

新・

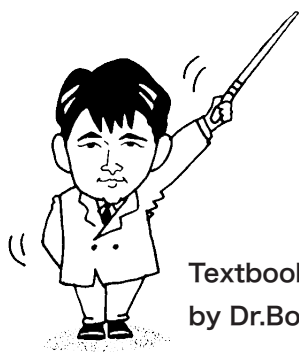
明解

柴田望洋
BohYoh Shibata

Javaで学ぶ アルゴリズムと データ構造 **第3版**

読者特典PDF書籍②

基本情報技術者試験 過去問67問の 徹底学習



Textbook **MEIKAI**

by Dr.BohYoh Shibata

SB Creative

「明解」および「新明解」は、(株)三省堂の登録商標です。

本文中の商品名は、一般に各社の商標または登録商標です。

本文中に、TM、®マークは明記しておりません。

© 2026 本書のプログラムを含むすべての内容は著作権法上の保護を受けています。

著者・発行者の許諾を得ず、無断で複写・複製をすることは禁じられています。

はじめに

『新・明解Javaで学ぶアルゴリズムとデータ構造 第3版』をお買い上げいただき、ありがとうございます。

豊富で的確なサンプルプログラムと見やすく分かりやすい数多くの図表を用いながらの、厳選されたアルゴリズムとデータ構造の学習は進みましたか。

このPDF書籍では、**基本情報技術者試験で過去に出題された67問の徹底学習**を通じて、アルゴリズムとデータ構造に対する知識と理解をさらに深めていきます。

章の構成は、書籍版の本文と同じです。ただし、書籍版では触れられていなかった、配列要素のシフト、フィボナッチ数列、文字列の連結なども学習します。

このPDF書籍が、みなさんの学習のお役に立てば幸いです。

2026年7月

柴田 望洋

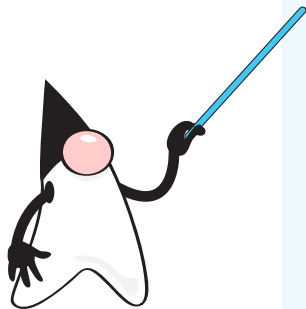
本特典書籍で示されている、基本情報技術者試験の問題の文書や図表は、出題されたオリジナルとおおむね同じとなるように再現されたものです。フォントや句読点の記号などのすべてが完全に再現されているものではありません。

なお、オリジナルの問題文には、（仮名や漢字の）表記ゆれなどの不統一な面があります。本特典書籍の文書にも表記のゆれなどがありますことをご了承ください。

基本情報技術者試験の 解答と解説

基本情報技術者試験の過去問題67問を徹底的に学習します。
本文と同様に、以下に示す全9章で構成されます。

- 1 基本的なアルゴリズム
- 2 基本的なデータ構造
- 3 探索
- 4 スタックとキュー
- 5 再帰的アルゴリズム
- 6 ソート
- 7 文字列探索
- 8 線形リスト
- 9 木構造と2分探索木



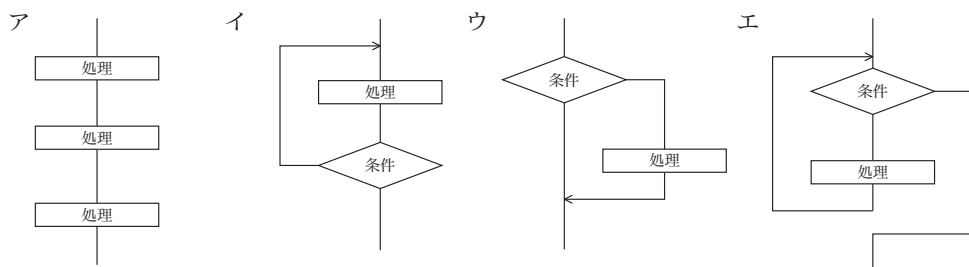
1

基本的なアルゴリズム

1

■ 平成9年度(1997年度)秋期 午前 問37

次のプログラムの制御構造のうち、選択構造はどれか。



■ 解答：ウ

プログラムの制御構造とフローチャート(流れ図)に関する問題です。

ア 順次構造です。三つの“処理”が順に実行されます。

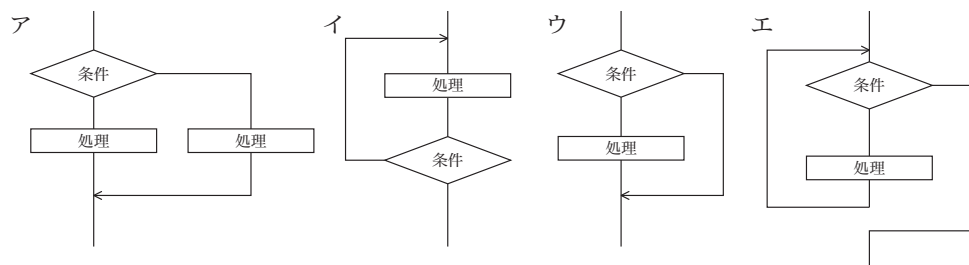
イ 後判定繰返し構造 (do 型の繰返し構造) です。“条件”が成立する限り“処理”の実行が繰り返されます。“条件”の判定が“処理”を行った後であるため、“処理”は少なくとも1回は実行されます。

ウ 選択構造 (双岐選択構造) です。“条件”が成立した場合のみ、“処理”が(1回だけ)実行されます。

エ 前判定繰返し構造 (while 型の繰返し構造) です。“条件”が成立する限り“処理”の実行が繰り返されます。なお、最初の判定で“条件”が成立しなければ、“処理”は1回も実行されません。

■ 平成18年度(2006年度)春期 午前 問36

プログラムの制御構造のうち、while 型の繰返し構造はどれか。



■ 解答：エ

while 型の繰返し構造は、“条件”が成立する限り“処理”の実行が繰り返される**前判定繰返し構造**ですから、正解は**エ**です。最初の判定で“条件”が成立しなければ、“処理”が1回も実行されないという特徴があります。

アと**ウ**は、“条件”が成立するか／しないかで、実行する“処理”が決定する**双岐選択構造**です。**ア**は**if～else～型**で、**ウ**は**if 型**です。

イは、“条件”が成立する限り“処理”の実行が繰り返される点でwhile型の繰返し構造と共通ではあるものの、“条件”の判定のタイミングが“処理”の実行後である点が異なります。この構造は、**後判定繰返し構造**、すなわち、**do 型の繰返し構造 (do～while 型の繰返し構造)**です。この構造の繰返しには、“処理”が必ず1回以上行われるという特徴があります。

■ 平成16年度(2004年度)秋期 午前 問41

プログラムの制御構造に関する記述のうち、適切なものはどれか。

- ア “後判定繰返し”は、繰返し処理の先頭で終了条件の判定を行う。
- イ “双岐選択”は、前の処理に戻るか、次の処理に進むかを選択する。
- ウ “多岐選択”は、二つ以上の処理を並列に行う。
- エ “前判定繰返し”は、繰返し処理の本体を1回も実行しないことがある。

