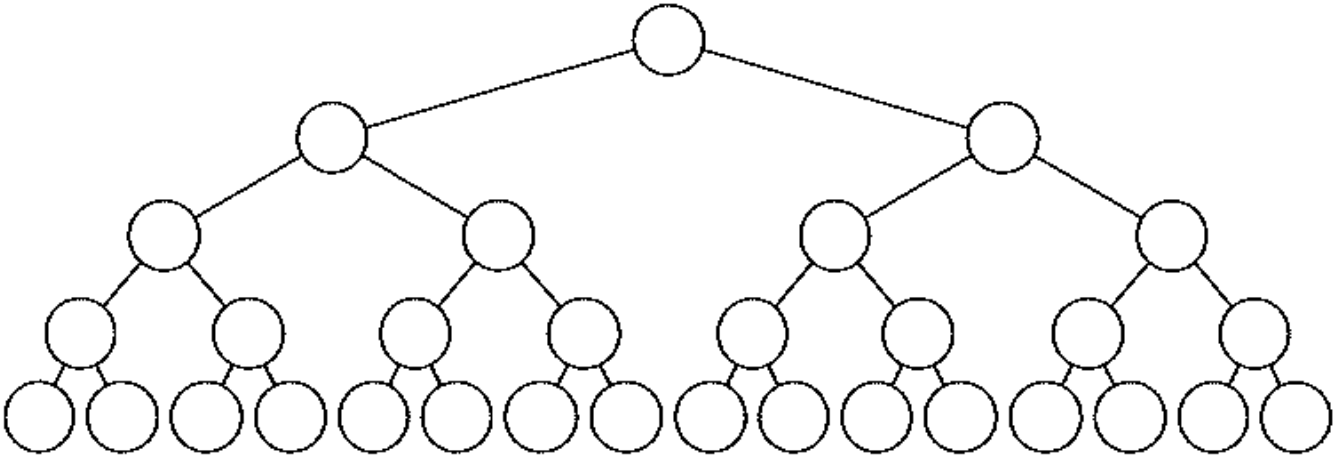


図 3.3　完全 2 分木

節点の数

$$1+2+4+8+\cdots+2^{h-1}=\sum_{k=0}^{h-1}2^k$$

数列の和の公式 $\sum_{i=0}^{n-1} a\cdot r^i = a\cdot\frac{1-r^n}{1-r}$

$$\sum_{k=0}^{h-1}2^k = 1\cdot\frac{1-2^h}{1-2}=2^h-1$$

節点の数

$$\sum_{k=0}^{h-1} 2^k = 1 \cdot \frac{1-2^h}{1-2} = 2^h - 1$$

●性質 3.3

完全 2 分木の節点の数は，その完全 2 分木の高さを h とすると，$2^h - 1 = O(2^h)$ である．

節点の数

$$\sum_{k=0}^{h-1} 2^k = 1 \cdot \frac{1-2^h}{1-2} = 2^h - 1$$

●性質 3.4

完全2分木の高さは, その完全2分木の節点の数を n とすると, $\log_2(n+1) = O(\log n)$ である.

節点の数を対数を求めて言い換えているだけ

木の性質のまとめ

これらの木に関する性質をまとめると，“完全２分木の高さは，節点の数や葉の数に対して対数関数となっており，完全２分木の節点の数や葉の数は，高さの指数関数となっている”ということである．これはさまざまなアルゴリズムの計算量を考えていくうえで重要な性質なので，おぼえておくとよい．

