

アルゴリズムとデータ構造

— 理論編 —

テキスト

山本真基

2017年4月

アルゴリズムとデータ構造の基礎知識及び基礎概念を学習する．アルゴリズムとは、「問題」を解くための機械的手順のことである．データ構造とは、(アルゴリズムが扱う) データを保持する形式のことである．ここでは、以下の目次にあるよう、探索・整列・走査に関する典型的なアルゴリズムとデータ構造を取り上げる．特に、アルゴリズムの正当性の証明と時間計算量の解析について解説する．

以降で使われる表記

- $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ をそれぞれ，自然数，整数，有理数，実数の集合とする．($\mathbb{Q}^+, \mathbb{R}^+$ をそれぞれ，正の有理数，正の実数，の集合とする．) また， $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ ， $\mathbb{Z}_n = \{0, 1, 2, \dots, n-1\}$ とする．更に，任意の $n \in \mathbb{N}$ について， $[n] = \{1, 2, \dots, n\}$ とする．
- 対数関数 \log, \ln について，底が 2 であるとき \log ，底が e であるとき \ln ，と表記する．（よって，任意の $x \in \mathbb{R}^+$ について $2^{\log x} = x$ ， $e^{\ln x} = x$ ．）

新出用語

本書で学習する新しい用語は，新出時に太字で書かれる．更に，それらは，（たいてい）以下のように「定義」として丸枠で囲われる．

定義 1.1

単一コスト RAM (**R**andom **A**ccess **M**emory) モデルとは，以下の操作が単一コスト（1 ステップ）で行えるという「計算モデル」である．

- データへのアクセス（読み取りと書き込み）
- 二つの値の四則演算
- 二つの値の大小比較

アルゴリズムが解く問題

本書で学習する「アルゴリズムが解く」問題は，すべて以下のように問題の名前を明記して丸枠で囲われる．

探索問題 (searching)

- 入力 : 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$
- 質問 : x が a_1, a_2, \dots, a_n に存在するか？

アルゴリズム

本書で学習するアルゴリズムは、すべて以下のようにシアン色で塗られる。

入力：整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$

1. それぞれの $i \in [n]$ について（順次）以下を繰り返す。
 - (a) $a_i = x$ であれば YES を出力して終了する。
2. NO を出力する。

命題，定理，補題，系

本書で学習する命題，定理，補題，系は、すべて以下のようにマゼンダ色で塗られる。

定理 1.1. 図 1.2 のアルゴリズム A は探索問題を解く。

本文中の問

本書にある問は、すべて以下のように灰色で塗られる。

問 1.1 (決定問題). 素数性判定問題，充足可能性問題とはどんな問題であるか調べ，探索問題や到達可能性問題になって問題の定義をそれぞれ示しなさい。（数学記号を適当に導入して，「入力」と「質問」を明確に示すこと.）

目次

第 1 章	アルゴリズムとは	1
1.1	探索問題を例に	3
1.2	最大公約数問題を例に	6
第 2 章	オーダー表記	11
第 3 章	データ構造	15
3.1	グラフ	15
3.2	木と根付き木	18
第 4 章	探索	23
4.1	線形探索	23
4.2	二分探索	23
第 5 章	整列	29
5.1	バブルソート	29
5.2	クイックソート	31
5.3	マージソート	34
5.4	ヒープソート	37
第 6 章	走査	43
6.1	幅優先探索とその応用	43
6.2	深さ優先探索とその応用	46
索引		58

第 1 章

アルゴリズムとは

アルゴリズムとは、「問題」を解くための機械的手順である。では、問題とは何か。問題には以下のように大きく二つの種類がある。

- 決定問題 (decision problem) : YES/NO や 1/0 や「はい・いいえ」など, 2 値のどちらかで答える問題.
- 関数問題 (function problem) : 自然数や集合や数列など, 「値」で答える問題.

決定問題

探索問題 (searching)

- 入力 : 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$
- 質問 : x が a_1, a_2, \dots, a_n に存在するか?

例 1.1 (探索問題).

- $x = 7, (a_1, \dots, a_5) = (7, 3, 5, 7, 11)$ の場合は YES.
- $x = 1, (a_1, \dots, a_5) = (7, 3, 5, 2, 11)$ の場合は NO.

到達可能性問題 (reachability)

- 入力 : グラフ $G = (V, E), s, t \in V$
- 質問 : s から t への経路が存在するか?

例 1.2 (到達可能性問題). 以下の図において, $V = \{1, 2, \dots, 7\}, s = 1, t = 7$ のとき, 入力グラフ G が左図 (a) の場合は YES, 右図 (b) の場合は NO.

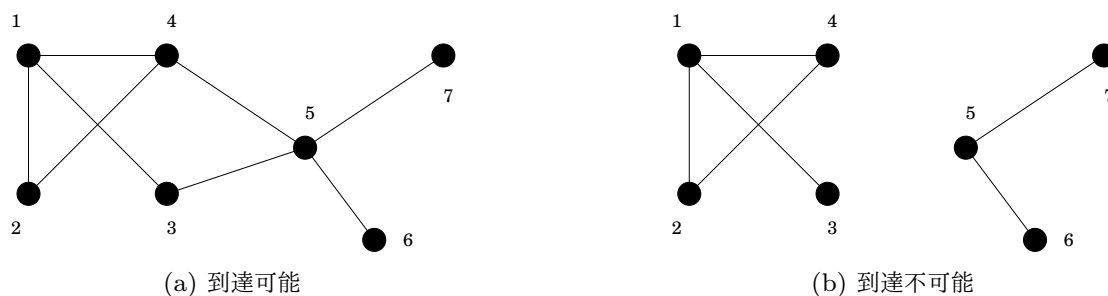


図 1.1: 到達可能性問題の例

関数問題

最大公約数問題 (greatest common divisor)

- 入力: 自然数 $x, y \in \mathbb{N}$
- 解: x, y の最大公約数

例 1.3 (最大公約数問題). $(x, y) = (12, 123)$ の場合, 解は 3.

整列問題 (sorting)

- 入力: 整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$
- 解: a_1, a_2, \dots, a_n の昇順, つまり, $a_{i_1}, a_{i_2}, \dots, a_{i_n}$ s.t. $a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_n}$

例 1.4 (整列問題). $(a_1, \dots, a_7) = (0, 7, -1, 11, 2, 3, 5)$ の場合, 解は $-1, 0, 2, 3, 5, 7, 11$.

他にも, 決定問題として, 素数性判定問題, 充足可能性問題などが, 関数問題として, 足し算・掛け算などの四則演算, 最短経路探索問題, 巡回セールスマン問題などがある. このように, (おおざっぱに言って) コンピュータで解くことのできる問題をアルゴリズムが解く「問題」とする.

問 1.1 (決定問題). 素数性判定問題, 充足可能性問題とはどんな問題であるか調べ, 探索問題や到達可能性問題にならって問題の定義をそれぞれ示しなさい. (数学記号を適当に導入して, 「入力」と「質問」を明確に示すこと.)

問 1.2 (関数問題). 最短経路探索問題, 巡回セールスマン問題とはどんな問題であるか調べ, 最大公約数問題や整列問題にならって問題の定義をそれぞれ示しなさい. (数学記号を適当に導入して, 「入力」と「解」を明確に示すこと.)

アルゴリズムとデータ構造の「理論」では, 提示されたアルゴリズムに対して, 以下を「数学的に」証明することが話の中心となる.

- アルゴリズムの 正当性 (どのような入力にも正しく動作する.)
- アルゴリズムの 計算量 (計算時間や計算領域など.)

1.1 探索問題を例に

アルゴリズムの擬似コード

以下の図 1.2 が, 探索問題を解く線形探索と呼ばれるアルゴリズムである.

入力: 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$

1. それぞれの $i \in [n]$ について (順次) 以下を繰り返す.
 - (a) $a_i = x$ であれば YES を出力して終了する.
2. NO を出力する.

図 1.2: 線形探索のアルゴリズム

このようなアルゴリズムの記述を擬似コードという. これを C++ 言語で記述すると図 1.3 のようになる.

アルゴリズムの正当性

定理 1.1. 図 1.2 のアルゴリズム A は探索問題を解く.

```

#include <iostream>
using namespace std;

#define n 100

int main()
{
    int a[n] = {321,2,3,-5,...};
    int x = -123;

    for (int i=0; i<n; i++) {
        if (a[i]==x) {
            cout << YES << endl;
            return 0;
        }
    }
    cout << NO << endl;
    return 0;
}

```

図 1.3: 線形探索の C++ コード

証明. 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$ を任意とする. 定理を示すためには以下のことを示せばよい.

$$\begin{array}{ll}
 x \in \{a_1, \dots, a_n\} & \text{のとき} \quad A(x, (a_1, \dots, a_n)) = \text{YES} \\
 x \notin \{a_1, \dots, a_n\} & \text{のとき} \quad A(x, (a_1, \dots, a_n)) = \text{NO}
 \end{array}$$

まず, 一つ目を示す. この場合, $\forall i \in [i_0 - 1][a_i \neq x]$ かつ $a_{i_0} = x$ を満たす $i_0 \in [n]$ が存在する. このとき, アルゴリズムより, $i = i_0$ において YES を出力して終了するのは明らかである. よって, $A(x, (a_1, \dots, a_n)) = \text{YES}$ である.

次に, 二つ目を示す. このとき, $\forall i \in [n][a_i \neq x]$ であることから, アルゴリズムより, ステップ 1-(a) の条件 ($a_i = x$) は成り立たないので, ステップ 1-(a) で YES を出力して終了することはない. ステップ 1 はいずれ終了するので, ステップ 2 で NO を出力して終了するのは明らかである. よって, $A(x, (a_1, \dots, a_n)) = \text{NO}$ である. ■

アルゴリズムの計算時間

定義 1.1

単一コスト RAM (Random Access Memory) モデル とは、以下の操作が単一コスト (1 ステップ) で行えるという「計算モデル」である。

- データへのアクセス (読み取りと書き込み)
- 二つの値の四則演算
- 二つの値の大小比較

以降、アルゴリズムの計算時間の解析には、それを単純化するため (計算時間を解析するにあたり「本質的」でない部分の解析を省略するため)、単一コスト RAM モデルを仮定する。

定義 1.2

入力 x に対して、 $|x|$ を x の長さとする。

各問題において、入力の長さ (入力長) をどのように考えればよいだろうか。例えば、探索問題の入力 $(3.14, (1.23, -1, 0, 456, 98.7))$ の入力長はいくらだろうか。

例 1.5. 下記の問題に対する入力長は、それぞれ以下のようにするのが慣習である。

探索問題	:	$ (x, (a_1, \dots, a_n)) $	=	n
到達可能性問題	:	$ (G = (V, E), s, t) $	=	$ V + E $
最大公約数問題	:	$ (x, y) $	=	$\log x + \log y$
整列問題	:	$ (a_1, \dots, a_n) $	=	n

定義 1.3

アルゴリズム A の計算時間が $f(n)$ であるとは、 $|x| = n$ である 任意の 入力 x に対して、 $A(x)$ が (単一コスト RAM モデルで) $f(n)$ 「ステップ」 以内 で終了することである。これを、最悪時間計算量 という。

定理 1.2. 図 1.2 のアルゴリズム A の計算時間は $3n + 1$ である。

証明. 最悪時間を見積もるので、 $x \notin \{a_1, \dots, a_n\}$, つまり、 $\forall i \in [n][a_i \neq x]$ である入力 $x, (a_1, \dots, a_n)$ の場合を考えればよい。

問 1.3. この理由を説明しなさい.