■ 解答：エ

ア 後判定繰返しでは、“終了条件”が成立しない限り“処理”の実行が繰り返されます。ただし、“終了条件”の判定が行われるタイミングは、(繰返し処理の先頭ではなくて)“処理”を行った後、すなわち**末尾**です。

※問題文で“終了条件”という語句が使われていますので**repeat～until 型の繰返し**です。

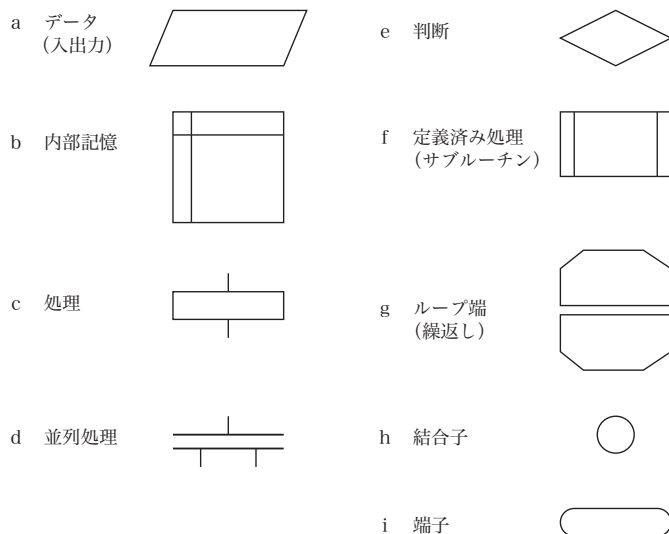
repeat～until 型の繰返し：終了条件が成立しないあいだ“処理”が繰り返される。

do～while 型の繰返し：継続条件が成立するあいだ“処理”が繰り返される。

- イ 双岐選択**では、“条件”が成立するか／しないかで、実行される“処理”が選択的に決定します。処理の流れが進んだり戻ったりすることはありません。
- ウ 多岐選択**では、“条件(多くの場合は、特定の変数や式の値)”に応じて、複数の“処理”のいずれか一つが選択的に実行されます。複数の処理が並列的に行われることはありません。
- エ 前判定繰返し (while 型の繰返し)**では、“条件”が成立する限り“処理”の実行が繰り返されます。なお、最初の判定で“条件”が成立しなければ、“処理”は1回も実行されません。

■ 平成6年度(1994年度)秋期 午前 問41

整構造プログラミング（構造化プログラミング）における基本3構造と呼ばれるものに、最も密接な関係のある流れ図記号の組合せはどれか。



ア a, b, c

イ a, b, d

ウ c, e, g

エ d, e, g

オ f, h, i

■ 解答：ウ

整構造プログラミング＝構造化プログラミング (structured programming) は、1972年に Edsger Wybe Dijkstraによって提唱されたプログラムの作成技法です。

構造化プログラミングでは、

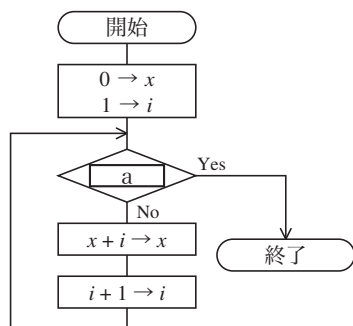
- **順次** (concatenation)
- **選択** (selection)
- **繰返し** (repetition)

の三つの基本制御構造のみを組み合わせるのが基本です。

それぞれに対応するのは、**cの処理** (を複数個並べたもの)、**eの判断**、**gのループ端**ですから、正解は**ウ**です。

■ 平成12年度(2000年度)春期 午前 問16

次の流れ図は、1 から N ($N \geq 1$) までの整数の総和 ($1 + 2 + \dots + N$) を求め、結果を変数 x に入れるアルゴリズムを示している。流れ図中の a に当てはまる式はどれか。



ア $i = N$

イ $i < N$

ウ $i > N$

エ $x > N$

■ 解答：ウ

変数 i の値を 1、2、…、 N とインクリメントしながら、変数 x への i の加算を繰り返すことによって 1 から N までの総和を求めるアルゴリズムです。

変数 x への i の最後の加算を検討します。

最後に加えられる変数 i の値は、 N と同じ値です (1 から 5 までの総和を求めるのであれば、最後に加えられるのは 5 です)。まず、処理 “ $x + i \rightarrow x$ ” によって i が x に加算されて総和が求められます。その後、処理 “ $i + 1 \rightarrow i$ ” によって、変数 i の値は、インクリメントされる結果として $N + 1$ と同じ値へと更新されます (変数 i の値は N を超えます)。

このタイミングで繰返しを終了するので、正解はウの $i > N$ です。

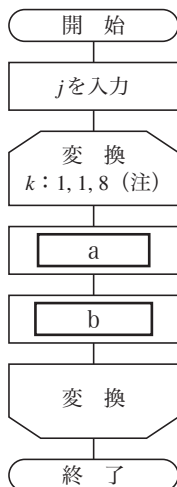
なお、アの $i = N$ だと、 N の加算が行われず、1 から $N - 1$ までの総和が求められることになってしまいます。

2

基本的なデータ構造

■ 令和元年度(2019年度)秋期 午前 問1

次の流れ図は、10進整数 j ($0 < j < 100$) を8桁の2進数に変換する処理を表している。2進数は下位桁から順に、配列の要素 NISHIN(1) から NISHIN(8) に格納される。流れ図の a 及び b に入る処理はどれか。ここで、 $j \div 2$ は j を 2 で割った商の整数部分を、 $j \bmod 2$ は j を 2 で割った余りを表す。



(注) ループ端の繰返し指定は、
変数名：初期値，増分，終値
を示す。

	a	b
ア	$j \leftarrow j \div 2$	$\text{NISHIN}(k) \leftarrow j \bmod 2$
イ	$j \leftarrow j \bmod 2$	$\text{NISHIN}(k) \leftarrow j \div 2$
ウ	$\text{NISHIN}(k) \leftarrow j \div 2$	$j \leftarrow j \bmod 2$
エ	$\text{NISHIN}(k) \leftarrow j \bmod 2$	$j \leftarrow j \div 2$