木に置き換えてアルゴリズムとデータ構造を考える場合、対数や指数で計算量を示すものがあるため、これら基本的性質は重要である。

木の実現

- 配列
- ヒープ（後述、今日は扱わない）

木の実現

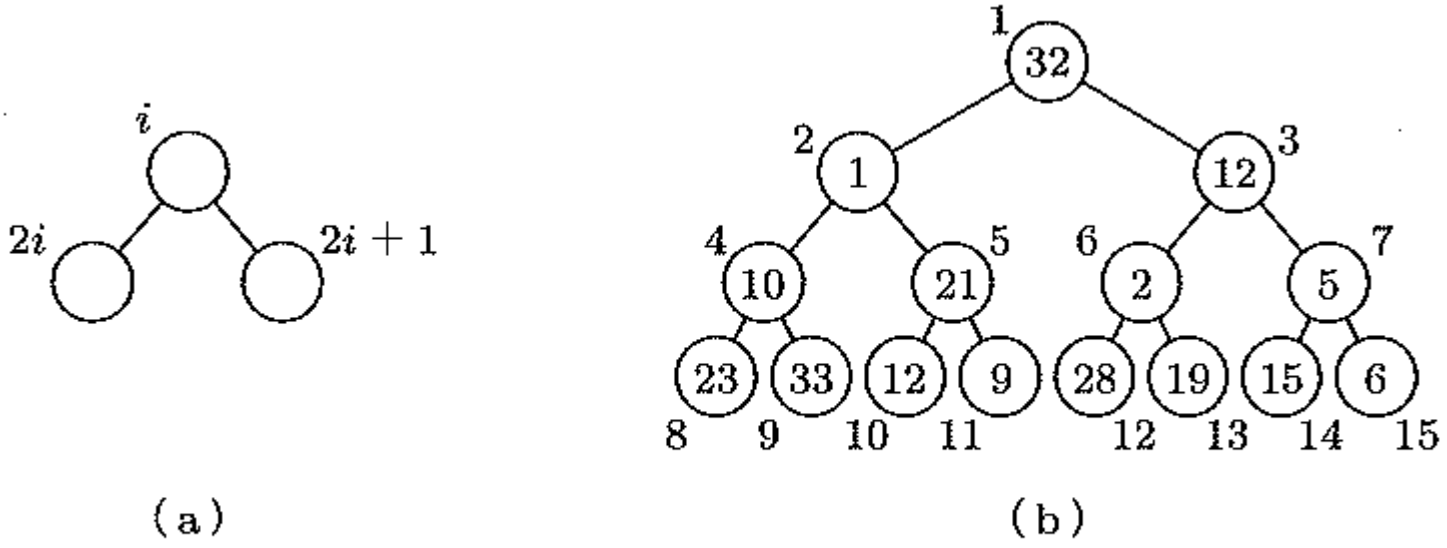


図 3.4　番号付けされた完全 2 分木とその番号付けの方針

木の実現

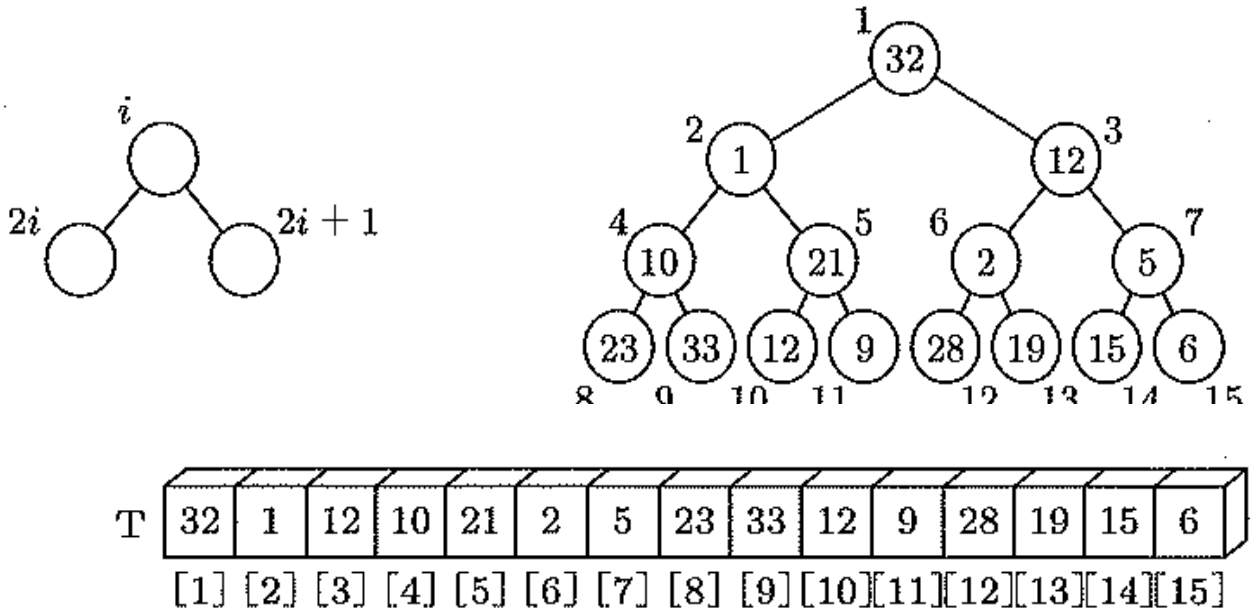


図 3.5　完全 2 分木を表す配列

左 2i
右 2i + 1

配列の位置で右・左の接点を表現している。

再帰

- 関数の中で関数自身を呼び出す手続き
- 再帰を含むアルゴリズムは再帰アルゴリズム

[問題 3.1]

ある細胞は，試験管中で 1 分経過すると分裂し，数が 2 倍になるが，分裂直後に全細胞のうち 1 つは死滅してしまう．最初に試験管に 10 個の細胞を入れたとき，細胞を入れてから n 分後の試験管中の細胞の数はいくつか．