アルゴリズムの各ステップの計算時間を見積もる. ステップ 1 の「それぞれの $i \in [n]$ について (順次) 以下を繰り返す」に時間を要する箇所は, i のインクリメント操作 ($i+1$ の演算) と $i \leq n$ の判定をする部分である. 単一コスト RAM モデルの仮定より, 計算時間はそれぞれ 1 となる. 「 $a_i = x$ であれば YES を出力して終了する」に時間を要する箇所は, $a_i = x$ の判定をする部分である. ($\forall i \in [n][a_i \neq x]$ であるので, YES の出力にかかる時間を考える必要はない.) 単一コスト RAM モデルの仮定より, 計算時間は 1 となる. これらの計算時間は合計で 3 となり, それぞれが n 回行われる. (よって, ステップ 1 にかかる計算時間の合計は $3n$ となる.) また, ステップ 2 の「NO を出力する」は, 単一コスト RAM モデルの仮定より, 計算時間は 1 となる. よって, アルゴリズム全体の計算時間は $3n + 1$. ■

問 1.4. 上の証明では, $\forall i \in [n][a_i \neq x]$ である入力 $x, (a_1, \dots, a_n)$ の場合の計算時間を見積もった. (証明の中の間で, その場合を考えれば十分であることを確認した.) では, 実際に, $\exists i \in [n][a_i = x]$ である入力 $x, (a_1, \dots, a_n)$ の場合の計算時間を見積もり, 上の間で示したことの正しさを確認しなさい. (計算時間の導出を示すこと.)

1.2 最大公約数問題を例に

以下の図 1.4 が, 最大公約数問題を解くユークリッドの互除法と呼ばれるアルゴリズムである.

入力: 自然数 x, y

1. $y > 0$ である限り以下を繰り返す.
 - (a) $r = x \bmod y$ とする.
 - (b) $x = y, y = r$ とする.
2. x を出力する.

図 1.4: ユークリッドの互除法のアルゴリズム

これを C++ 言語で記述すると図 1.5 のようになる.

```
#include <iostream>
using namespace std;

int main()
{
    int x = 123, y = 12, r;

    while (y > 0) {
        r = x % y;
        x = y;
        y = r;
    }
    cout << x << endl;
    return 0;
}
```

図 1.5: ユークリッドの互除法の C++ コード

定理 1.3. 図 1.4 のアルゴリズム A は最大公約数問題を解く.

証明. 略. ■

問 1.5. この定理を証明しなさい.

定理 1.4. $x \geq y$ とする. 図 1.4 のアルゴリズム A の計算時間は $8 \log y + 9$ である.

証明. まず, アルゴリズムのステップ 1 の繰り返し回数を見積もる. その回数を $k+1$ とする. 任意の $i \in [k+1]$ について, ステップ 1 の第 i 回目の繰り返し後の r を r_i とする. ($r_{-1} = x, r_0 = y$ とする.) このとき, ある整数 q_1, q_2, \dots, q_{k+1} が存在して, 以下の等式が成り立つ. (左の列は繰り返し回数を表す.)

$$\begin{array}{rclclcl}
 1 & : & x & = & q_1 y & + & r_1 \\
 2 & : & y & = & q_2 r_1 & + & r_2 \\
 3 & : & r_1 & = & q_3 r_2 & + & r_3 \\
 & \vdots & & & & & \\
 k-1 & : & r_{k-3} & = & q_{k-1} r_{k-2} & + & r_{k-1} \\
 k & : & r_{k-2} & = & q_k r_{k-1} & + & r_k \\
 k+1 & : & r_{k-1} & = & q_{k+1} r_k & + & r_{k+1}
 \end{array}$$

問 1.6. r_{k+1} はいくつか.

主張 1.1. 任意の $i \in \{-1, 0, 1, 2, \dots, k-1\}$ について,

$$r_{i+2} \leq \frac{r_i}{2}.$$

証明. アルゴリズムより, (第 $i+2$ 回目の繰り返しにおいて) ある整数 q が存在して,

$$r_i = q r_{i+1} + r_{i+2}$$

$r_{i+1} \leq r_i/2$ の場合, 余りの定義より,

$$r_{i+2} < r_{i+1} \leq \frac{r_i}{2}.$$

$r_{i+1} > r_i/2$ の場合, $q = 1$ より,

$$r_i = r_{i+1} + r_{i+2}$$

よって,

$$r_{i+2} = r_i - r_{i+1} < r_i - \frac{r_i}{2} = \frac{r_i}{2}.$$

■

k が偶数であるとする．この主張より，($r_{k+1} = 0$ より $r_k \geq 1$ であるから)

$$1 \leq r_k \leq \frac{r_{k-2}}{2} \leq \frac{r_{k-2 \cdot 2}}{2^2} \leq \frac{r_{k-2 \cdot 3}}{2^3} \leq \dots \leq \frac{r_{k-2 \cdot i}}{2^i} \leq \dots \leq \frac{r_0}{2^{k/2}} = \frac{y}{2^{k/2}}.$$

よって， $2^{k/2} \leq y$ ，つまり， $k \leq 2 \log y$ ．（ステップ 1 の繰り返し回数は高々 $2 \log y + 1$ ．）

問 1.7. k が奇数の場合の繰り返し回数は高々いくつになるか．

ステップ 1 の 1 回の繰り返しでは，繰り返しの条件 $y > 0$ の判定と， $r = x \bmod y$ ， $x = y$ ， $y = r$ の三つの操作が行われる．（よって，1 回の繰り返しにつき 4 つの操作が行われる．）単一コスト RAM モデルの仮定より，計算時間はそれぞれ 1 となる．また，ステップ 2 の「 x を出力する」は，単一コスト RAM モデルの仮定より，計算時間は 1 となる．よって，アルゴリズム全体の計算時間は，(k が奇数の場合の方が繰り返し回数は多いので) 高々，

$$4 \cdot (2 \log y + 2) + 1 = 8 \log y + 9.$$

■

問 1.8. 自然数は二進数で扱われるとする．アルゴリズムの入力を $(x, y) \in \mathbb{N} \times \mathbb{N}$ とした場合，入力長 $|(x, y)|$ の（およその）値を示しなさい．

次章以降，アルゴリズムはすべて疑似コードで記述される．（C++コードは省略する．）また，計算時間の解析には，単一コスト RAM モデルを仮定する．

第 2 章

オーダー表記

オーダー表記 $O(\cdot)$ の記号 “ O ” は、ランダウの記号 (Landau symbol) と呼ばれる。

定義 2.1

$f, g: \mathbb{Z} \rightarrow \mathbb{Z}$ を関数とする。 $f(n) = O(g(n))$ であるとは、

$$\exists c \in \mathbb{Z}, \exists n_0 \in \mathbb{Z}, \forall n \geq n_0 [f(n) \leq c \cdot g(n)].$$

直感的に、 $f(n) = O(g(n))$ とは、 $\lim_{n \rightarrow \infty} f(n)/g(n)$ が定数 (n によらない整数) 「以下」になることである。(つまり、 $f(n) \leq c \cdot g(n)$.)

例 2.1. 以下は正しいオーダー表記である。

- $1 = O(1)$, $123 = O(1)$
- $1 = O(n)$, $n = O(n)$, $123n + 12345 = O(n)$
- $100n = O(n \log n)$
- $n = O(n^{1.1})$
- $n = O(n^2)$
- $100n^3 + n^2 + n + 2^{100} = O(n^3)$
- $n^{100} = O(2^n)$

一方、以下は正しくないオーダー表記である。

- $n = O(1)$
- $n \log n = O(n)$
- $1.001^n = O(n^{100})$

以降、特に断りがない限り、対数関数 \log の底は 2 とする。

注 2.1. オーダー表記を用いると、定理 1.2 より線形探索は $O(n)$ で、定理 1.4 よりユークリッドの互除法は ($x \geq y$ の場合) $O(\log y)$ で、アルゴリズムが終了するということがある。

問 2.1. 以下の極限を求めなさい。

- $\lim_{n \rightarrow \infty} \frac{n \log n}{n^{1.01}}$
- $\lim_{n \rightarrow \infty} \frac{n^{100}}{1.01^n}$

問 2.2. 以下のうち正しいものを選び出さないさい。

- | | |
|--|--|
| 1. $2^{100} = O(1)$ | 2. $\log n = O(1)$ |
| 3. $n + \sqrt{n} + \log n = O(n)$ | 4. $n \log n - 100n = O(n)$ |
| 5. $n\sqrt{n} + n^{1.1} \log n = O(n^2)$ | 6. $n^{1.001} = O(n \log n)$ |
| 7. $n = O(2^{\log n})$ | 8. $n^3 = O(2^{2 \log n})$ |
| 9. $n^{100} = O(1.001^n)$ | 10. $n^{1.001} = O(2^{\sqrt{\log n}})$ |
| 11. $n^{\log n} = O(1.001^{\sqrt{n}})$ | 12. $1.001^n = O(2^{\sqrt{n}})$ |

定義 2.2

n を入力長とする。アルゴリズム A が多項式時間で終了するとは、ある（正の）定数 $k = O(1)$ が存在して、 A の計算時間が $O(n^k)$ であることである。また、アルゴリズム A が指数時間で終了するとは、ある（正の）定数 $a = O(1)$ が存在して、 A の計算時間が $O(a^n)$ であることである。

例 2.2. n を入力長とする。以下で示された計算時間は多項式時間である。

- 100
- $2n^2 \log n + n(\log n)^{10} + 5$
- $n^{1000} + 2^{100}$

以下で示された計算時間は（多項式時間ではない）指数時間である。

- 2^n
- $100^{n/100}$
- $n^{\log n} + n^3 + 10$

注 2.2. 線形探索, ユークリッドの互除法は多項式時間アルゴリズムである. (線形探索は入力 $x, (a_1, \dots, a_n)$ に対して $O(n)$, ユークリッドの互除法は入力 x, y に対して^{*1} ($x \geq y$ の場合) $O(\log y)$.) 一方, 充足可能性問題や巡回セールスマン問題に対しては, 指数時間アルゴリズムしか発見されていない.

問 2.3. n を入力長とする. 以下で示された計算時間のうち多項式時間を選び出さない.

- | | |
|---------------------------------|-----------------------------|
| 1. 2^{100} | 2. $\log n$ |
| 3. $n\sqrt{n} + n^{1.1} \log n$ | 4. $n^{1.001}$ |
| 5. $n^{100} + 100n$ | 6. $n^{1.001} - 100n - 100$ |
| 7. $n^{\log n}$ | 8. 1.001^n |

^{*1} 入力長は (およそ) $\log x + \log y$ である.

第3章

データ構造

3.1 グラフ

定義 3.1

V を頂点の集合としたとき，頂点对の集合 $E \subseteq V \times V$ を辺の集合という．このとき， $G = (V, E)$ を V 上の有向グラフという．辺の集合 E において， (u, v) と (v, u) を同じ辺とみなすとき， $G = (V, E)$ を無向グラフという．有向グラフの辺を有向辺，無向グラフの辺を無向辺という．

以降では，特に断らない限り，グラフといった場合，無向グラフを指すものとする．

例 3.1 (グラフ)．以下は， $V = \{1, 2, 3, 4, 5\}$ 上の無向グラフ $G = (V, E)$ である．

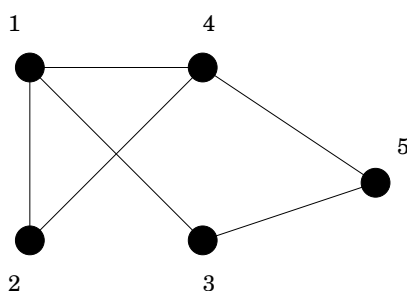


図 3.1: 無向グラフ

このグラフ G の頂点集合 V ，辺集合 E は，

$$\begin{aligned} V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1, 2), (1, 3), (1, 4), (2, 3), (3, 4), (3, 5), (4, 5)\} \end{aligned}$$

問 3.1. 任意のグラフ $G = (V, E)$ について, $|E| \leq |V|(|V| - 1)/2$ である. この事実を示しなさい.

定義 3.2

$G = (V, E)$ をグラフとする. 頂点 $u, v \in V$ について $(u, v) \in E$ であるとき, u と v は隣接するという. 頂点 u に隣接する頂点の集合を N_u と表記する. つまり, $N_u = \{v \in V : (u, v) \in E\}$. u の次数とは, u の隣接頂点の個数, つまり, $|N_u|$ のことである.

例 3.2 (次数). 図 3.1 のグラフ $G = (V, E)$ では,

$$\begin{aligned} N_1 &= \{2, 3, 4\} \\ N_2 &= \{1, 4\} \\ N_3 &= \{1, 5\} \\ N_4 &= \{1, 2, 5\} \\ N_5 &= \{3, 4\} \end{aligned}$$

よって, 頂点 1, 4 の次数は 3, 頂点 2, 3, 5 の次数は 2 である.