■ 解答：エ

10進整数から2進整数への基数変換のアルゴリズムに関する問題です。

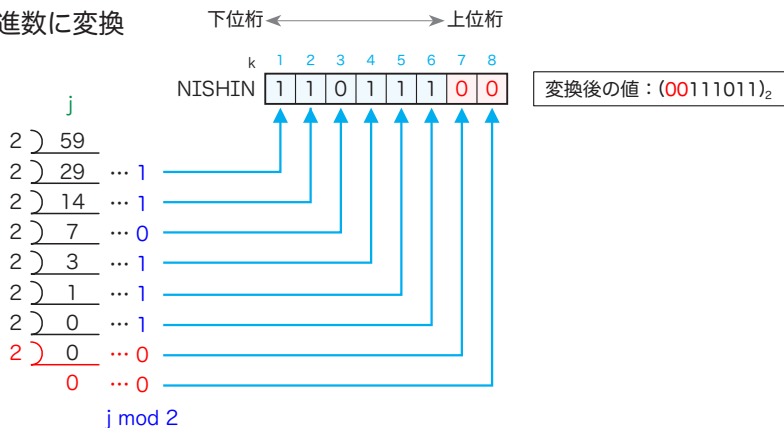
10進整数を2進整数へと基数変換する際は、変換元の10進数(変数 j)を2で割る除算を行って得られる商に対して除算が繰り返されます。なお、除算の際に得られる剰余を逆に並べたものが、変換後の2進数です。

日常の基数変換であれば、除算の過程で求められた商が0になった段階で変換を終了するのに対して、本問では、要素数8の配列への格納が前提であるため、繰返しの回数は8回に固定されています。

10進整数59を2進数に変換する具体例の図を見ながら検討していきましょう。

- 問題文で使われている演算の名称 **div** は除算 (division) に由来し、**mod** は剰余演算 (modulo) に由来します。

59を2進数に変換



フローチャートが示すように、8回の繰返しは、変数 **k** を1、2、…、8とインクリメントすることで行われています。

変数 **j** を2で割る除算で求められた剰余 ($j \bmod 2$) の格納先は、配列 NISHIN の要素 NISHIN(**k**) です。□ a が $\text{NISHIN}(k) \leftarrow j \bmod 2$ であることが分かります。

- たとえば、繰返しの最初の3回は次のように代入が行われます。
- 繰返しの初回で **k** が1のときは、 $59 \bmod 2$ (59を2で割った剰余) の1が NISHIN(1) に代入される。
 - 繰返し2回目で **k** が2のときは、 $29 \bmod 2$ (29を2で割った剰余) の1が NISHIN(2) に代入される。
 - 繰返し3回目で **k** が3のときは、 $14 \bmod 2$ (14を2で割った剰余) の0が NISHIN(3) に代入される。

なお、除算の際に変数 **j** は2で割られます。すなわち、変数 **j** の値は2で割った商へと更新されます。□ b が $j \leftarrow j \div 2$ であることが分かります。

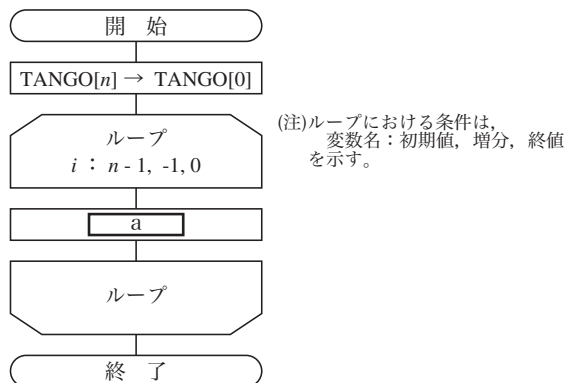
- 変数 **j** の値は、 $59 \Rightarrow 29 \Rightarrow \dots$ と更新されます。

正解は**1**です。

なお、変数 **j** の値が0となった後も、**k** の値が8になるまで繰返しが行われるため、配列 NISHIN の末尾側には0が埋められます。

■ 平成23年度(2011年度)秋期 午前 問7

要素番号が0から始まる配列 TANGO がある。 n 個の単語が TANGO[1] から TANGO[n] に入っている。図は、 n 番目の単語を TANGO[1] に移動するために、TANGO[1] から TANGO[$n-1$] の単語を順に一つずつ後ろにずらして単語表を再構成する流れ図である。 a に入れる処理として正しいものはどれか。



- ア TANGO[i] → TANGO[$i+1$]
- イ TANGO[i] → TANGO[$n-i$]
- ウ TANGO[$i+1$] → TANGO[$n-i$]
- エ TANGO[$n-i$] → TANGO[i]

■ 解答：ア

具体例を示した図を見ながら考えていきましょう。

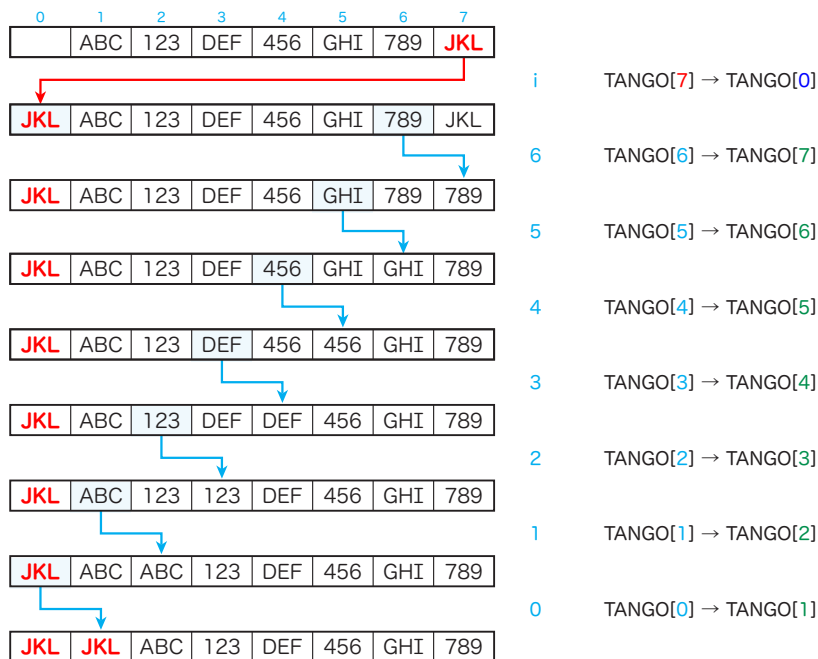
n が7であって、TANGO[1] から TANGO[7] までに、"ABC"、"123"、"DEF"、"456"、"GHI"、"789"、"JKL" が格納されています。

処理の目的は、末尾の単語 "JKL" を TANGO[1] に挿入（挿入前の TANGO[1] ～ TANGO[6] の単語を TANGO[2] ～ TANGO[7] に移動した上で、TANGO[1] に "JKL" を代入）することです。

0	1	2	3	4	5	6	7
	ABC	123	DEF	456	GHI	789	JKL
	JKL	ABC	123	DEF	456	GHI	789

なお、単語が格納されていない TANGO[0] は、自由に使ってよい作業用の要素です。

処理の手順を詳細化した下の図とフローチャートとを対比しながら考えていきます。



“開始”の直後に“TANGO[n] → TANGO[0]”が実行されるため、TANGO[7] すなわち "JKL" が作業用の要素 TANGO[0] に代入されます。