細胞の増殖(再帰)

- nを時間経過(分)
- c(n) = 2 * c(n) - 1

[問題 3.1]

ある細胞は，試験管中で1分経過すると分裂し，数が2倍になるが，分裂直後に全細胞のうち1つは死滅してしまう．最初に試験管に10個の細胞を入れたとき，細胞を入れてからn分後の試験管中の細胞の数はいくつか．

細胞の増殖(再帰)

アルゴリズム 3.1

```
cell(n) {
  if (n==0) { return 10; }
  else { return 2*cell(n-1)-1; }
}
```

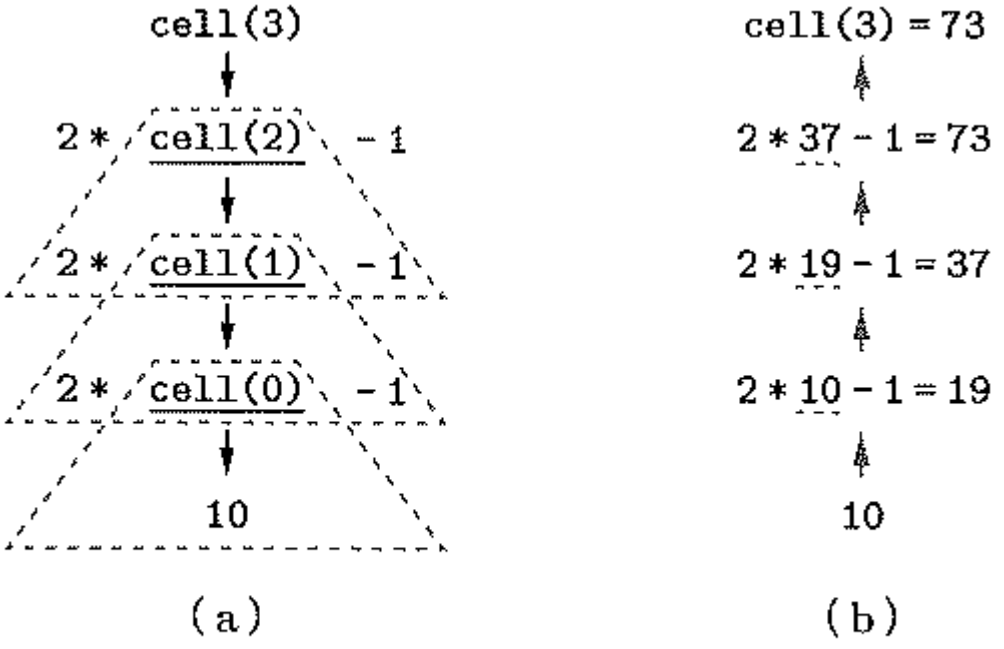
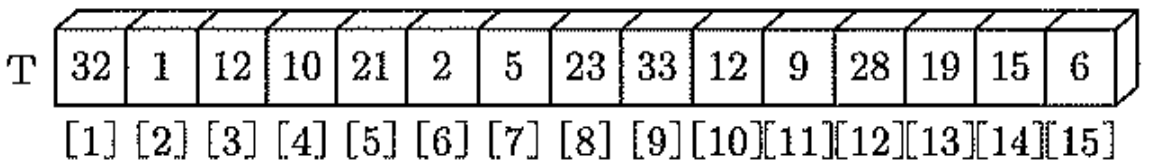