命題 3.1. $G = (V, E)$ をグラフとする.

$$\sum_{u \in V} |N_u| = 2|E|.$$

問 3.2. 上の命題を示しなさい.

$G = (V, E)$ をグラフとする. G を表すデータ構造として, 以下の二つがある.

- 隣接行列
- 隣接リスト

例 3.3 (隣接行列, 隣接リスト). 図 3.1 の ($V = \{1, 2, 3, 4, 5\}$ 上の) グラフを隣接行列

で表すと以下となる.

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

また, 隣接リストで表すと以下となる.

$$\begin{aligned} 1 & : 2, 3, 4 \\ 2 & : 1, 4 \\ 3 & : 1, 5 \\ 4 & : 1, 2, 5 \\ 5 & : 3, 4 \end{aligned}$$

問 3.3. $G = (V, E)$ をグラフとする. G を隣接行列で保持した場合, 必要なメモリ領域は $O(|V|^2)$. 一方, 隣接リストで保持した場合は $O(|V| + |E|)$. この事実の理由を説明しなさい.

定義 3.3

$G = (V, E)$ をグラフとする. 任意の $s, t \in V$ について, s から t への G 上の経路 (パス) とは, 以下を満たす頂点の列 $P = (v_0, v_1, v_2, \dots, v_k)$ である.

1. $v_0 = s$
2. $v_k = t$
3. $\forall i \in [k][(v_{i-1}, v_i) \in E]$

特に, 任意の $i, j \in [k] \cup \{0\}$ ($i \neq j$) について $v_i \neq v_j$ であるとき, P を単純経路という. 単純経路 $P = (v_0, v_1, v_2, \dots, v_k)$ の長さを k とする.

例 3.4 (経路). 図 3.2 で示された太線は, 図 1.1 (a) のグラフ $G = (V, E)$ 上の, 頂点 1 から頂点 7 への (単純) 経路である.

定義 3.4

$G = (V, E)$ をグラフとする. $P = (v_0, v_1, \dots, v_k)$ を v_0 から v_k への経路とする. $v_0 = v_k$ であるとき, P を閉路という. 特に, 任意の $i, j \in [k] \cup \{0\}$ ($i \neq j$) について $v_i \neq v_j$ であるとき, P を単純閉路という. 単純閉路 $P = (v_0, v_1, v_2, \dots, v_k)$ の長さを k とする.

例 3.5 (閉路). 図 3.3 で示された太線は, 図 1.1 (a) のグラフ $G = (V, E)$ 上の, 頂点 1

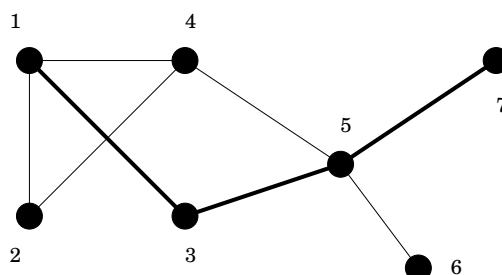


図 3.2: 経路

を通る（単純）閉路である．

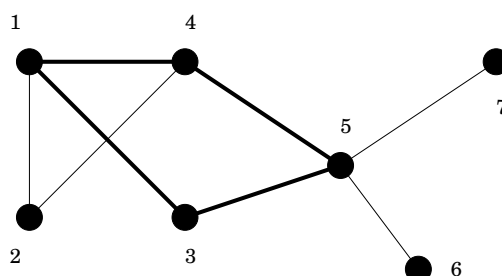


図 3.3: 閉路

以降では、特に断らない限り、経路・閉路といった場合、単純経路・単純閉路を指すものとする．

3.2 木と根付き木

定義 3.5

$G = (V, E)$ をグラフとする．頂点 u から頂点 v への経路が存在するとき、 u, v は連結しているという．任意の頂点 $u, v \in V$ について u, v が連結しているとき、 G を連結グラフという．

例 3.6 (連結グラフ)．以下の図の $V = \{1, 2, \dots, 7\}$ 上のグラフにおいて、左図 (a) は連結、右図 (b) は非連結．

定義 3.6

$G = (V, E)$ を連結グラフとする． G に閉路が存在しないとき、 G を木という．

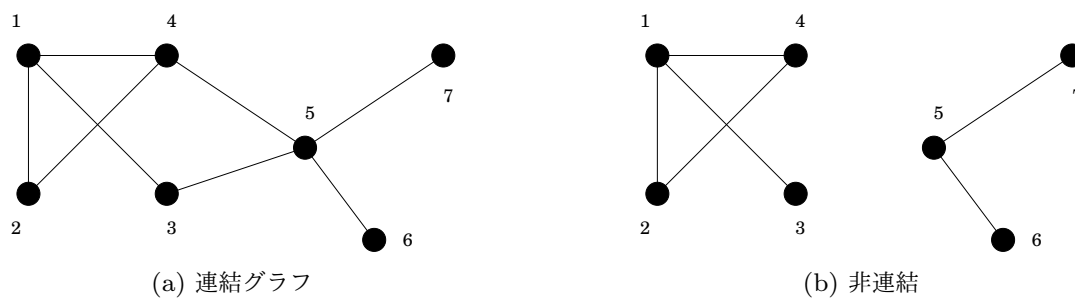


図 3.4: 連結・非連結

例 3.7 (木). 以下は, $V = \{1, 2, 3, 4, 5\}$ 上の木 $G = (V, E)$ である.

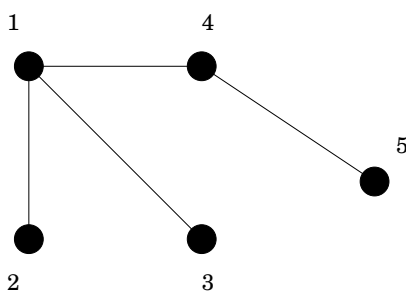


図 3.5: 木

定理 3.2. $G = (V, E)$ を木とする. このとき,

- 任意の頂点 u, v について, u から v への経路は一意的である.
- どの辺 $(u, v) \in E$ を除去しても連結でなくなる.
- どの辺 $(u, v) \in (V \times V) \setminus E$ を追加しても唯一の閉路ができる.

命題 3.3. $G = (V, E)$ が木であれば $|E| = |V| - 1$.

系 3.4. 任意の連結グラフ $G = (V, E)$ について, $|V| - 1 \leq |E|$. よって, $|V| = O(|E|)$.

問 3.4. 上の系を示しなさい.

定義 3.7

$G = (V, E)$ を連結グラフとする. 任意の $u, v \in V$ について, u から v への経路の長さの最小を u から v への距離といい, $d(u, v)$ と表す.

命題 3.5. $G = (V, E)$ を木, $r \in V$ を任意の頂点とする. このとき, 任意の $(u, v) \in E$ について, $|d(r, u) - d(r, v)| = 1$

定義 3.8

$T = (V, E)$ を木, $r \in V$ を T の任意の頂点とする. T の任意の無向辺 $(u, v) \in E$ について, $d(r, u) < d(r, v)$ となるように有向辺 (u, v) を考えた場合 (上の命題よりそのような向きは一意である), 「有向木」と頂点 r の対 (T, r) を r を根とした根付き木という.

T の任意の有向辺 (u, v) について, u を v の親, v を u の子という. 子をもたない頂点を T の葉という. r から葉までの距離の最大を T の深さという.

例 3.8 (根付き木). 以下の図は, 頂点 r を根とした根付き木である. (各有向辺は下向きであり矢印は省略されている.)

定義 3.9

$(T = (V, E), r)$ を根付き木とする. 任意の頂点 $v \in V$ について, $(u, v) \in E$ となる頂点 u の個数を v の入次数, $(v, w) \in E$ となる頂点 w の個数を v の出次数という.

定義 3.10

(T, r) を根付き木とする. 葉以外の任意の頂点の出次数が「高々」2 であるとき, T を二分木という. 更に, 任意の葉 u について $d(r, u)$ が一定であるとき, 二分木 T を完全二分木という.

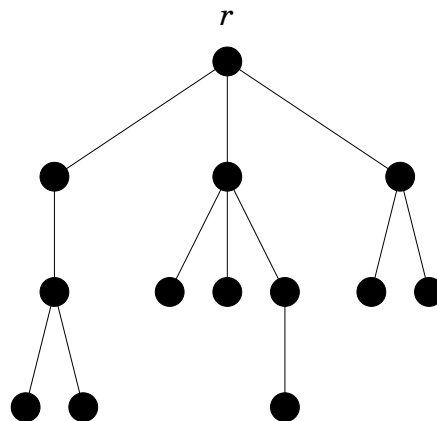


図 3.6: 根付き木

例 3.9 (二分木). 以下の図 3.7 は r を根とした二分木と完全二分木である.

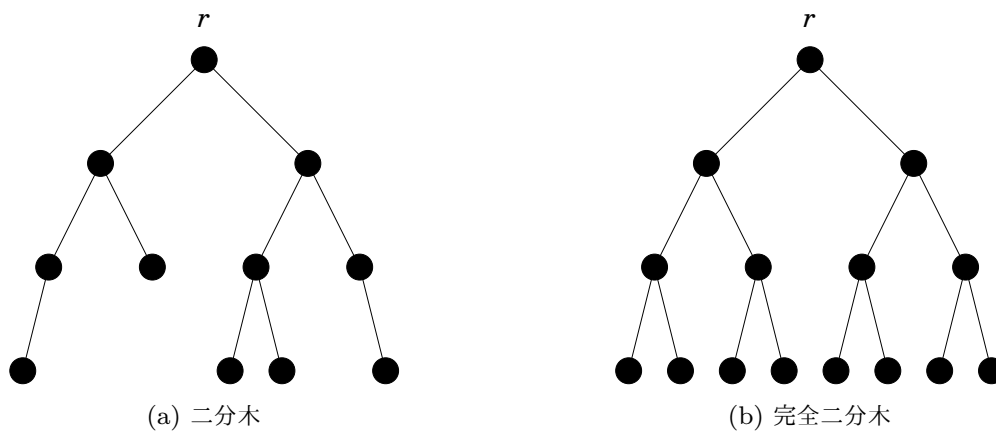


図 3.7: 二分木と完全二分木

命題 3.6. $(T = (V, E), r)$ を完全二分木とする. $|V| = n$ のとき, T の深さは $\log(n + 1) - 1$ である.

問 3.5. 上の命題を示しなさい.

第 4 章

探索

探索問題 (searching)

- 入力 : 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$
- 質問 : x が a_1, a_2, \dots, a_n に存在するか?

4.1 線形探索

第 1.1 節を参照.

4.2 二分探索

図 4.1 に二分探索のアルゴリズムを示す. これは, 入力 (a_1, \dots, a_n) が昇順 (や降順) になっていた場合の探索手法である. (そうでない場合はそのような探索はできない.)

入力 : 整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$, ただし, $a_1 \leq a_2 \leq \dots \leq a_n$

- $s = 1, t = n$ として以下を繰り返す.
 1. $s > t$ であれば NO を出力して終了する.
 2. $l = \lfloor (s + t) / 2 \rfloor$ とする.
 3. $a_l = x$ であれば YES を出力して終了する.
 4. $a_l < x$ であれば $s = l + 1$, $a_l > x$ であれば $t = l - 1$ とする.