その後で実行される前判定繰返しの“ループ”では、変数 **i** の値が **n - 1** すなわち 6 から 0 ま でデクリメントされます。流れ図の a では、TANGO[**i**] を、一つ後ろ（右隣り）に位置する TANGO[**i** + 1] に代入すればよいことが図から分かるでしょう。

正解は **ア** の “TANGO[**i**] → TANGO[**i** + 1]” です。

2

令和元年度(2019年度)秋期 午前 問9

配列 A が図2の状態のとき、図1の流れ図を実行すると、配列 B が図3の状態になった。
図1のaに入れる操作はどれか。ここで、配列 A 、 B の要素をそれぞれ $A(i, j)$ 、 $B(i, j)$ とする。

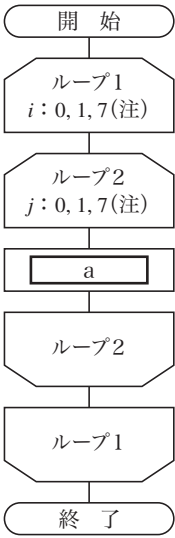


図1 流れ図

	j							
	0	1	2	3	4	5	6	7
0		*	*	*	*	*	*	*
1	*	*						
2	*							
3	*	*	*	*				
4	*							
5	*							
6	*							
7	*							

図2 配列Aの状態

	j							
	0	1	2	3	4	5	6	7
0								
1	*	*	*	*	*	*	*	*
2					*			*
3					*			*
4				*				*
5								*
6								*
7								

図3 実行後の配列Bの状態

(注)ループ端の繰返し指定は、
変数名:初期値, 増分, 終値
を示す。

- ア $B(7-i, 7-j) \leftarrow A(i, j)$
- イ $B(7-j, i) \leftarrow A(i, j)$
- ウ $B(j, 7-j) \leftarrow A(i, j)$
- エ $B(j, 7-i) \leftarrow A(i, j)$

解答：エ

2重の繰返しによって、2次元配列の行と列(縦横)を反転させるアルゴリズムです。
図2と図3を対比すれば、 $A(i, j)$ が $B(j, 7 - i)$ に代入されていることが分かります。したがって、正解はエの $B(j, 7 - i) \leftarrow A(i, j)$ です。

C言語、C++、Javaであれば、コードは次のようになります。

```
for (int i = 0; i <= 7; i++)
    for (int j = 0; j <= 7; j++)
        B[j][7 - i] = A[i][j];
```

Pythonであれば、コードは次のようになります (AとBがlist型であるとします)。

```
for i in range(0, 8):
    for j in range(0, 8):
        B[j][7 - i] = A[i][j]
```

3

探索

3

探索

平成24年度(2012年度)秋期 午前 問3

探索方法とその実行時間のオーダの適切な組合せはどれか。ここで、探索するデータの数を n とし、ハッシュ値が衝突する（同じ値になる）確率は無視できるほど小さいものとする。また、実行時間のオーダが n^2 であるとは、 n 個のデータを処理する時間が cn^2 (c は定数) で抑えられることをいう。

	2分探索	線形探索	ハッシュ探索
ア	$\log_2 n$	n	1
イ	$n \log_2 n$	n	$\log_2 n$
ウ	$n \log_2 n$	n^2	1
エ	n^2	1	n

解答：ア

各探索法の実行時間のオーダは、探索に要する平均処理回数から求められます。

2分探索

探すべきキー値が、整列済み配列の中央に位置する要素と等しいか／大きい／小さいかの判定を行って、探索の対象範囲を（ほぼ）半分ずつに絞り込む処理を繰り返していく探索法です。探索に要する平均比較回数は $\log_2 n$ であり、オーダは $\log_2 n$ です。

線形探索

配列の先頭から順に、探すべきキー値との比較を行っていく探索法です。探索に要する平均比較回数は $n / 2$ であり、オーダは n です。

ハッシュ探索

キー値に対して何らかの変換関数を適用することによって、そのデータを格納する配列内の添字（インデックス）を求める手法です。

添字を求める処理は、キー値をもとにした単純な計算のみで行えます。異なるキーに対するハッシュ値が同一になってしまう衝突が発生しない限り、処理の回数はデータ数に依存することなく1回であり、オーダは1です。

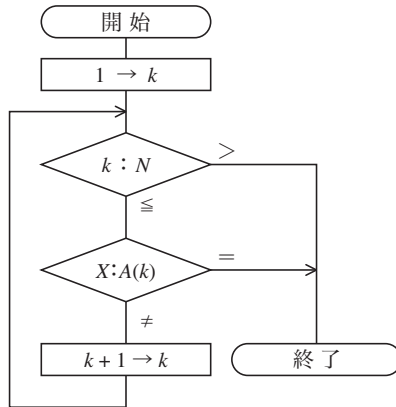
正解はアです。

平成16年度(2004年度)春期 午前 問15

配列 A の 1 番目から N 番目の要素に整数が格納されている ($N > 1$)。次の図は、 X と同じ値が何番目の要素に格納されているかを調べる流れ図である。この流れ図の実行結果として、正しい記述はどれか。

3

探索



ア X と同じ値が配列中にない場合、 k には 1 が設定されている。

イ X と同じ値が配列中にない場合、 k には N が設定されている。

ウ X と同じ値が配列の 1 番目と N 番目の 2 か所にある場合、 k には 1 が設定されている。

エ X と同じ値が配列の 1 番目と N 番目の 2 か所にある場合、 k には N が設定されている。

■ 解答：ウ

与えられたアルゴリズムは、配列の先頭要素から末尾要素へと走査しながら探索のための判定を行う**線形探索**です。 k は現在着目している配列要素の添字（インデックス）であって、“ $k + 1 \rightarrow k$ ”の実行によって、1、2、…、 N とインクリメントされていきます。

まず、 X と同じ値が配列中にない場合（選択肢アとイ）を検討します。末尾要素に着目した最後の判定“ $x : A(k)$ ”では、配列の末尾要素 $A(k)$ すなわち $A(N)$ と X とが等しいかどうか調べられて、その直後に“ $k + 1 \rightarrow k$ ”によってインクリメントされた **k の値は $N + 1$ へと更新されます**（“ $k : N$ ”による大小関係の判定結果が“ $k > N$ ”となるため、走査が終了して**探索失敗**と判定されます）。したがって、アとイのいずれも正解ではありません。

次は、 X と同じ値が配列の 1 番目と N 番目の両方に存在する場合（選択肢ウとエ）を検討します。走査の過程で X と等しい要素を見つけた（“ $X : A(k)$ ”による大小関係の判定結果が“ $X = A(k)$ ”となった）ときに**探索成功**となって走査を終了します。等しい要素に初めて出会った時点で走査が終了するため、 **k の値は 1 となります**。よって、正解はエではなく**ウ**です。