図 3.6　再帰の様子

再帰木

- 配列に数値が格納されている



アルゴリズム **3.2**　和の計算

```
sum=0;
for (i=0; i<n; i=i+1) { sum=sum+A[i]; }
sum を出力;
```

再帰木

- 再帰でも実装できる

アルゴリズム 3.3　和の計算を行う再帰アルゴリズム(その 1)

```
recursive_sum1(A[0], A[1], ..., A[n-1])  {
  if (入力の引数が A[0] のみである) { return A[0]; }
  else return recursive_sum1(A[0], A[1], ..., A[n-2])+A[n-1];
}
```

再帰木の計算量

- n個の数値の総和を考える
- T(n)を全体の計算量とする

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

再帰木の計算量(図で考える)

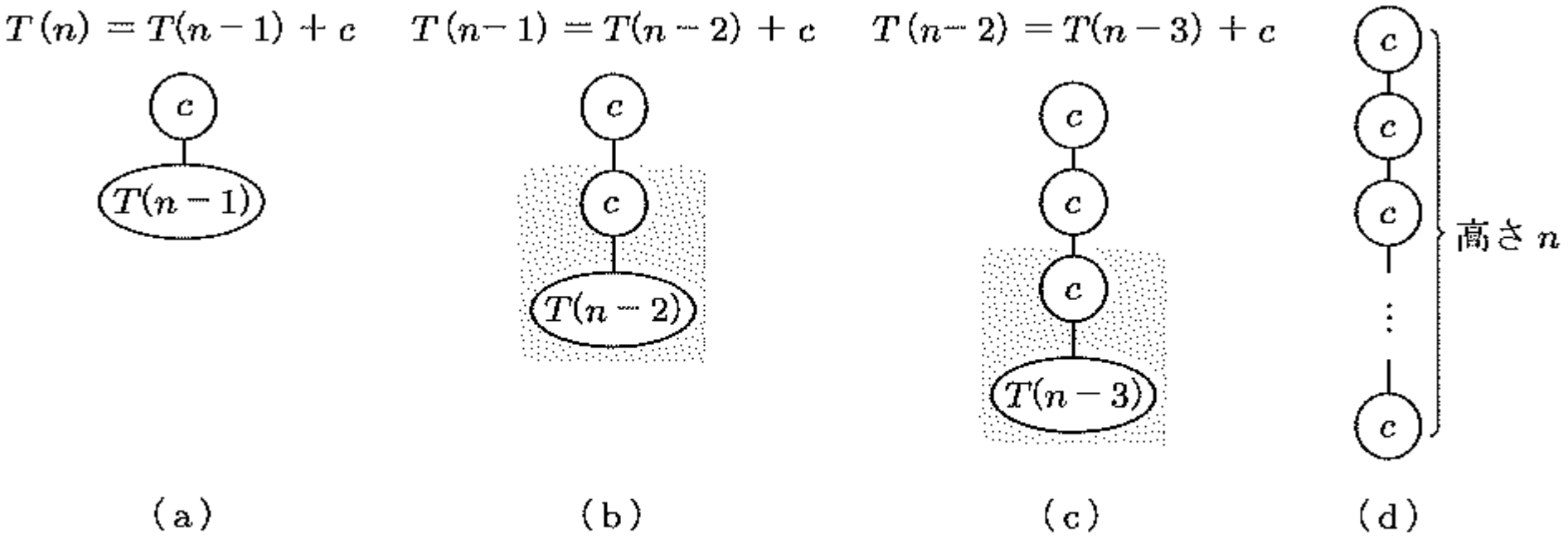


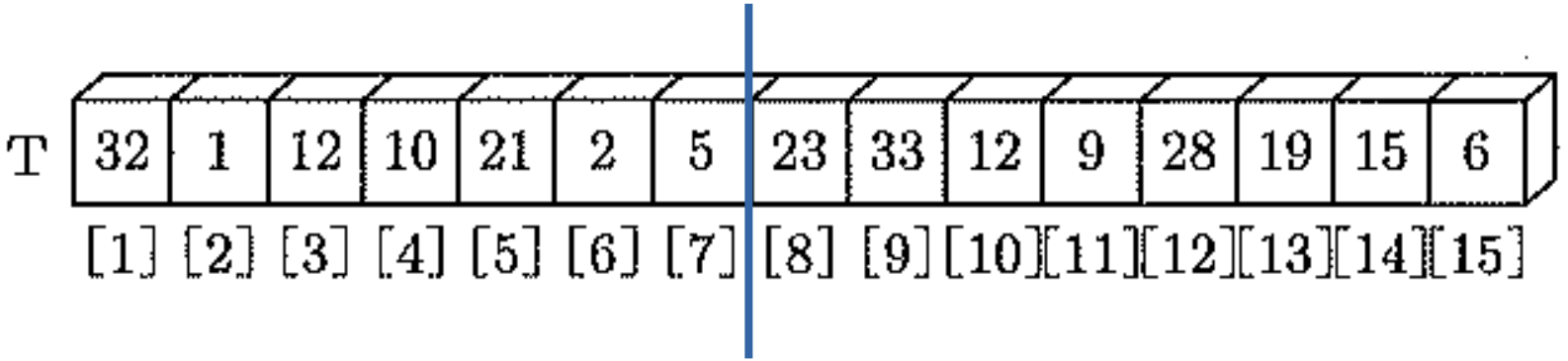
図 3.7　アルゴリズム 3.3 の再帰木

cは固定的な計算量

再帰木の計算量

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(分割)



分割線を中心に左・右に分けて再帰の計算量を考える

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

再帰木(分割)