図 4.1: 二分探索

問 4.1. YES の場合と NO の場合それぞれに対して、ある一つの整数と 7 個の整数からなる昇順列を具体的にあげ、それに対して二分探索のアルゴリズムを適用させなさい。このとき、その入力によるアルゴリズムの動作を説明すること。

定理 4.1. 図 4.1 のアルゴリズム A は探索問題を解く。また、 A の計算時間は $O(\log n)$ である。

証明. まず、アルゴリズムの正当性を示す。そのためには以下のことを示せばよい。

$$\begin{array}{ll} x \in \{a_1, \dots, a_n\} & \text{のとき} \quad A(x, (a_1, \dots, a_n)) = \text{YES} \\ x \notin \{a_1, \dots, a_n\} & \text{のとき} \quad A(x, (a_1, \dots, a_n)) = \text{NO} \end{array}$$

まず、二つ目を示す。このとき、 $\forall i \in [n][a_i \neq x]$ であることから、アルゴリズムより、ステップ 3 の条件は成り立たないので、ステップ 3 で YES を出力して終了することはない。また、 $t - s$ の値は単調に減少していくので、いずれはステップ 1 の条件が成り立って、 $A(x, (a_1, \dots, a_n)) = \text{NO}$ となる。

次に、一つ目を示す。 $\exists i \in [n][a_i = x]$ とする。このとき、 $A(x, (a_1, \dots, a_n)) = \text{YES}$ となることを、 $n = t - (s - 1)$ (データの個数) についての帰納法により示す。そのため、 A を以下のように再帰を用いた記述に書き換える。(アルゴリズムとしては全く同じである。)

入力：整数 $x \in \mathbb{Z}$ と整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$ ，ただし、 $a_1 \leq a_2 \leq \dots \leq a_n$

- $s = 1, t = n$ として $\text{bsearch}(s, t)$ を実行する。

$\text{bsearch}(s, t)$

1. $s > t$ であれば NO を出力して終了する。
2. $l = \lfloor (s + t) / 2 \rfloor$ とする。
3. $a_l = x$ であれば YES を出力して終了する。
4. $a_l < x$ であれば $s = l + 1$ ， $a_l > x$ であれば $t = l - 1$ として、 $\text{bsearch}(s, t)$ を実行する。

図 4.2: 二分探索：再帰を用いた記述

【初期段階】 $n = 1$ のときは明らか。

問 4.2. その理由を詳細に説明しなさい.

【帰納仮定】 $k \geq 2$ である自然数 $k \in \mathbb{N}$ を任意に固定する. 任意の自然数 $n \leq k - 1$ について, アルゴリズムが正しく動作するとする.

問 4.3. $k \geq 2$ という条件がある理由を詳細に説明しなさい.

【帰納段階】 上の仮定を用いて, $n = k$ について, アルゴリズムが正しく動作することを示す. $\exists i \in [n][a_i = x]$ とする. $s = 1, t = n$ として, $\text{bsearch}(s, t)$ の動作を考える. $s \leq t$ ($s = 1, t = n, n = k \geq 2$) であることからステップ 1 の条件は満たされない. $l = \lfloor (s + t)/2 \rfloor$ とする. $a_l = x$ であれば YES を出力して終了するので, アルゴリズムは正しく動作する. そうでないとき, $a_l < x$ の場合, a_1, \dots, a_n は昇順で並んでいることから, $(\exists i \in [n][a_i = x])$ が満たされているので $\exists i \in [n] \setminus [\ell][a_i = x]$ (x が上位に存在する) が成り立つ.

問 4.4. $a_l > x$ の場合はどういう条件が成り立つか. 条件式を示しなさい.

$a_l < x$ のとき, 帰納仮定より, $\exists i \in [n] \setminus [\ell][a_i = x]$ の場合は $\text{bsearch}(l + 1, n)$ は正しく動作する. また, $a_l > x$ のときは ($a_l < x$ のときと同様の理由で) $\text{bsearch}(1, l - 1)$ も正しく動作する. よって, $n = k$ のときもアルゴリズムは正しく動作することがいえる. 以上より, アルゴリズムの正当性が示される.

次に, 計算時間を見積もる. アルゴリズムの正当性を示したときと同様, 再帰を用いた記述のアルゴリズムの計算時間を見積もる. (アルゴリズムとして全く同じであることから計算時間も同じになる.) $\text{bsearch}(1, n)$ の計算時間を見積もればよい. このために, 次のように漸化式を立てる. $\text{bsearch}(s, t)$ (ただし, $t - (s - 1) = \alpha$) の実行にかかる計算時間を $T(\alpha)$ とする. 再帰を用いた記述より, ある定数 c, c' が存在して,

$$\begin{aligned} T(\alpha) &\leq T(\lfloor \alpha/2 \rfloor) + c \\ T(1) &\leq c' \end{aligned} \tag{4.1}$$

問 4.5. 上の不等式のうち一つ目が成り立つ理由を説明しなさい. 特に, 単一コスト RAM モデルでは, $c = 10$ としておけば 十分 である. これら単一時間ですむ箇所をすべてあげなさい.

問 4.6. 上の不等式のうち二つ目が成り立つ理由を説明しなさい。特に、 c' と c の大小関係を、アルゴリズムが YES を出力して終了する場合と NO を出力して終了する場合に分けて説明しなさい。

この漸化式は次のように解くことができる。まず、ある非負整数 k に対して $n = 2^k \geq 1$ であるとする。（ $k = \log n$ 。）このとき、

$$T(n) \leq T(n/2) + c \quad (4.2)$$

$$T(n/2) \leq T(n/2^2) + c \quad (4.3)$$

$$T(n/2^2) \leq T(n/2^3) + c \quad (4.4)$$

$$\vdots$$

上の式において、不等式 (4.3) を不等式 (4.2) へ代入すると、

$$\begin{aligned} T(n) &\leq T(n/2) + c \\ &\leq (T(n/2^2) + c) + c \\ &= T(n/2^2) + 2c. \end{aligned}$$

更に、この不等式に不等式 (4.4) を代入すると、

$$\begin{aligned} T(n) &\leq T(n/2^2) + 2c \\ &\leq (T(n/2^3) + c) + 2c \\ &= T(n/2^3) + 3c. \end{aligned}$$

このように、順次、不等式を代入していけば、

$$\begin{aligned} T(n) &\leq T(n/2) + c \\ &\leq T(n/2^2) + 2c \\ &\leq T(n/2^3) + 3c \\ &\vdots \\ &\leq T(n/2^i) + ic. \end{aligned}$$

$n = 2^k$ より、上の不等式で $i = k$ とおけば、

$$T(n) \leq T(1) + k \cdot c.$$

更に、 $k = \log n$ ($\because n = 2^k$), $T(1) \leq c'$ (\because 不等式 (4.1)) より、

$$T(n) \leq c \log n + c' = O(\log n).$$

一般の自然数 n についても、上の式が成り立つことを帰納法により示す。

問 4.7. 任意の自然数 n について, $T(n) \leq c \log n + c'$ が成り立つことを帰納法により示しなさい. (定数 c, c' は不等式 (4.1) のものに同じ.)

よって, 任意の自然数 n について, $T(n) \leq c \log n + c' = O(\log n)$.



第 5 章

整列

整列問題 (sorting)

- 入力: 整数の列 $a_1, a_2, \dots, a_n \in \mathbb{Z}$
- 解: a_1, a_2, \dots, a_n の昇順, つまり, $a_{i_1}, a_{i_2}, \dots, a_{i_n}$ s.t. $a_{i_1} \leq a_{i_2} \leq \dots \leq a_{i_n}$

以下, 疑似コード中の配列は, 配列名を a , 配列の大きさを n としたとき, 配列のインデックス番号は, 1 から n となっているものとする. つまり, $a[1], a[2], \dots, a[n]$ に値が入っているものとする. (C++ 言語では, $a[0], a[1], \dots, a[n-1]$ となる.)

5.1 バブルソート

図 5.1 に, バブルソートのアルゴリズムを示す.

入力: 整数の列 $a_1, \dots, a_n \in \mathbb{Z}$

1. a を大きさ n の配列として, それぞれの $i \in [n]$ について $a[i] = a_i$ とする.
2. それぞれの $i \in [n-1]$ について (順次) 以下を繰り返す.
 - (a) それぞれの $j \in [n-i]$ について (順次) 以下を繰り返す.
 - $a[j] > a[j+1]$ であれば $a[j]$ と $a[j+1]$ の値を入れ替える.
3. 配列 a の値を (順次) 出力する.

図 5.1: バブルソート

問 5.1. 異なる7個の整数の列を具体的にあげ、それに対してバブルソートのアルゴリズムを適用させなさい。このとき、その入力によるアルゴリズムの動作を説明すること。

定理 5.1. 図 5.1 のアルゴリズム A は整列問題を解く。つまり、入力 a_1, \dots, a_n の A の出力 $A(a_1, \dots, a_n)$ は、 a_1, \dots, a_n の昇順である。また、 A の計算時間は $O(n^2)$ である。

証明. まず、アルゴリズムの正当性を示す。

主張 5.1. 任意の $i \in [n-1]$ について次のことが成り立つ。アルゴリズムのステップ 2 の第 i 回目の繰り返しにおいて、ステップ 2-(a) の繰り返し後、 $a[1], \dots, a[n-i+1]$ の最大のものが $a[n-i+1]$ に移動する。

証明. $a[1], \dots, a[n-i+1]$ の最大を $a[\ell]$ とおく。 ($1 \leq \ell \leq n-i+1$.) ■

問 5.2. この主張の証明を完成させなさい。

主張 5.2. 任意の $i \in [n-1]$ について次のことが成り立つ。アルゴリズムのステップ 2 の第 i 回目の繰り返しにおいて、ステップ 2-(a) の繰り返し後、以下のことが成り立つ。

1. 任意の $j \in [n-i]$ について $a[j] \leq a[n-i+1]$.
2. $a[n-i+1], a[n-i+2], \dots, a[n]$ は昇順である。

証明. $i \in [n-1]$ についての帰納法により証明する。 $i=1$ のとき、上の主張より、ステップ 2-(a) の繰り返しによって、 $a[1], \dots, a[n]$ から最大のものが $a[n]$ に移動する。これより、主張の二つの条件が成り立つことは明らかである。

問 5.3. 何がどのように明らかなのか、略された証明をそれぞれ示しなさい。

$i=k$ のとき、主張の二つの条件が成り立つとする。 $i=k+1$ のとき、ステップ 2 の第 $k+1$ 回目の繰り返しを考える。上の主張より、ステップ 2-(a) の繰り返しによって、 $a[1], \dots, a[n-k]$ から最大のものが $a[n-k]$ に移動する。よって、すべての $j \in [n-k]$ について $a[j] \leq a[n-k]$ 。(主張の一つ目の条件が示された。) また、帰納仮定より、 $a[n-k] \leq a[n-k+1]$ 。

問 5.4. ここで用いた帰納仮定とはどういう仮定か.

更に, (帰納仮定より) $a[n-k+1], \dots, a[n]$ は昇順であることから, $a[n-k], a[n-k+1], \dots, a[n]$ は昇順となる. (主張の二つ目の条件が示された.) ■

この主張に $i = n-1$ を適用すればアルゴリズムの正当性が示される.

問 5.5. なぜ, この主張に $i = n-1$ を適用すればアルゴリズムの正当性が示されるのか, その理由を説明しなさい.

次に, アルゴリズムの計算時間を見積もる. まず, ステップ 1, ステップ 3 にかかる計算時間は, それぞれ $O(n)$ である. よって, ステップ 2 全体にかかる計算時間が $O(n^2)$ であることを示せばよい. まず, それぞれの $i \in [n-1]$ について, ステップ 2-(a) の繰り返し全体にかかる計算時間は $O(n-i)$ である. つまり, ある定数 c が存在して $c(n-i)$ 以下である. よって, ステップ 2 全体にかかる計算時間は, 高々,

$$\sum_{i \in [n-1]} c(n-i) = c \cdot \sum_{i \in [n-1]} i = c \cdot \frac{n(n-1)}{2} = O(n^2).$$

■

注 5.1. 選択ソート, 挿入ソートは, バブルソートと本質的に同じアルゴリズムである.

5.2 クイックソート

図 5.2 に, クイックソートのアルゴリズムを示す.

注 5.2. 図 5.2 のアルゴリズムの `qsort(a, x, y)` (のステップ 4) において, $a[x]$ をピボットという.

問 5.6. 異なる 7 個の整数の列を具体的にあげ, それに対してクイックソートのアルゴリズムを適用させなさい. このとき, その入力によるアルゴリズムの動作を説明すること.

入力：整数の列 $a_1, \dots, a_n \in \mathbb{Z}$

1. a を大きさ n の配列として、それぞれの $i \in [n]$ について $a[i] = a_i$ とする.
2. $\text{qsort}(a, 1, n)$ を実行する.
3. 配列 a の値を（順次）出力する.