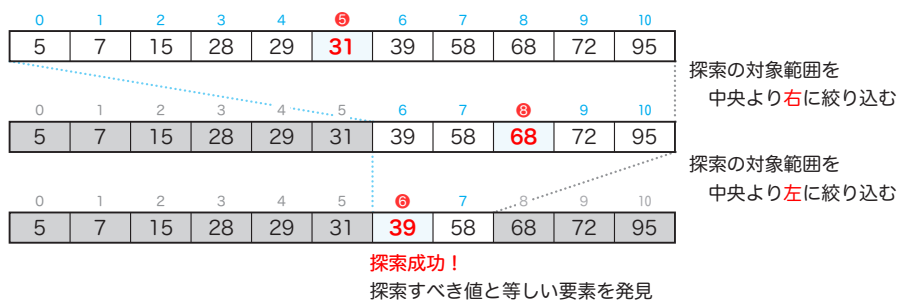
■ 平成17年度(2005年度)秋期 午前 問14

2分探索に関する記述のうち、適切なものはどれか。

- ア 2分探索するデータ列は整列されている必要がある。
- イ 2分探索は線形探索より常に速く探索できる。
- ウ 2分探索は探索をデータ列の先頭から開始する。
- エ n 個のデータの探索に要する比較回数は、 $n \log_2 n$ に比例する。

■ 解答：ア

- ア 2分探索法は、整列済みの配列の中央に位置する要素が、探索する値と等しいか／大きいか／小さいかを判定して、探索対象の範囲を（ほぼ）半分に絞り込んでいく処理を繰り返す探索法です。次に示す配列 $a[0] \sim a[10]$ から 39 を探索する具体例で考えましょう。



中央に位置する $a[5]$ の値は 31 ですから、探索すべき対象範囲は、中央要素より後ろ側の $a[6] \sim a[10]$ に絞られます。このように、中央要素との比較を行って探索すべき範囲を半分に絞っていき、その値が見つかって探索に成功するか、あるいは探索すべき範囲がなくなって探索に失敗するまで繰り返します。

ここでは、データが昇順に並んでいる例を示しましたが、降順に並んでいる場合も同様な探索が可能です。いずれにせよ、**探索対象のデータ列は整列されている必要があります。**

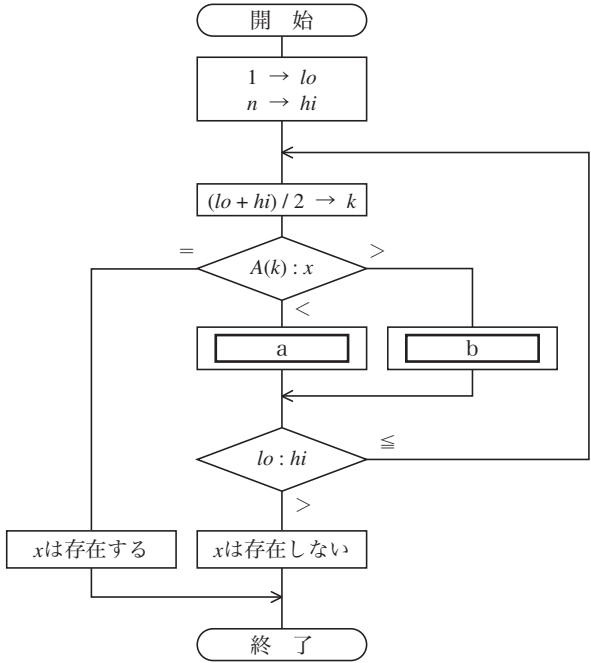
- イ 2分探索の時間計算量のオーダーは $\log_2 n$ であり、線形探索の時間計算量のオーダーは n です。したがって、データ数が極めて少ない場合を除き、2分探索のほうが高速です。ただし、いずれのアルゴリズムも、実際に必要となる計算時間は、データの並びに依存するため、いずれか一方のアルゴリズムが常に高速になることはありません。
- ウ 2分探索はデータ列の中央から探索を開始します。先頭から開始するのは、線形探索です。
- エ n 個のデータの探索に要する比較回数は、平均で $\log_2 n$ 回です。

平成19年度(2007年度)秋期 午前 問14

昇順に整列された n 個のデータが格納されている配列 A がある。流れ図は、2分探索法を用いて配列 A からデータ x を探し出す処理を表している。a、bに入る操作の正しい組合せはどれか。ここで、除算の結果は小数点以下が切り捨てられる。

3

探索



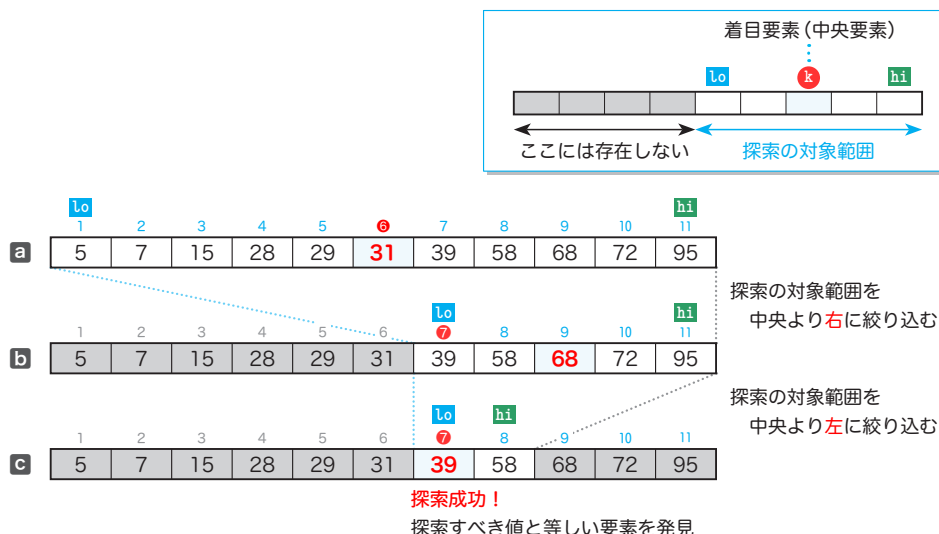
	a	b
ア	$k + 1 \rightarrow hi$	$k - 1 \rightarrow lo$
イ	$k - 1 \rightarrow hi$	$k + 1 \rightarrow lo$
ウ	$k + 1 \rightarrow lo$	$k - 1 \rightarrow hi$
エ	$k - 1 \rightarrow lo$	$k + 1 \rightarrow hi$

■ 解答：ウ

2分探索法は、整列済みの配列の中央に位置する要素が、探索する値と等しいか／大きい／小さいかを判定して、探索対象の範囲を（ほぼ）半分に絞り込んでいく処理を繰り返す探索法です。

与えられた流れ図は、昇順に整列（ソート）済みの要素数 n の配列 A から、 x と等しい値の要素を2分探索によって探すものです。なお、配列の添字（インデックス）は、先頭要素が1で、末尾要素が n です。