アルゴリズム **3.4**　和の計算を行う再帰的なアルゴリズム(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;
  }
}
```

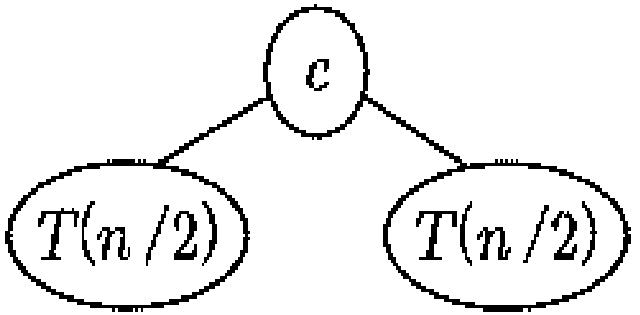
再帰木(分割)

- アルゴリズム全体の計算量をT(n)
- 定数の計算量をc
- 半分に分割するので(n/2)となる

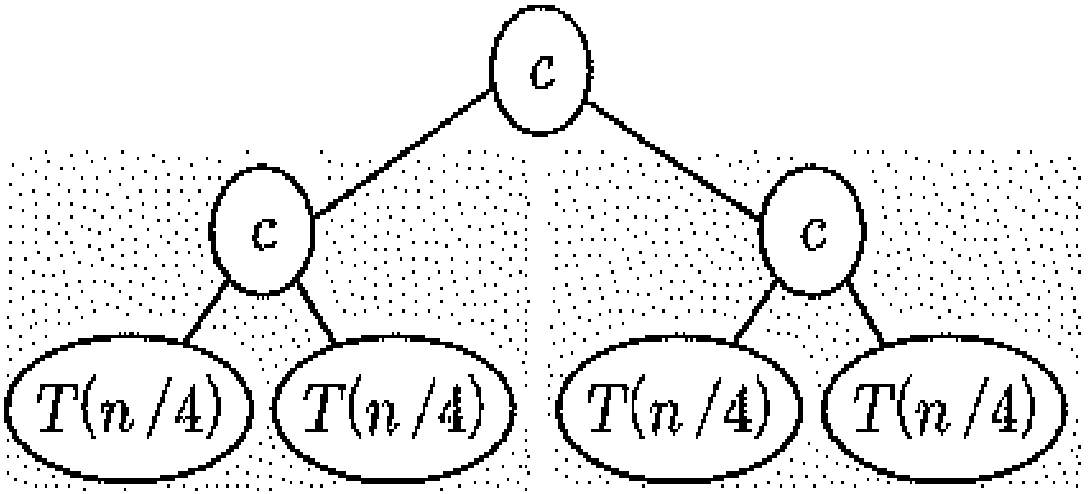
$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + c & (n \geq 2 \text{ の場合}) \\ c & (n = 1 \text{ の場合}) \end{cases}$$

再帰木(分割)

$T(n) = 2T(n/2) + c$



$T(n/2) = 2T(n/4) + c$



計算の分割を2回しているため、係数が2となっている。右の考えたも同じ、1/2ずつnが小さくなり、係数を2倍している。

再帰木(分割)

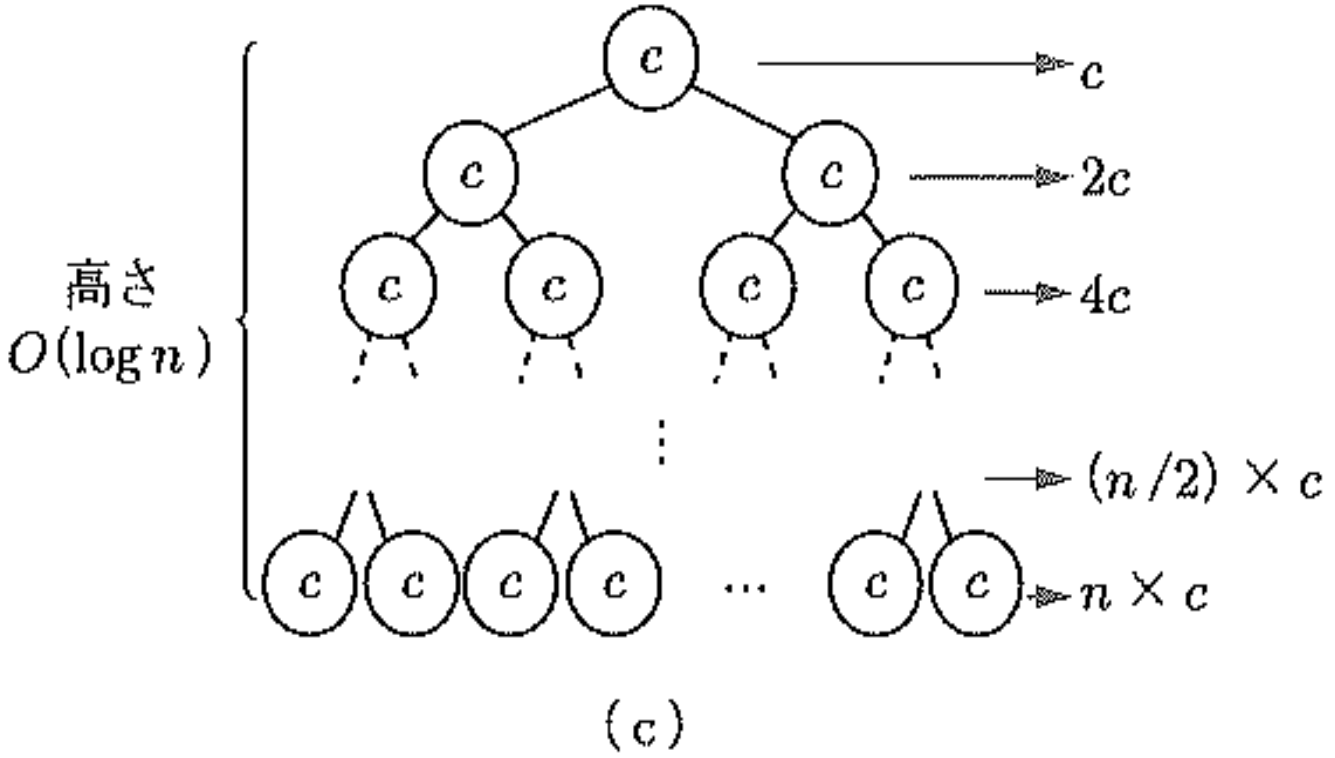


図 3.8　アルゴリズム 3.4 の再帰木

再帰木(分割)

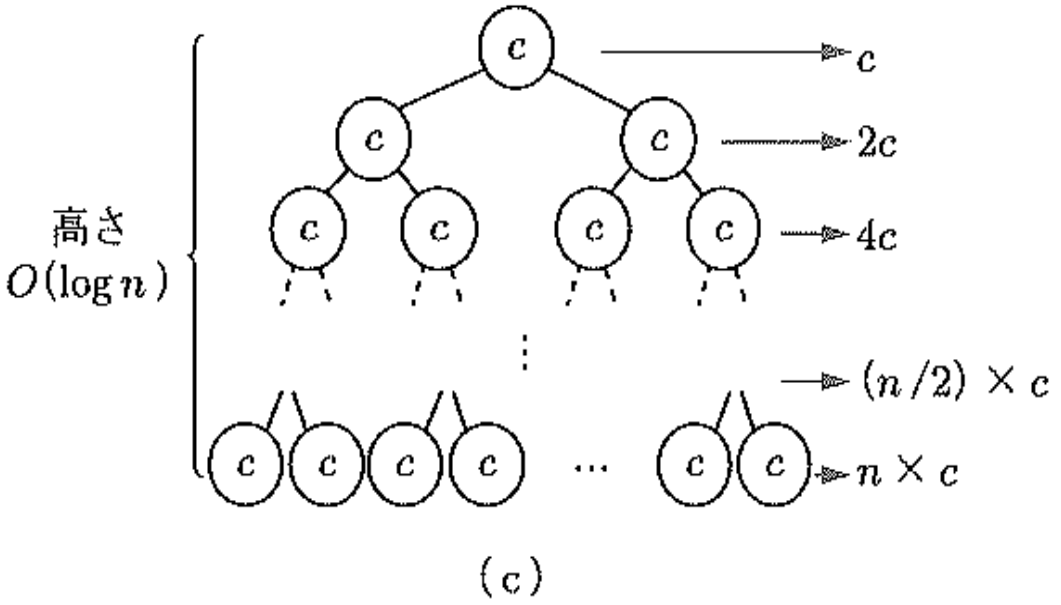


図 3.8　アルゴリズム 3.4 の再帰木