$\text{qsort}(a, x, y)$

1. $x \geq y$ であればリターン.
2. b を大きさ $y - x + 1$ の配列とする.
3. $i = 1, j = y - x + 1$ とする.
4. それぞれの整数 $z : x + 1 \leq z \leq y$ について（順次）以下を繰り返す.
 - (a) $a[z] < a[x]$ であれば、 $b[i] = a[z]$, $i++$ とする.
 - (b) $a[z] \geq a[x]$ であれば、 $b[j] = a[z]$, $j--$ とする.
5. $b[i] = b[j] = a[x]$ とする. ($i = j$ となっている.)
6. $a[x, \dots, y] = b[1, \dots, y - x + 1]$ とする.
7. $z = x + i - 1$ として、 $\text{qsort}(a, x, z - 1)$, $\text{qsort}(a, z + 1, y)$ を実行する.

図 5.2: クイックソート

定理 5.2. 図 5.2 のアルゴリズム A は整列問題を解く. つまり, 入力 a_1, \dots, a_n の A の出力 $A(a_1, \dots, a_n)$ は, a_1, \dots, a_n の昇順である. また, A の計算時間は $O(n^2)$ である.

証明. まず, アルゴリズムの正当性を示す.

主張 5.3. $x, y \in [n]$ を任意 (ただし $x \leq y$) とする. $\text{qsort}(a, x, y)$ により, $a[x, \dots, y]$ がソートされる. (昇順になる.)

証明. $y - x$ の値についての帰納法により示す. $y - x = 0$ のとき, つまり, $x = y$ であるとき, $a[x, \dots, y]$ は要素が一つの配列となっている. $\text{qsort}(a, x, y)$ のアルゴリズムよりステップ 1 でリターンされることから, $a[x, \dots, y]$ がソートされることは明らかである. $y - x \leq k$ のとき, $\text{qsort}(a, x, y)$ により $a[x, \dots, y]$ がソートされるとする. $y - x = k + 1$ のときの $\text{qsort}(a, x, y)$ の実行を考える. $\text{qsort}(a, x, y)$ のステップ 7 の

実行直前において（このとき $a[z]$ にピボットの値が移ったとして），任意の $\ell < z$ について $a[\ell] < a[z]$ ，更に，任意の $\ell > z$ について $a[\ell] \geq a[z]$ となっている。

問 5.7. この事実（任意の $\ell < z$ について $a[\ell] < a[z]$ ，任意の $\ell > z$ について $a[\ell] \geq a[z]$ ）が成り立つ理由を説明しなさい。

$\text{qsort}(a, x, z-1)$, $\text{qsort}(a, z+1, y)$ の実行を考える． $z-1-x = z-x-1 \leq k$ ，更に， $y-(z+1) = y-z-1 \leq k$ より帰納仮定が適用できるので， $\text{qsort}(a, x, z-1)$ の実行により $a[x, \dots, z-1]$ が， $\text{qsort}(a, z+1, y)$ の実行により $a[z+1, \dots, y]$ がソートされる．以上のことから，ステップ 7 の実行後， $a[x, \dots, y]$ がソートされることが分かる．

■

この主張に $x = 1, y = n$ を適用すればアルゴリズムの正当性が示される．

次に，アルゴリズムの計算時間を見積もる． $\text{qsort}(a, x, y)$ の計算時間が $O((y-x+1)^2)$ であることを示せば十分である．（ $x = 1, y = n$ とすればよいので．）つまり，ある定数 c が存在して $c(y-x+1)^2$ 以下であることを示す．（ c の値は以降で適切に定められる．）これを $y-x$ の値についての帰納法により示す． $y-x = 0$ のとき，つまり， $x = y$ であるとき， $a[x, \dots, y]$ は要素が一つの配列となっている． $\text{qsort}(a, x, y)$ のアルゴリズムよりステップ 1 でリターンされることから，アルゴリズムは定数時間で終了する．よって，その定数時間より c の値を大きくとれば，（ $y-x = 0$ のとき）計算時間が $c(y-x+1)^2$ 以下となることが示される． $y-x \leq k$ のとき， $\text{qsort}(a, x, y)$ の計算時間が $c(y-x+1)^2$ 以下であるとする． $y-x = k+1$ のときの $\text{qsort}(a, x, y)$ の実行を考える． $\text{qsort}(a, x, y)$ のステップ 1 からステップ 6 までの計算時間が $O(y-x+1)$ であることは明らかである．

問 5.8. この事実（計算時間が $O(y-x+1)$ であること）が成り立つ理由を説明しなさい．

つまり，ある定数 c' が存在して $c'(y-x+1)$ 以下である．このとき，一般性を失うことなく $c' \leq c$ であるとする．（もしそうでなければ c の値を c' より大きくとればよい．）ステップ 7 の実行において，（ $(z-1)-x \leq k, y-(z+1) \leq k$ より^{*1}）帰納仮定が適用できることから， $\text{qsort}(a, x, z-1)$ の計算時間が $c(z-x)^2$ ， $\text{qsort}(a, z+1, y)$ の計算時間が $c(y-z)^2$ である．よって， $\text{qsort}(a, x, y)$ 全体の計算時間は，

$$c'(y-x+1) + c(z-x)^2 + c(y-z)^2 = c'(y-x+1) + c((z-x)^2 + (z-y)^2).$$

^{*1} $x \leq z \leq y$ より，それぞれ， $(z-1)-x \leq (y-1)-x = k, y-(z+1) \leq y-(x+1) = k$ ．

問 5.9. 任意の $x, y \in [n]$ ($x \leq y$), 任意の $z \in [n]$ ($x \leq z \leq y$) に対して,

$$(z - x)^2 + (z - y)^2 \leq (y - x)^2$$

この問より, アルゴリズムの計算時間は,

$$\begin{aligned} c'(y - x + 1) + c((z - x)^2 + (z - y)^2) &\leq c'(y - x + 1) + c(y - x)^2 \\ &\leq c((y - x + 1) + (y - x)^2) \quad (\because c' \leq c) \\ &\leq c(y - x + 1)^2 \quad (\because y - x \geq 0) \end{aligned}$$

■

問 5.10. 上の証明では, クイックソートの「最悪の」計算時間を見積もった. どのような入力であるときに計算時間が最悪となるか説明しなさい. また, その場合に計算時間が $O(n^2)$ となる理由を説明しなさい.

問 5.11. クイックソートの「最良の」計算時間は $O(n \log n)$ である. どのような入力であるときに計算時間が最良 (最小) となるか説明しなさい. また, その場合に計算時間が $O(n \log n)$ となる理由を説明しなさい.

5.3 マージソート

図 5.3 に, マージソートのアルゴリズムを示す.

問 5.12. 異なる 7 個の整数の列を具体的にあげ, それに対してマージソートのアルゴリズムを適用させなさい. このとき, その入力によるアルゴリズムの動作を説明すること.

定理 5.3. 図 5.3 のアルゴリズム A は整列問題を解く. つまり, 入力 a_1, \dots, a_n の A の出力 $A(a_1, \dots, a_n)$ は, a_1, \dots, a_n の昇順である. また, A の計算時間は $O(n \log n)$ である.

入力：整数の列 $a_1, \dots, a_n \in \mathbb{Z}$

1. a を大きさ n の配列として、それぞれの $i \in [n]$ について $a[i] = a_i$ とする.
2. $\text{msort}(a, 1, n)$ を実行する.
3. 配列 a の値を（順次）出力する.

$\text{msort}(a, x, y)$

1. $x = y$ であればリターン.
2. $m = \lceil (y + x)/2 \rceil$
3. $\text{msort}(a, x, m - 1)$, $\text{msort}(a, m, y)$ をそれぞれ実行する.
4. b を大きさ $y - x + 1$ の配列とする.
5. $i = x, j = m$ とする.
6. それぞれの整数 $z : 1 \leq z \leq y - x + 1$ について（順次）以下を繰り返す.
 - (a) $i = m$ または $j = y + 1$ の場合、以下のうち一方を実行してステップ 7 へ.
 - $i = m$ であれば $b[z, \dots, y - x + 1] = a[j, \dots, y]$
 - $j = y + 1$ であれば $b[z, \dots, y - x + 1] = a[i, \dots, m]$
 - (b) $a[i] < a[j]$ であれば、 $b[z] = a[i]$, $i++$ とする.
 - (c) $a[i] \geq a[j]$ であれば、 $b[z] = a[j]$, $j++$ とする.
7. $a[x, \dots, y] = b[1, \dots, y - x + 1]$ とする.

図 5.3: マージソート

証明. まず, アルゴリズムの正当性を示す.

主張 5.4. $x, y \in [n]$ を任意（ただし $x \leq y$ ）とする. $\text{msort}(a, x, y)$ により, $a[x, \dots, y]$ がソートされる.（昇順になる.）

証明. $y - x$ の値についての帰納法により示す. $y - x = 0$ のとき, つまり, $x = y$ であるとき, $a[x, \dots, y]$ は要素が一つの配列となっている. $\text{msort}(a, x, y)$ のアルゴリズムよりステップ 1 でリターンされることから, $a[x, \dots, y]$ がソートされることは明らかである. $y - x \leq k$ のとき, $\text{msort}(a, x, y)$ により, $a[x, \dots, y]$ がソートされるとする. $y - x = k + 1$ のときの $\text{msort}(a, x, y)$ の実行を考える. $\text{msort}(a, x, y)$ のステップ 3 の実行において, $m - x + 1 \leq k$, $y - m \leq k$ より帰納仮定が適用できるので, $\text{msort}(a, x, m)$ の実行により $a[x, \dots, m]$ が, $\text{msort}(a, m + 1, y)$ の実行により $a[m + 1, \dots, y]$ がソート

される。その状況のもと、ステップ 6 で示されたように、それぞれの先頭 ($a[i]$ と $a[j]$) から小さい方を (順に) 選んで並べれば、 $a[x, \dots, y]$ がソートされることが分かる。 ■

この主張に $x = 1, y = n$ を適用すればアルゴリズムの正当性が示される。

次に、アルゴリズムの計算時間を見積もる。 $\text{msort}(a, x, y)$ の計算時間が $O((y - x + 1) \log(y - x + 1))$ であることを示せば十分である。 $(x = 1, y = n$ とすればよいので。) つまり、ある定数 c, d が存在して $c(y - x + 1) \log(y - x + 1) + d$ 以下であることを示す。 $(c, d$ の値は以降で適切に定められる。)

注 5.3. $y - x + 1$ の値は、 x から y までに並ぶ整数の個数を意味する。

以降、 $y - x + 1$ の値を n と表記する。この表記では、示すべきことは、 $\text{msort}(a, x, y)$ の計算時間が $cn \log n + d$ 以下であることになる。これを n の値についての帰納法により示す。 $n = 1$ のとき、つまり、 $x = y$ であるとき、 $a[x, \dots, y]$ は要素が一つの配列となっている。 $\text{msort}(a, x, y)$ のアルゴリズムよりステップ 1 でリターンされることから、アルゴリズムは定数時間で終了する。よって、その定数時間より d の値を大きくとれば、 $(n = 1$ のとき) 計算時間が、

$$cn \log n + d = d$$

以下となることが示される*2。 $n \leq k$ のとき、 $\text{msort}(a, x, y)$ の計算時間が $cn \log n + d$ 以下であるとする。 $n = k + 1$ のときの $\text{msort}(a, x, y)$ の実行を考える。ステップ 3 を除いた計算時間 (の合計) が $O(n)$ であることは明らかである。

問 5.13. この事実 (計算時間が $O(n)$ であること) が成り立つ理由を説明しなさい。

つまり、ある定数 c' が存在して $c'n$ 以下である。このとき、一般性を失うことなく、 c, c', d について $(c \log 1.5 - c')n \geq d$ が成り立つとする。(もしそうでなければ、 c を c', d より十分に大きくとればよい。) ステップ 3 の実行において、帰納仮定より、 $\text{msort}(a, x, m - 1)$ の計算時間が $c \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + d$ 、 $\text{msort}(a, m, y)$ の計算時間が $c \lceil n/2 \rceil \log \lceil n/2 \rceil + d$ である。よって、 $\text{msort}(a, x, y)$ 全体の計算時間は、(十分大きい n について)

$$\begin{aligned} & c'n + (c \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + d) + (c \lceil n/2 \rceil \log \lceil n/2 \rceil + d) \\ & \leq c'n + (c(\lfloor n/2 \rfloor + \lceil n/2 \rceil) \log(n/1.5) + 2d) \quad (\because n \text{ が十分大きい}) \\ & = c'n + cn \log(n/1.5) + 2d \\ & = c'n + cn \log n - cn \log 1.5 + 2d \end{aligned}$$