配列 $A[1] \sim A[11]$ から39を探索する具体例を示した右ページの図で考えていきましょう。



最初に行われる“ $1 \rightarrow lo$ ”と“ $n \rightarrow hi$ ”の代入によって、 lo は先頭要素の添字1となって、 hi は末尾要素の添字11となります。

a “ $(lo + hi) / 2 \rightarrow k$ ”で求められる k は、探索の対象範囲の中央に位置する要素の添字です。

$(1 + 11) / 2$ で得られた6が k に代入されます (中央要素 $A[6]$ に着目します)。

その後、“ $A(k) : x$ ”によって、探索すべき値 x すなわち39と、中央要素 $A[k]$ すなわち31の大小関係が判定されます。

判定結果は“ $A(k) < x$ ”すなわち $A[6] < 39$ ですから、実行されるのは、フローチャートの **a** です。 $A[lo] \sim A[k]$ の要素は x よりも小さいことが明らかであるため、探索の対象範囲を、中央要素 $A[k]$ より後方＝右側の $A[k + 1] \sim A[hi]$ に絞り込まなければなりません。その絞り込みは、 lo の値を中央要素の一つ後方の $k + 1$ に更新することで行えます。したがって、**a**は“ $k + 1 \rightarrow lo$ ”です。

b プログラムの流れは“ $(lo + hi) / 2 \rightarrow k$ ”に戻ります。 k には $(7 + 11) / 2$ で得られた9が代入されます。その後、“ $A(k) : x$ ”によって、探索すべき値 x すなわち39と、中央要素 $A[k]$ すなわち68の大小関係が判定されます。

判定結果は“ $A(k) > x$ ”すなわち $A[9] > 39$ ですから、実行されるのは、フローチャートの **b** です。 $A[k] \sim A[hi]$ の要素は x よりも大きいことが明らかであるため、探索の対象範囲を、中央要素 $A[k]$ より前方＝左側の $A[lo] \sim A[k - 1]$ に絞り込まなければなりません。その絞り込みは、 hi の値を中央要素の一つ前方の $k - 1$ に更新することで行えます。したがって、**b**は“ $k - 1 \rightarrow hi$ ”です。

正解が**c**であることが確認できました。

※探索に成功する**c**では、フローチャートの **a** と **b** を通らずに“終了”します。

■ 平成11年度(1999年度)春期 午前 問26

次の探索方法のうちで番兵が有効なものはどれか。

ア 2分探索 イ 線形探索 ウ ハッシュ探索 エ 幅優先探索

3

探索

■ 解答：イ

番兵 (sentinel) は、繰返しの終了の判定回数を減少させるために使われる“目印”であって、**線形探索** (linear search) において有効です。よって正解は**イ**です。

詳細を理解していきましょう。

配列の要素を先頭から順に走査する線形探索では、繰返しのたびに、次の二つの終了条件の判定が行われます。

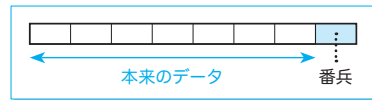
- ① 目的の値が見つからず終端を通り越した(通り越しそうになった)。 ⇨ **探索失敗**
- ② 目的の値と等しい要素を見つけた。 ⇨ **探索成功**

いずれも単純な判定とはいえ、“塵も積もれば山となる”ため、そのコストは決して無視できません。

判定のコストを抑えるのが、番兵を活用する**番兵法** (sentinel method) です。

番兵法で探索行う具体例が、右の図に示されています(配列名をaとします)。

各配列中のa[0]～a[6]の要素が**本来のデータ**です。番兵と呼ばれる末尾のa[7]は、次のように探索するキーと同じ値として格納(追加)されたものです。



a 2を探索 (探索成功)

0	1	2	3	4	5	6	7
6	4	3	2	1	2	8	2

探索する値と等しい要素を発見

b 5を探索 (探索失敗)

0	1	2	3	4	5	6	7
6	4	3	2	1	2	8	5

探索する値と等しい要素を発見
※ただし見つけたのは番兵

図a：2の探索 ⇨ a[7]に番兵2を格納

図b：5の探索 ⇨ a[7]に番兵5を格納

ここで、図bに着目しましょう。目的の5が本来のデータ内に存在しないにもかかわらず、a[7]まで走査した時点で、終了条件②(目的の値と等しい要素を見つけた)が成立します(その結果、走査が終了します)。

すなわち、終了条件①(目的の値が見つからず終端を通り越した)の判定は不要ということです。**番兵が、線形探索における繰返しの終了判定のコストを削減する役割をもつことが分りました。**

2分探索において、データの個数が4倍になると、最大探索回数はどうなるか。

- ア 1回増える。 イ 2回増える。
ウ 約2倍になる。 エ 約4倍になる。

■ 解答：イ

2分探索法は、整列済みの配列の中央に位置する要素が、目的の値と等しいか／大きいのか／小さいかを判定して、探索対象の範囲を（ほぼ）半分に絞り込んでいく処理を繰り返す探索法です。

データの個数を n とすると、探索に要する(キー値と着目要素の値の大小関係の)判定の回数は $\log_2 n$ です。

そのため、データの個数が4倍になった場合、最大探索回数は**2回増える**ことになります。

■ 平成30年度(2018年度)春期 午前 問7

表探索におけるハッシュ法の特徴はどれか。

- ア 2分木を用いる方法の一種である。
- イ 格納場所の衝突が発生しない方法である。
- ウ キーの関数値によって格納場所を決める。
- エ 探索に要する時間は表全体の大きさにほぼ比例する。

■ 解答：ウ

ハッシュ法は、キー値に対して何らかの変換関数を適用することによって、そのデータを格納する位置（レコード番号や配列の添字など）を求める手法です。なお、適用される変換関数は**ハッシュ関数**と呼ばれます。

- ア ハッシュ法では2分木は用いられません。
- イ 異なるキー値から同一のハッシュ値が得られた場合は、格納場所の衝突が発生します。
- ウ キー値をハッシュ関数に適用することによって格納場所を決めます。
- エ 探索に要する時間は表全体の大きさに依存することなく、一定の時間で求められます。

■ 平成20年度(2008年度)秋期 午前 問30

ハッシュ法の説明として、適切なものはどれか。

- ア 関数を用いてレコードのキー値からレコードの格納アドレスを求めることによってアクセスする方法
- イ それぞれのレコードに格納されている次のレコードの格納アドレスを用いることによってアクセスする方法
- ウ レコードのキー値とレコードの格納アドレスの対応表を使ってアクセスする方法
- エ レコードのキー値をレコードの格納アドレスとして直接アクセスする方法

■ 解答：ア

ハッシュ法は、キー値に対して何らかの変換関数を適用することによって、そのデータを格納する位置（レコード番号や配列の添字など）を求める手法です。なお、適用される変換関数は**ハッシュ関数**と呼ばれます。