木の高さの関係が満たすべき式(分割して最後は1個という意味)

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

再帰木(分割)

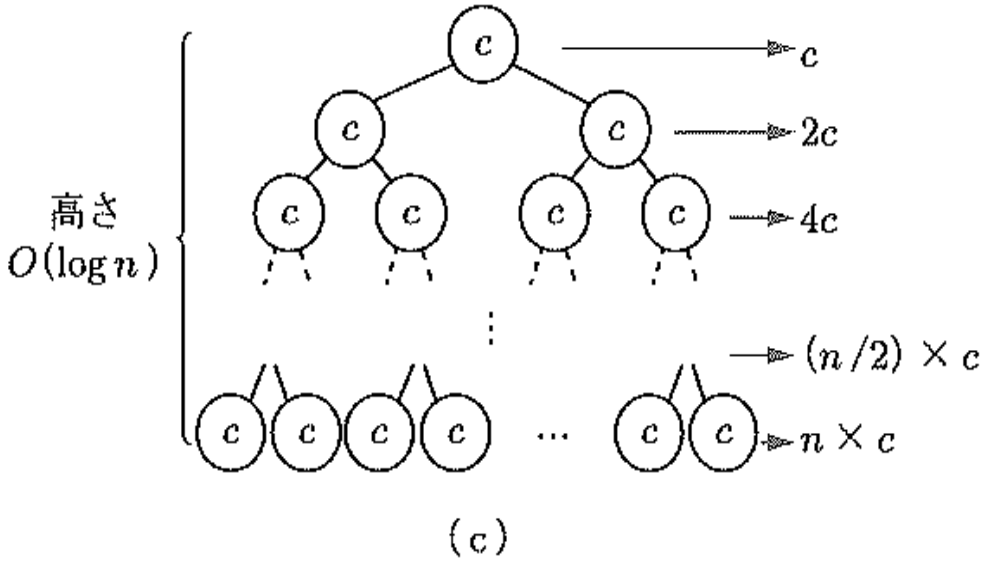


図 3.8　アルゴリズム 3.4 の再帰木

再帰木の各レベルに含まれる節点の値の和は $c,\ 2c,\ 4c,\ \ldots,\ \frac{n}{2}c,\ nc$

再帰木(分割)

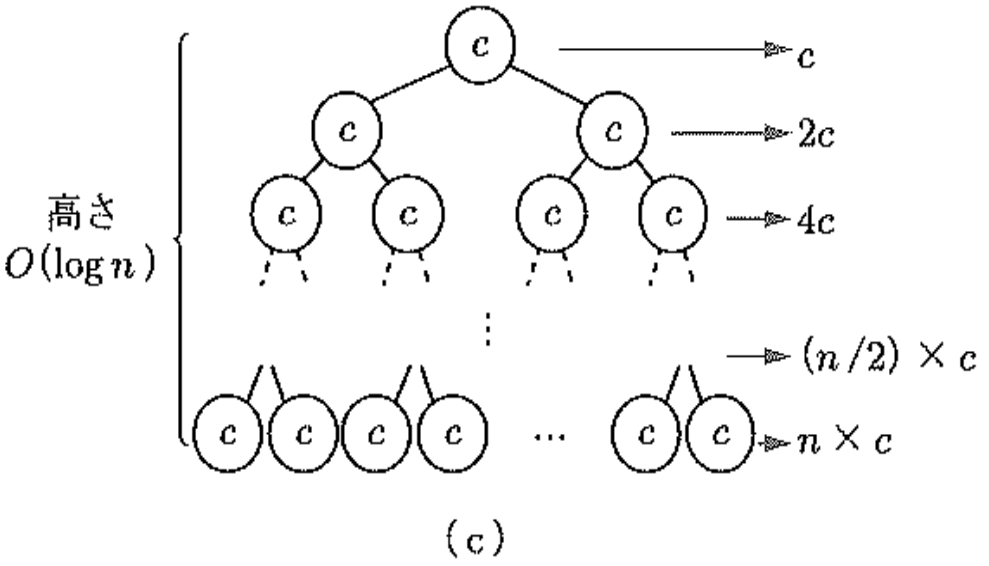


図 3.8　アルゴリズム 3.4 の再帰木