*2 $n = 1$ の場合も成り立つように、定数として (c だけでなく) d も必要である。

$$\begin{aligned}
&= cn \log n + d - ((c \log 1.5 - c')n - d) \\
&\leq cn \log n + d. \quad (\because (c \log 1.5 - c')n - d \geq 0)
\end{aligned}$$

■

5.4 ヒープソート

ヒープの構築

定義 5.1

ラベル付き二分木とは、各頂点に値が付いた根付き二分木である。ヒープとは、以下のようなラベル付き二分木である。

1. 頂点の個数を n としたとき、木の深さは $d = \lfloor \log n \rfloor$ である。 $d \geq 1$ の場合、深さ $d-1$ までは完全二分木であり、深さ d の葉は木の左部分に詰められている。
2. 頂点 v のラベルを $x_v \in \mathbb{R}$ としたとき、任意の親子頂点对 (u, v) について $x_u \geq x_v$ である。

深さが d の任意のヒープを T とする。 T の「頂点の名前」を次のように名付ける。根を 1 とし、深さ $i \in [d]$ の頂点を左から順に $2^i, 2^i + 1, \dots, 2^{i+1} - 1$ とする。

注 5.4. 上の定義において、頂点番号は頂点の「名前」であって、頂点に付けられた「ラベル」のことではない。

例 5.1 (ヒープ). 以下の図は、 $n = 10$ とした場合のヒープの一例である。二分木の頂点は○で表示されている。頂点の名前は○の上に、頂点のラベルは○の中に表示されている。

命題 5.4. T を n 頂点のヒープとする。 $i \in [n]$ を T の任意の頂点とする。 $p, q \in [n]$ を i の子とする。このとき、 $x_i \geq \max\{x_p, x_q\}$ 。

問 5.14. 上の命題を証明しなさい。

命題 5.5. T を n 頂点のヒープとする。任意の $i \in [n]$ について、頂点 i を根とした T の部分木を T_i とする。(つまり、 i から下の部分。) このとき、任意の $i \in [n]$ について、

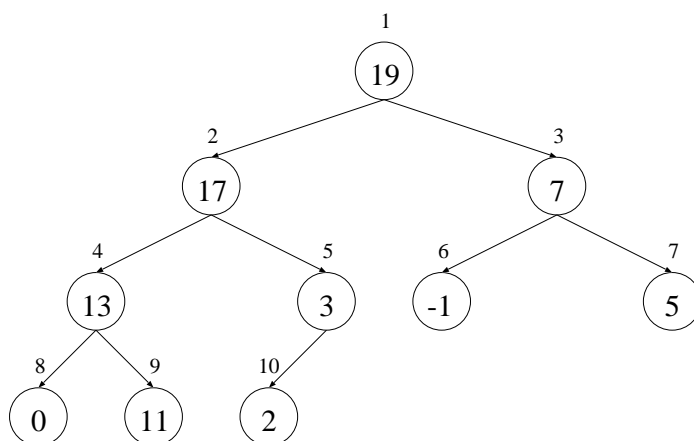


図 5.4: ヒープの例

T_i の頂点のラベルの最大は x_i である.

問 5.15. 上の命題を証明しなさい.

定理 5.6. 図 5.5 のアルゴリズム A によって構築される二分木 T はヒープである. また, A の計算時間は $O(n \log n)$ である.

問 5.16. 異なる 10 個の整数の列を具体的にあげ, それに対してヒープの構築のアルゴリズムを適用させなさい.

問 5.17. ヒープの構築のアルゴリズムにおいて, $\text{mindown}(i)$ のステップ 3 では, どのようなことが行われるか, 簡潔に述べなさい.

証明. まず, アルゴリズムの正当性を示す. 任意の $i \in [n]$ について, 頂点 i を根とした T の部分木を T_i とする.

主張 5.5. $i \in [n]$ を T の任意の頂点とする. i の子を (あれば) $p, q \in [n]$ とする. T_p ,

入力：整数の列 $a_1, \dots, a_n \in \mathbb{Z}$

1. T をヒープの条件 1 を満たす n 頂点の二分木とする。（ラベルはついていない。）
2. 任意の $i \in [n]$ について（順次）以下を繰り返す。
 - T の頂点 i のラベル x_i を a_i とする。（ $x_i = a_i$ 。）
3. 任意の $i \in \{n, n-1, n-2, \dots, 2, 1\}$ について（降順に）以下を繰り返す。
 - $\text{mindown}(i)$ を実行する。

$\text{mindown}(i)$

1. 頂点 i が葉であればリターン。
2. 頂点 i の子を $p, q \in [n]$ とする。
3. x_p, x_q の大小に応じて、以下のうちどちらか一方を実行する。
 - $x_p < x_q$: $x_i < x_q$ なら、 i と q のラベルを交換して $\text{mindown}(q)$ を実行する。
 - $x_p \geq x_q$: $x_i < x_p$ なら、 i と p のラベルを交換して $\text{mindown}(p)$ を実行する。

図 5.5: ヒープの構築

T_q がヒープであれば、 $\text{mindown}(i)$ の実行後の T_i はヒープになる。

証明. T_i の木の深さ d についての帰納法により示す. $d = 0$ のとき、(i は葉であるので) $\text{mindown}(i)$ のアルゴリズムよりステップ 1 でリターンされることから、 T_i がヒープになることは明らかである. $d = k$ のとき、主張が成り立つとする. $d = k + 1$ のとき、 $\text{mindown}(i)$ の実行を考える. 一般性を失うことなく、 $x_p < x_q$ とする. ($x_p \geq x_q$ の場合も同様に示される.) $x_i \geq x_q$ であれば、 $x_i \geq \max\{x_p, x_q\}$ であり、(仮定より) T_p, T_q がヒープであることから、 T_i は (既に) ヒープとなっている. (アルゴリズムはステップ 3 で (何もすることなく) リターンして終了する.) $x_i < x_q$ のとき、 $\text{mindown}(i)$ のアルゴリズムより、 i と q のラベルが交換され $\text{mindown}(q)$ が実行される. $\text{mindown}(q)$ 実行後の頂点 i, p, q のラベルを、それぞれ y_i, y_p, y_q とする. ($\text{mindown}(q)$ 実行前の頂点 i, p, q のラベルが x_i, x_p, x_q .) i と q のラベルが交換されたことから $y_i = x_q$. T_p のラベルは変更されていないことから $y_p = x_p$. これら二つ (と仮定 $x_p < x_q$) より、

$$y_i = x_q > x_p = y_p.$$

よって、(*) $y_i > y_p$. また、 T_q がヒープであったことから、命題 5.5 (と $x_i < x_q$) より、 $x_q \geq y_q$.

問 5.18. この事実 ($x_q \geq y_q$) が成り立つ理由を説明しなさい.

これら二つより,

$$y_i = x_q \geq y_q$$

よって, $(**) y_i \geq y_q$. つまり, $(*)$, $(**)$ より, $y_i \geq \max\{y_p, y_q\}$. また,

- T_p はヒープである. (仮定より)
- T_q はヒープである. (帰納仮定より)

問 5.19. T_p, T_q がヒープである根拠が, それぞれ「仮定より」と「帰納仮定より」と異なる理由を説明しなさい.

以上より, T_i はヒープである. ■

主張 5.6. アルゴリズムのステップ 3 の実行において, 任意の $i \in [n]$ について, $\text{mindown}(i)$ 実行後の T_i はヒープである.

証明. i ($i = \{n, n-1, n-2, \dots, 1\}$) についての (降順の) 帰納法により示す. $i = n$ のとき, 頂点 i は T の葉であることから, $\text{mindown}(i)$ のアルゴリズムよりステップ 1 でリターンされることから, T_i がヒープになることは明らかである. 任意の $i > k$ について主張が成り立つとする. (降順の帰納法.) $i = k$ の $\text{mindown}(i)$ の実行を考える. 頂点 i が葉のとき, ($i = 0$ のときと同様) T_i がヒープになることは明らかである. そうでないとき, 一般性を失うことなく, i は二つの子をもち, それらの子を $p, q \in [n]$ とする. 帰納仮定 (とアルゴリズム) より, T_p, T_q はヒープである.

問 5.20. この事実 (T_p, T_q がヒープになる) が成り立つ理由を説明しなさい.

よって, 主張 5.5 より, T_i はヒープとなる. ■

この主張に $i = n$ を適用すれば, アルゴリズムの正当性が示される.

次に, アルゴリズムの計算時間を見積もる. アルゴリズムのステップ 3 の計算時間が $O(n \log n)$ であることを示せば十分である. ステップ 3 において, 任意の $i \in [n]$ について, $\text{mindown}(i)$ を実行した場合, 再帰関数 mindown の呼び出し回数は, 高々 $\lfloor \log n \rfloor + 1$ である.

問 5.21. この事実（呼び出し回数が高々 $\lfloor \log n \rfloor + 1$ ）が成り立つ理由を説明しなさい。（なぜ、 $+1$ が必要なのか？）

よって、`mindown` のアルゴリズム自体の計算時間は $O(1)$ であることから、アルゴリズム全体の計算時間は、ある定数 c に対して、高々、

$$\sum_{i \in [n]} c \log n = cn \log n = O(n \log n).$$

■

ヒープソート

図 5.6 に、ヒープソートのアルゴリズムを示す。

入力：整数の列 $a_1, \dots, a_n \in \mathbb{Z}$

1. 図 5.5 のアルゴリズムを実行する。（それによって構築されたヒープを T とする。）
2. 任意の $i \in \{n, n-1, n-2, \dots, 2, 1\}$ について（降順に）以下を繰り返す。
 - (a) $b_i = x_1$ とする。
 - (b) $x_1 = x_i$ とする。
 - (c) T から頂点 i を削除して、`mindown(1)` を実行する。
3. b を出力する。

図 5.6: ヒープソート

問 5.22. 異なる 10 個の整数の列を具体的にあげ、それに対してヒープソートのアルゴリズムを適用させなさい。このとき、その入力によるアルゴリズムの動作を説明すること。

定理 5.7. 図 5.6 のアルゴリズム A は整列問題を解く。つまり、入力 a_1, \dots, a_n の A の出力 $A(a_1, \dots, a_n)$ は、 a_1, \dots, a_n の昇順である。また、 A の計算時間は $O(n \log n)$ である。

証明. まず, アルゴリズムの正当性を示す. アルゴリズムのステップ 1 の実行後, T は n 頂点のヒープになっている. 任意の $i \in [n]$ について, アルゴリズムのステップ 2 の i についての実行前のラベル付き二分木を T_i とする. ($T_n = T, T_0 = \emptyset$.)

主張 5.7. 任意の $i \in [n]$ について, T_i はヒープである.

証明. i ($i = \{n, n-1, n-2, \dots, 1\}$) についての (降順の) 帰納法により示す. $i = n$ のときは, T がヒープであることから明らかである. 帰納段階は, 主張 5.5 から証明される*3.

問 5.23. 帰納段階が, どのように主張 5.5 から示されるか説明しなさい. (帰納仮定は何か?)

■

主張 5.8. 任意の $i \in [n]$ について, b_i, b_{i+1}, \dots, b_n は昇順である.

証明. 任意の $i \in [n]$ について, $b_i \leq b_{i+1}$ を示せばよい. 任意の $i \in [n]$ について, T_i の根のラベルが b_i となることから, T_i の根のラベルが T_{i+1} の根のラベル以下であることを示せばよい. これは, T_{i+1} がヒープであり, T_i の i 個のラベルは T_{i+1} の根以外の i 個のラベルと同一であることから示される. ■

この主張に $i = n$ を適用すればアルゴリズムの正当性が示される.

次に, アルゴリズムの計算時間を見積もる. まず, ステップ 1 にかかる計算時間は, 定理 5.6 より, $O(n \log n)$ である. また, ステップ 2 において, それぞれの繰り返しにかかる計算時間は, $\text{mindown}(1)$ にかかるかかる計算時間が $O(\log n)$ である (問 5.21 を参照) ことから, $O(\log n)$ である. 最後に, ステップ 3 にかかる計算時間は $O(n)$ である. 以上より, アルゴリズム全体の計算時間は, ある定数 c, c', c'' に対して, 高々,

$$c \cdot n \log n + n \cdot (c' \log n) + c''n = O(n \log n).$$

■

*3 ここでの T_i と主張 5.5 での T_i は定義が異なる.