ア **ハッシュ法**は、ハッシュ関数を用いてレコードのキー値からレコードの格納アドレスを求めることによってアクセスする方法です。

イ それぞれのレコードに格納されている次のレコードの格納アドレスを用いることによってアクセスするのは、**単方向の線形リスト**です。

ウ レコードのキー値とレコードの格納アドレスの対応表を使ってアクセスするのは、**索引編成**の基本的な考え方です。

エ レコードのキー値をレコードの格納アドレスとして直接アクセスするのは、**直接編成**の基本的な考え方です。

■ 平成26年度(2014年度)秋期 午前 問2

0000 ～ 4999 のアドレスをもつハッシュ表があり、レコードのキー値からアドレスに変換するアルゴリズムとして基数変換法を用いる。キー値が 55550 のときのアドレスはどれか。ここでの基数変換法は、キー値を 11 進数とみなし、10 進数に変換した後、下 4 桁に対して 0.5 を乗じた結果（小数点以下は切捨て）をレコードのアドレスとする。

ア 0260

イ 2525

ウ 2775

エ 4405

■ 解答：ア

基数変換法を用いたハッシュ関数の問題です。問題文の指示どおりに変換を行きましょう。

■ キー値を 11 進数とみなして 10 進数に変換

11 進数のキー値 55550 を 10 進数に変換します。

$$5 \times 11^4 + 5 \times 11^3 + 5 \times 11^2 + 5 \times 11^1 + 0 \times 11^0 = 80520$$

■ 変換で得られた 10 進数の下 4 桁に対して 0.5 を乗じる（小数点以下は切捨て）

80520 の下 4 桁の 0520 に 0.5 を乗じます。

$$520 \times 0.5 = 260$$

※アドレスは上位 0 詰めの 4 桁であるため 0260

キー値から得られたアドレスは 0260 ですから、正解は **ア** です。

■平成23年度(2011年度)秋期 午前 問6

次の規則に従って配列の要素 $A[0], A[1], \dots, A[9]$ に正の整数 k を格納する。 k として 16, 43, 73, 24, 85 を順に格納したとき、85 が格納される場所はどれか。ここで、 $x \bmod y$ は x を y で割った剰余を返す。また、配列の要素は全て 0 に初期化されている。

〔規則〕

- (1) $A[k \bmod 10] = 0$ ならば、 $k \rightarrow A[k \bmod 10]$ とする。
- (2) (1) で格納できないとき、 $A[(k + 1) \bmod 10] = 0$ ならば、 $k \rightarrow A[(k + 1) \bmod 10]$ とする。
- (3) (2) で格納できないとき、 $A[(k + 4) \bmod 10] = 0$ ならば、 $k \rightarrow A[(k + 4) \bmod 10]$ とする。

ア $A[3]$ イ $A[5]$ ウ $A[6]$ エ $A[9]$

■解答：エ

ハッシュ法におけるオープンアドレス法の問題です。

正の整数値であるデータの格納先は、規則(1)によって、10 で割った剰余を添字（インデックス）とする要素です。ただし、求められた剰余が同一のデータは格納先が衝突するため、規則(2)と規則(3)によって格納先が調整されます。

なお、配列に格納するデータは“正の整数”であって、配列の全要素の初期値0は、データが格納されていないことを示す目印となっています。

問題文の指示どおりに、与えられたデータを格納していきましょう。

■16を格納する

$16 \bmod 10$ は6です。16を $A[6]$ に格納します。

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	16	0	0	0

■43を格納する

$43 \bmod 10$ は3です。43を $A[3]$ に格納します。

0	1	2	3	4	5	6	7	8	9
0	0	0	43	0	0	16	0	0	0

■73を格納する

$73 \bmod 10$ は3です。 $A[3]$ は既に埋まっていますので、規則(2)を適用します。 $(73 + 1) \bmod 10$ は4です。73を $A[4]$ に格納します。

0	1	2	3	4	5	6	7	8	9
0	0	0	43	73	0	16	0	0	0

■ 24を格納する

24 mod 10は4です。A[4]は既に埋まっていますので、規則(2)を適用します。(24 + 1) mod 10は5です。24をA[5]に格納します。

0	1	2	3	4	5	6	7	8	9
0	0	0	43	73	24	16	0	0	0

■ 85を格納する

85 mod 10は5です。A[5]は既に埋まっていますので、規則(2)を適用します。(85 + 1) mod 10は6です。A[6]も既に埋まっていますので、規則(3)を適用します。(85 + 4) mod 10は9です。85をA[9]に格納します。

0	1	2	3	4	5	6	7	8	9
0	0	0	43	73	24	16	0	0	85

正解が**エ**であることが分かりました。

■ 令和元年度(2019年度)秋期 午前 問10

10進法で5桁の数 $a_1 a_2 a_3 a_4 a_5$ を、ハッシュ法を用いて配列に格納したい。ハッシュ関数を $\text{mod}(a_1 + a_2 + a_3 + a_4 + a_5, 13)$ とし、求めたハッシュ値に対応する位置の配列要素に格納する場合、54321 は配列のどの位置に入るか。ここで、 $\text{mod}(x, 13)$ は、 x を13で割った余りとする。

位置	配列
0	
1	
2	
⋮	⋮
11	
12	

ア 1 イ 2 ウ 7 エ 11

■ 解答：イ

ハッシュ法は、キー値に対して何らかの変換関数を適用することによって、そのデータを格納する位置(レコード番号や配列の添字など)を求める手法です。なお、適用される変換関数は**ハッシュ関数**と呼ばれます。

本問のハッシュ関数は、キー値の各桁の合計を13で割った余りを格納すべき配列の添字とするという単純なものです。

5 + 4 + 3 + 2 + 1は15であって、その15を13で割った剰余は2です。54321は配列の添字**2**の位置に格納されるため、正解は**イ**です。

平成16年度(2004年度)春期 午前 問13

16進数で表される9個のデータ1A、35、3B、54、8E、A1、AF、B2、B3を順にハッシュ表に入れる。ハッシュ値をハッシュ関数 $f(\text{データ}) = \text{mod}(\text{データ}, 8)$ で求めたとき、最初に衝突が起こる(既に表にあるデータと等しいハッシュ値になる)のはどのデータか。ここで、 $\text{mod}(a, b)$ は a を b で割った余りを表す。

- ア 54
- イ A1
- ウ B2
- エ B3

解答：ウ

ハッシュ法は、キー値に対して何らかの変換関数を適用することによって、そのデータを格納する位置(レコード番号や配列の添字など)を求める手法です。なお、適用される変換関数は**ハッシュ関数**と呼ばれます。

本問のハッシュ関数は単純であって、キー値であるデータを8で割った余りをハッシュ値とするものです。表に入れられる9個のデータ(与えられた16進値と、変換した10進値)と、そのハッシュ値をまとめると、次のようになります。