等比数列の和の公式 $\sum_{i=0}^{n-1} a \cdot r^i = a \cdot \frac{1-r^n}{1-r}$;

再帰木(分割)

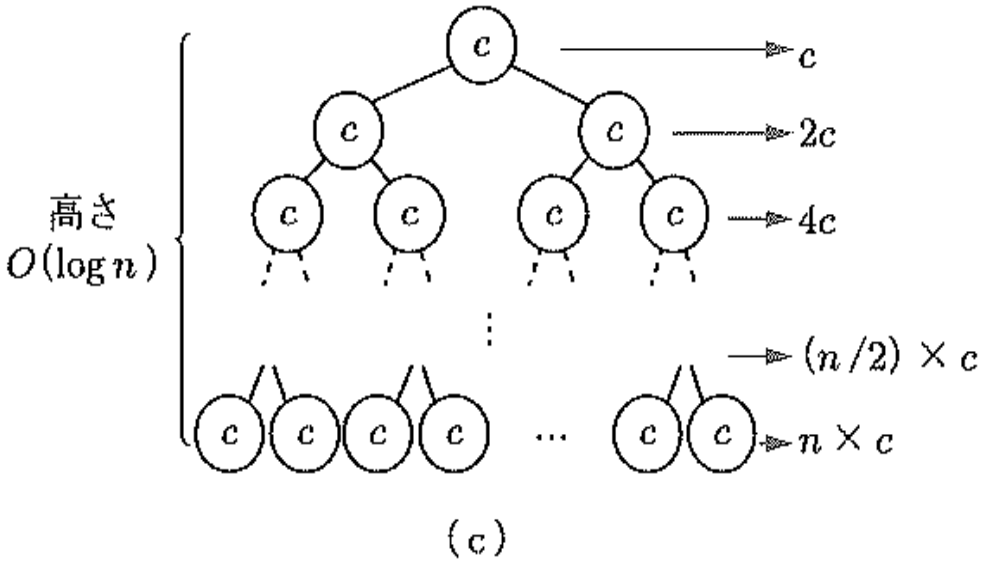


図 3.8　アルゴリズム 3.4 の再帰木

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

木の応用　2分探索木

- 教科書(解きながら～のp.第9章)

木の応用　平衡木等

- 教科書(解きながら～のp.第9章)
- 木のバランスを考慮している
- アルゴリズムの内容としては発展的なもののため省略(本当は大事)