第 6 章

走査

ここでは、無向グラフを扱う。

到達可能性問題 (reachability)

- 入力 : グラフ $G = (V, E)$, $s, t \in V$
- 質問 : s から t への経路が存在するか？

6.1 幅優先探索とその応用

図 6.1 に、幅優先探索 (BFS: Breadth-First-Search) のアルゴリズムを示す。

入力 : グラフ $G = (V, E)$, $s, t \in V$

1. s を既訪問, $V \setminus \{s\}$ を未訪問として, Q を s だけからなるキュー (queue) とする.
2. $Q \neq \emptyset$ である限り以下を繰り返す.
 - (a) Q からディキューしてその要素を u とする.
 - (b) N_u のうち未訪問の要素 $U \subseteq N_u$ を既訪問にする.
 - (c) U の頂点を Q にインキューする.
3. t が既訪問なら YES を, そうでないなら NO を出力する.

図 6.1: 幅優先探索

注 6.1. アルゴリズムのステップ 2 でディキューしたとき, $u = t$ となった時点で YES を

出力して終了してもよい．（「プログラムの」には速くなる*1.）

問 6.1. 10 頂点上の適当なグラフ（と始点・終点）を具体的にあげ、それに対して幅優先探索のアルゴリズムを適用させなさい．このとき、その入力によるアルゴリズムの動作を説明すること．

定理 6.1. 幅優先探索のアルゴリズム A は到達可能性問題を解く．また、 A の計算時間は $O(|V| + |E|)$ である．（ G が連結グラフであれば $O(|E|)$ ．）

証明. まず、アルゴリズムの正当性を示す．そのためには以下のことを示せばよい．

$$\begin{aligned} s \text{ から } t \text{ への経路が存在する} & : A(G = (V, E), s, t) = \text{YES} \\ s \text{ から } t \text{ への経路が存在しない} & : A(G = (V, E), s, t) = \text{NO} \end{aligned}$$

$X \subseteq V$ を s から到達可能な頂点の集合、 $Y \subseteq V$ をアルゴリズム終了後に既訪問となっている頂点の集合とする．

主張 6.1. $X = Y$ ．

証明. ($X \subseteq Y$) $x \in X$ を任意とする．（ $x \in Y$ を示せばよい．）これは、 $s = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_k = x$ という経路が存在することを意味する．このとき、 $v_k \in Y$ であることを示せばよい．これを $i \in [k] \cup \{0\}$ についての帰納法により示す． $i = 0$ のとき、 $s \in Y$ は明らかである．次に、 $v_i \in Y$ であるとする．この帰納仮定は、ある時点で（つまり、初めて v_i が既訪問になったとき）インキューされたことを意味する．アルゴリズムより、 $Q \neq \emptyset$ である限りデキューされ続けることから、いずれは（デキューしたとき） $u = v_i$ となる． $v_{i+1} \in N_{v_i}$ より、 v_{i+1} が未訪問である場合には既訪問になる．（つまり、 $v_{i+1} \in Y$ ．）

($X \supseteq Y$) アルゴリズムのステップ 2 の繰り返しを k 回実行されたとする．任意の $i \in [k] \cup \{0\}$ について、 $Y_i \subseteq V$ を、ステップ 2 の第 i 回目の繰り返し終了後、既訪問である頂点の集合とする．（よって、 $Y_0 = \{s\}$, $Y_0 \subseteq Y_1 \subseteq \cdots \subseteq Y_k = Y$ ．）以下、任意の $i \in [k] \cup \{0\}$ について $Y_i \subseteq X$ を示す．これを、ステップ 2 の繰り返し回数 $i \in [k] \cup \{0\}$ についての帰納法により示す．まず、 $i = 0$ のとき、 $Y_0 = \{s\}$ であることから、 $Y_0 \subseteq X$ である．（ $s \in X$ であるので．）次に、 $i = j$ のとき $Y_j \subseteq X$ であるとする．ステップ 2 の $j + 1$ 回目の繰り返しにおいて、デキューされた要素を u として、インキューされ

*1 「アルゴリズム的」という最悪時の計算時間を考えた場合は変わらない．

る頂点の集合を U とする. このとき, $u \in Y_j$, $Y_{j+1} = Y_j \cup U$. よって, $U \subseteq N_u$ より, $U \subseteq X$.

問 6.2. $U \subseteq X$ となる理由を説明しなさい.

この主張と $Y_j \subseteq X$, $Y_{j+1} = Y_j \cup U$ より $Y_{j+1} \subseteq X$. 以上より, $Y_k \subseteq X$ となる. ■

この主張より, X の頂点が既訪問となり $V \setminus X$ の頂点が未訪問となる. これより, アルゴリズムの正当性が示される.

次に, 計算時間を見積もる. ステップ 1 とステップ 3 にかかる計算時間はそれぞれ $O(|V|)$, $O(1)$ である. 以降, ステップ 2 にかかる計算時間を見積もる. ステップ 2 の主な操作は以下である.

- キューの操作 (ディキュー・インキュー).
- N_u の特定 (u の隣接頂点の走査).

それぞれについて, $Q = \emptyset$ になるまでにかかった「合計の」計算時間をも積もる.

主張 6.2. 任意の頂点についてインキューされるのは高々 1 回である. よって, デキューされるのも高々 1 回である.

問 6.3. 上の主張の理由を説明しなさい.

まず, キューの操作について, 一つの頂点につきディキュー・インキューにかかる時間は $O(1)$ であることから, 上の主張より, キューの操作にかかる計算時間は合計で $O(|V|)$ となる. 次に, N_u の特定について, 任意の頂点 u について u がディキューされるのは高々 1 回であり, 一つの頂点 u につき N_u の特定にかかる時間は $O(|N_u|)$ である. (グラフは隣接リストで保持されているので.) よって, 隣接頂点の走査にかかる計算時間の合計は, ある定数 c が存在して高々 $c \cdot \sum_{u \in V} |N_u|$. 命題 3.1 より, これは $c \cdot 2|E| = O(|E|)$ である. 以上より, 幅優先探索の計算時間は $O(|V|) + O(|E|) = O(|V| + |E|)$ となる. (G が連結グラフであれば, 系 3.4 より $O(|E|)$ となる.) ■

BFSの応用

最短経路探索問題 (shortest path search)

- 入力 : グラフ $G = (V, E)$, 始点 $s \in V$, 終点 $t \in V$
- 解 : s から t への最短経路

この問題は、最短経路を出力する「関数問題」である。(到達可能性問題は「決定問題」である。) 図 6.2 に、最短経路探索のアルゴリズムを示す。(各辺の距離が単一である場合のアルゴリズムである。)

入力 : グラフ $G = (V, E)$, $s, t \in V$

1. s を既訪問, $V \setminus \{s\}$ を未訪問として, Q を s だけからなるキュー (queue) とする.
2. 任意の頂点 $u \in V$ について関数 $p(u)$ を未定義とする. (p が最短経路を指し示す.)
3. $Q \neq \emptyset$ である限り以下を繰り返す.
 - (a) Q からディキューしてその要素を u とする.
 - (b) $u = t$ なら関数 p を出力して終了する.
 - (c) N_u のうち未訪問の要素 $U \subseteq N_u$ を既訪問にする.
 - (d) それぞれの $v \in U$ について $p(v) = u$ と定義する.
 - (e) U の頂点を Q にインキューする.
4. NO を出力する.

図 6.2: 最短経路探索

定理 6.2. 最短経路探索のアルゴリズム A は最短経路探索問題を解く. A の計算時間は $O(|V| + |E|)$ である. (G が連結グラフであれば $O(|E|)$.)

6.2 深さ優先探索とその応用

図 6.3 に、深さ優先探索 (DFS: Depth-First-Search) のアルゴリズムを示す.

入力：グラフ $G = (V, E)$, $s, t \in V$

1. s を既訪問, $V \setminus \{s\}$ を未訪問とする.
2. $\text{dfs}(s)$ を実行する.
3. t が既訪問なら YES を, そうでないなら NO を出力する.

$\text{dfs}(u)$

1. それぞれの $v \in N_u$ について以下を実行する.
 - v が未訪問なら v を既訪問にして $\text{dfs}(v)$ を実行する.

図 6.3: 深さ優先探索

注 6.2. $\text{dfs}(u)$ において, $u = t$ であった時点で YES を出力して終了してもよい. (「プログラムの」には速くなる*2.)

問 6.4. 10 頂点上の適当なグラフ (と始点・終点) を具体的にあげ, それに対して深さ優先探索のアルゴリズムを適用させなさい. このとき, その入力によるアルゴリズムの動作を説明すること.

定理 6.3. 深さ優先探索のアルゴリズム A は到達可能性問題を解く. また, A の計算時間は $O(|V| + |E|)$ である. (G が連結グラフであれば $O(|E|)$.)

証明. まず, アルゴリズムの正当性を示す. そのためには以下のことを示せばよい.

$$\begin{array}{ll} s \text{ から } t \text{ への経路が存在する} & : A(G = (V, E), s, t) = \text{YES} \\ s \text{ から } t \text{ への経路が存在しない} & : A(G = (V, E), s, t) = \text{NO} \end{array}$$

$X \subseteq V$ を s から到達可能な頂点の集合, $Y \subseteq V$ をアルゴリズム終了後に既訪問となっている頂点の集合とする.

主張 6.3. $X = Y$

証明. ($X \subseteq Y$) $x \in X$ を任意とする. ($x \in Y$ を示せばよい.) これは, $s = v_0 \rightarrow v_1 \rightarrow$

*2 「アルゴリズム的」という最悪時間を考えた場合は変わらない.

$v_2 \rightarrow \cdots \rightarrow v_k = x$ という経路が存在することを意味する。このとき、 $v_k \in Y$ であることを示せばよい。これを $i \in [k] \cup \{0\}$ についての帰納法により示す。 $i = 0$ のとき、 $s \in Y$ は明らかである。次に、 $v_i \in Y$ であるとする。この帰納仮定は、ある時点で（つまり、初めて v_i が既訪問になったとき） $\text{dfs}(v_i)$ が呼び出されたことを意味する。任意の頂点 u について $\text{dfs}(u)$ は有限時間内に終了することから、 $\text{dfs}(v_i)$ のステップ 1 の繰り返しにおいて、いずれは $(v_{i+1} \in N_{v_i} \text{ より}) v = v_{i+1}$ となり、 v_{i+1} が未訪問である場合には既訪問になる。（つまり、 $v_{i+1} \in Y$ 。）

問 6.5. 任意の頂点 u について $\text{dfs}(u)$ が有限時間内に終了する理由を説明しなさい。

$(X \supseteq Y)$ $\text{dfs}(u)$ が（アルゴリズム全体を通して） k 回呼び出されたとする。任意の $i \in [k]$ について、 $Y_i \subseteq V$ を、第 i 回目の $\text{dfs}(u)$ の呼び出し直前に既訪問である頂点の集合とする。（よって、 $Y_1 = \{s\}$, $Y_1 \subseteq Y_2 \subseteq \cdots \subseteq Y_k = Y$. k 回目の呼び出しで新たに既訪問となる頂点はないことから $Y_k = Y$ 。）以下、任意の $i \in [k]$ について $Y_i \subseteq X$ を示す。これを、 $\text{dfs}(u)$ の呼び出し回数 $i \in [k]$ についての帰納法により示す。まず、 $i = 1$ のとき、 $Y_1 = \{s\}$ であることから、 $Y_1 \subseteq X$ である。（ $s \in X$ であるので。）次に、 $i = j$ のとき $Y_j \subseteq X$ であるとする。第 j 回目の呼び出しが $\text{dfs}(u)$ 、第 $j+1$ 回目の呼び出しが $\text{dfs}(v)$ であったとする。このとき、 $u \in Y_j$, $Y_{j+1} = Y_j \cup \{v\}$. よって、 $v \in N_u$ より、 $v \in X$.

問 6.6. $v \in X$ となる理由を説明しなさい。

この主張と $Y_j \subseteq X$, $Y_{j+1} = Y_j \cup \{v\}$ より $Y_{j+1} \subseteq X$. 以上より、 $Y_k \subseteq X$ となる。■

この主張より、 X の頂点が既訪問となり $V \setminus X$ の頂点が未訪問となる。これより、アルゴリズムの正当性が示される。

次に、計算時間を見積もる。ステップ 1 とステップ 3 にかかる計算時間はそれぞれ $O(|V|)$, $O(1)$ である。以降、ステップ 2 にかかる計算時間を見積もる。

主張 6.4. 任意の頂点 $u \in V$ について $\text{dfs}(u)$ が呼び出されるのは高々 1 回である。

問 6.7. 上の主張の理由を説明しなさい。

$\text{dfs}(u)$ での主な操作は以下である。

- N_u の特定（ u の隣接頂点の走査）。

上の主張より, 任意の頂点 u について $\text{dfs}(u)$ が呼び出されるのは高々 1 回であり, 一つの頂点 u につき N_u の特定にかかる時間は $O(|N_u|)$ である. (グラフは隣接リストで保持されているので.) よって, ステップ 2 にかかる計算時間の合計は, ある定数 c が存在して高々 $c \cdot \sum_{u \in V} |N_u|$. 命題 3.1 より, これは $c \cdot 2|E| = O(|E|)$ となる. 以上より, 深さ優先探索の計算時間は $O(|V|) + O(|E|) = O(|V| + |E|)$ となる. (G が連結グラフであれば, 系 3.4 より $O(|E|)$ となる.) ■

問 6.8. グラフ $G = (V, E)$ が連結である場合, 深さ優先探索において, 再帰関数 $\text{dfs}(u)$ は何回呼び出されるか.

DFS の応用

トポロジカルソート問題 (topological sort)

- 入力: 有向グラフ $G = (V, E)$
- 解: $[n]$ 上の順列 $\sigma: [n] \rightarrow [n]$, つまり, $v_{\sigma(1)}, v_{\sigma(2)}, \dots, v_{\sigma(n)}$ s.t. すべての有向辺 $(v_i, v_j) \in E$ に対して, $\sigma(i) < \sigma(j)$

問の略解

1. アルゴリズムとは

1. • 素数性判定問題
 - 入力：自然数 $n \in \mathbb{N}$
 - 質問： n は素数か？
 - 略.
2. • 最短経路探索問題
 - 入力：グラフ $G = (V, E)$, 始点 $s \in V$, 終点 $t \in V$
 - 解： s から t への最短経路
 - 略.
3. $\exists i \in [n][a_i = x]$ であれば, 繰り返し回数が n 以下となるので.
4. 最悪の計算時間を見積もるので, $\forall i \in [n-1][a_i \neq x]$ かつ $a_n = x$ である入力 $x, (a_1, \dots, a_n)$ の場合の計算時間を見積もる. ($3n+1$ となる.)
5. 情報基礎のテキスト「第6章：整数, ユークリッドの互除法」を参照.
6. $r_{k+1} = 0$.
7. $2\log y + 2$. ($\because 1 \leq r_k \leq \dots \leq r_1/2^{(k-1)/2} \leq y/2^{(k-1)/2}$ より $k \leq 2\log y + 1$.)
8. (およそ) $\log x + \log y$.

2. オーダー表記

1. ともに 0.
2. 1, 3, 5, 7, 9, 11
3. 1, 2, 3, 4, 5, 6

3. データ構造

1. n を頂点の個数としたとき、辺の個数の最大は ${}_nC_2 = n(n-1)/2$ であるので.
2. 任意の辺 $(a, b) \in E$ は, $\sum_{u \in V} |N_u|$ の中で, a と b のちょうど2回数え上げられる.
3. 略.
4. 二つの定理より (一つ目の定理は二つ目の項目より), G が連結なら $|V| - 1 \leq |E|$.
5. 深さを k としたとき, $n = 2^{k+1} - 1$ であることから.

4. 探索

1.
 - YES の場合 : $x = 2$, $(-1, 0, 2, 3, 5, 5, 7)$ に対するアルゴリズムの動作は次のようである. まず, $s = 1, t = 7$ とされ, 繰り返しが実行される. $l = 4$ となり, $a_4 = 3 > x$ より, $s = 1, t = 3$ となる. 次に, $l = 2$ となり, $a_2 = 0 < x$ より, $s = 3, t = 3$ となる. 次に, $l = 3$ となり, $a_3 = 2 = x$ より, YES が出力され終了する.
 - NO の場合 : $x = 4$, $(-1, 0, 2, 3, 5, 5, 7)$ に対するアルゴリズムの動作は次のようである. まず, $s = 1, t = 7$ とされ, 繰り返しが実行される. $l = 4$ となり, $a_4 = 3 < x$ より, $s = 5, t = 7$ となる. 次に, $l = 6$ となり, $a_6 = 5 > x$ より, $s = 5, t = 5$ となる. 次に, $l = 5$ となり, $a_5 = 5 > x$ より, $s = 5, t = 4$ となる. 最後に, $s > t$ となり NO が出力され終了する.
2. $s = t = 1$ なので.
3. $n \leq k - 1$ についての仮定なので.
4. $\exists i \in [l - 1][a_i = x]$.
5. $T(\alpha)$ は, 再帰呼び出し $\text{bsearch}(s, t)$ の実行にかかる時間と, 単一時間ですむ部分にかかる時間の和となる. 単一時間ですむ部分には, $s > t$ の比較演算, l の値を求める演算などがある.
6. YES の場合 $c' < c$. NO の場合 $c' > c$. NO の場合, ステップ 4 の再帰関数呼び出し直後のステップ 1 で ($s > t$ となり) 終了する. よって, $c' = 2c$ としておけば十分である.
7. 帰納法による証明の概略.
 - 【初期段階】 $n = 1$ のとき, $n = 2^k$ の場合 ($k = 0$ のとき) より成り立つ.
 - 【帰納仮定】 任意の自然数 $n \leq k-1$ (ただし, $k \geq 2$) に対して $T(n) \leq c \log n + c'$

であると仮定する.

【帰納段階】 $T(k) \leq c \log k + c'$ が満たされることを示す. 帰納仮定と不等式 (4.1) を用いて以下のように示される.

$$\begin{aligned}
 T(k) &\leq T(\lfloor k/2 \rfloor) + c \quad (\because (4.1)) \\
 &\leq (c \log \lfloor k/2 \rfloor + c') + c \quad (\because \text{帰納仮定}) \\
 &\leq c \log(k/2) + c' + c \\
 &= c(\log k - 1) + c' + c \\
 &= c \log k + c'.
 \end{aligned}$$

5. 整列

1. $(a_1, \dots, a_7) = (5, 7, 2, 9, 5, 0, 3)$ に対するアルゴリズムの動作は次のようである. ステップ 2 において, $i = 1$ のとき, ステップ 2-(a) の各繰り返し後の実行結果 (整数列) は以下のようである.

$$\begin{aligned}
 j = 1: & (5, 7, 2, 9, 5, 0, 3) \\
 2: & (5, 2, 7, 9, 5, 0, 3) \\
 3: & (5, 2, 7, 9, 5, 0, 3) \\
 4: & (5, 2, 7, 5, 9, 0, 3) \\
 5: & (5, 2, 7, 5, 0, 9, 3) \\
 6: & (5, 2, 7, 5, 0, 3, 9)
 \end{aligned}$$

つまり, ステップ 2 の $i = 1$ において, 整数列が $(5, 7, 2, 9, 5, 0, 3)$ から $(5, 2, 7, 5, 0, 3, 9)$ に変わる. 同様にして, ステップ 2 の各 $i \in [n]$ 回目の繰り返し後の実行結果 (整数列) は以下のようである.

$$\begin{aligned}
 i = 1: & (5, 2, 7, 5, 0, 3, 9) \\
 2: & (2, 5, 5, 0, 3, 7, 9) \\
 3: & (2, 5, 0, 3, 5, 7, 9) \\
 4: & (2, 0, 3, 5, 5, 7, 9) \\
 5: & (0, 2, 3, 5, 5, 7, 9) \\
 6: & (0, 2, 3, 5, 5, 7, 9)
 \end{aligned}$$

よって, $(0, 2, 3, 5, 5, 7, 9)$ が出力される.

以降, 問 4.1, 問 5.1 と同じような問題 (具体的な入力に対するアルゴリズムの動作) の解答は省略される.

2. $j = \ell - 1$ のときを考える.....
3. 一つ目は上の主張より, 二つ目は要素数が 1 であることから.
4. 任意の $j \in [n - k]$ について $a[j] \leq a[n - k + 1]$.
5. 一つ目より, $a[1] \leq a[2]$. 二つ目より, $a[2], \dots, a[n]$ が昇順である.
6. 略. (問 5.1 の解答の記述にならうこと.)
7. ステップ 4 でそのように配列 b を構築したため.
8. 各ステップの計算時間が $O(y - x + 1)$ であるから.
9. 与式は, $(z - x)(z - y) \leq 0$ となるため.
10. 入力が大きいく順になっているとき.
11. 再帰呼び出しが常に半分ずつになっていくような入力であるとき.
12. 略.
13. 各ステップの計算時間が $O(n)$ であるから.
14. ヒープの定義より.
15. ヒープの定義より.
16. 略.
17. x_i, x_p, x_q の三つのうち最大のものが (三頂点の中の) 最も上に置かれる.
18. T_q のラベルの最大が x_q であるので.
19. `mindown`(q) を実行したとしているので.
20. $p, q < i$ であるので.
21. T_i の深さが 高々 $\lceil \log n \rceil$ であるので. (葉においても呼び出されるので木の深さ +1 が必要.)
22. 略.
23. T_{k+1} がヒープであるとする. (これが帰納仮定である.) 根の子を根とした T_{k+1} の二つの部分木はそれぞれヒープである. (これが帰納仮定から導かれる事実である.) このことを主張 5.5 に適用すれば, T_k がヒープになることが示される.

6. 走査

1. 略.
2. 帰納仮定より $u \in X$ であり, $U \subseteq N_u \subseteq X$ であるため.
3. インキューされる頂点は既訪問になるため. (インキューされるのは未訪問である頂点だけ.)
4. 略.
5. 未訪問の頂点 u についてだけ, それを既訪問にして `dfs`(u) が呼び出されるので.

(未訪問の頂点は単調に減少していく.)

6. 帰納仮定より $u \in X$ であり, $v \in N_u \subseteq X$ であるため.
7. $\text{dfs}(u)$ が呼び出される直前で u は既訪問になるので. (未訪問の頂点 u にだけ $\text{dfs}(u)$ が呼び出される.)
8. $|V|$ 回.

参考図書

本テキストは，アルゴリズムとデータ構造のごく一部（の初歩的なこと）しか扱っていない．アルゴリズムとデータ構造について更に学びたい学生は，以下の教科書や，そこであげられている参考図書を参照するとよい．

1. Cによるアルゴリズムとデータ構造，茨木俊秀著，オーム社，2014．
2. Algorithms (4th Edition), Robert Sedgewick, Addison-Wesley, 2011. （邦訳：アルゴリズムC：第1巻～第3巻，近代科学社，1996．）
3. Introduction to Algorithms (3rd Edition), Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, 2009. （邦訳：アルゴリズムイントロダクション（第3版），近代科学社，2013．）

索引

B
BFS, 43

D
DFS, 46

R
RAM, 5

あ
入次数, 20
親, 20

か
完全二分木, 20
木, 18
疑似コード, 3
距離, 20
クイックソート, 31
経路, 17
子, 20

さ
最悪時間計算量, 5
次数, 16
指数時間, 12
線形探索, 23

た
多項式時間, 12
単一コスト RAM モデル, 5
単純経路, 17
単純閉路, 17
頂点, 15
出次数, 20

な
二分木, 20
二分探索, 23
根, 20
根付き木, 20

は
葉, 20
パス, 17
幅優先探索, 43
バブルソート, 29

ヒープ, 37
ヒープソート, 37, 41
ピボット, 31
深さ, 20
深さ優先探索, 46
閉路, 17
辺, 15

ま
マージソート, 34
無向グラフ, 15
無向辺, 15

や
有向グラフ, 15
有向辺, 15

ら
ラベル付き二分木, 37
隣接する, 16
連結グラフ, 18
連結している, 18