データ	16進数	1A	35	3B	54	8E	A1	AF	B2	B3
	10進数	26	53	59	84	142	161	175	178	179
ハッシュ値		2	5	3	4	6	1	7	2	3

この表は、二つの衝突が発生していることを示しています。

- 8番目のデータ**B2**のハッシュ値**2**が、1番目のデータ**1A**のハッシュ値**2**と衝突。
- 9番目のデータ**B3**のハッシュ値**3**が、3番目のデータ**3B**のハッシュ値**3**と衝突。

題意により、ハッシュ表に“順に”データを入れていく過程で、最初に起こる衝突を選ばなければなりませんので、正解は選択肢**ウ**の**B2**です。

剰余の効率よい求め方

ハッシュ値を求めるために、与えられた16進数を10進数へと変換した上で、8で割った剰余を求めました。もっと効率のよい求め方があります。

まずは、10進数を5で割った剰余について考えましょう。次のように対応しているため、最下位桁のみから剰余を求めることが可能です。

10進数の最下位桁	0	1	2	3	4	5	6	7	8	9
5で割った剰余	0	1	2	3	4	0	1	2	3	4

たとえば、59を5で割った剰余は、最下位桁9に着目した“9→4”の対応から4が得られます。基数10が、割る数である5の2倍と一致するからです。

16進数を8で割った剰余も同様であって、次の対応に示すとおり、剰余は最下位桁のみから求められます。

16進数の最下位桁	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8で割った剰余	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

さきほどの表のように、いったん10進数に変換する必要はありません。ハッシュ値は、与えられた16進数の最下位桁のみから得ることができます。

■平成11年度(1999年度)春期 午前 問31

キー値の分布が1～1,000,000の範囲で一様ランダムであるデータ5個を、大きさ10のハッシュ表に登録する場合、衝突の起こる確率はおよそ幾らか。ここで、ハッシュ値はキー値をハッシュ表の大きさに割った余りを用いる。

ア 0.2 イ 0.5 ウ 0.7 エ 0.9

■解答：ウ

ハッシュ値は、キー値を10で割った剰余ですから、0～9の10種類です。

衝突が発生しない確率は、

$$1 \times (9/10) \times (8/10) \times (7/10) \times (6/10)$$

です。

この値を1から引くと、およそ0.7となりますので、正解はウです。

4

スタックとキュー

4

スタックとキュー

平成18年度(2006年度)春期 午前 問12

空の状態のキューとスタックの二つのデータ構造がある。
右の手続を順に実行した場合、変数 x に代入されるデータ
はどれか。ここで、

データ y をスタックに挿入することを $\text{push}(y)$ 、
スタックからデータを取り出すことを $\text{pop}()$ 、
データ y をキューに挿入することを $\text{enq}(y)$ 、
キューからデータを取り出すことを $\text{deq}()$ 、
とそれぞれ表す。

```
push(a)
push(b)
enq(pop())
enq(c)
push(d)
push(deq())
 $x \leftarrow \text{pop}()$ 
```

- ア a
- イ b
- ウ c
- エ d

■ 解答：イ

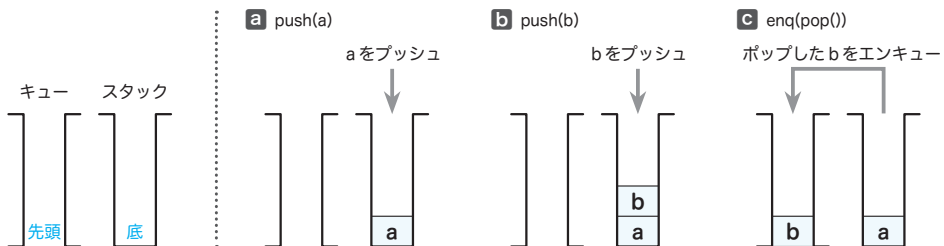
スタック(stack)では、データが**後入れ先出し**= **LIFO**(last-in first-out)で蓄えられます。ちょうど、机の上に重ねられた皿のように、最も上の皿が優先的に取り出されるのと同様です。

キュー(queue)では、データが**先入れ先出し**= **FIFO**(first-in first-out)で蓄えられます。ちょうど、銀行の待ち行列のように、より早く到着して待っているお客さんから、優先的に手続きが行われるのと同様です。

問題文にしたがって、スタックとキューに対して操作を行きましょう。

- ▶ 各図は、左側がキューで右側がスタックです。
- キュー 図の下側が先頭で上側が末尾です。
エンキューは末尾(上側)に押し込まれ、デキューは先頭(下側)から取り出されます。
- スタック 図の下側が底で上側が頂上です。
プッシュは頂上(上側)に積みまれ、ポップも頂上(上側)から取り出されます。

- 図a スタックにaをプッシュします。
- 図b スタックにbをプッシュします。
- 図c スタックからのポップで取り出されたbを、キューにエンキューします。

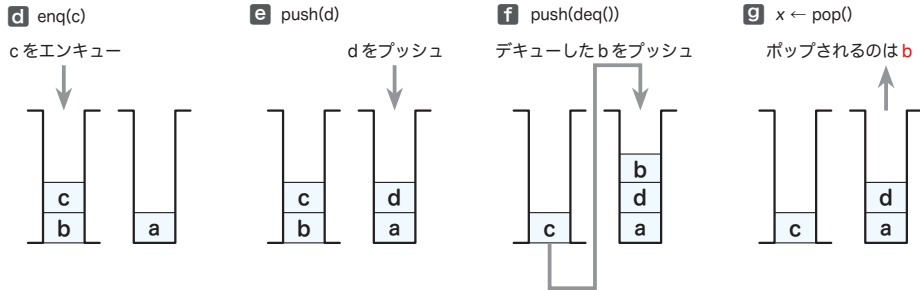


図d キューにcをエンキューします。

図e スタックにdをプッシュします。

図f キューからのデキューで取り出されたbを、スタックにプッシュします。

図g スタックからのポップで取り出されてxに代入される値は**b**であり、正解は**イ**です。



4

スタックとキュー

平成11年度(1999年度)秋期 午前 問13

FIFO (First-In First-Out) の処理に適したデータ構造はどれか。

ア 2分木

イ キュー

ウ スタック

エ ヒープ

■ 解答：イ

キュー (queue) では、データが**先入れ先出し** = **FIFO** (first-in first-out) で蓄えられます。ちょうど、銀行の待ち行列のように、より早く到着して待っているお客さんから、優先的に手続きが行われるのと同様です。

したがって、正解は**イ**です。

平成15年度(2003年度)秋期 午前 問13

スタック操作の特徴を表す用語はどれか。

ア FIFO

イ LIFO

ウ LILO

エ LRU

■ 解答：イ

スタック (stack) では、データが**後入れ先出し** = **LIFO** (last-in first-out) で蓄えられます。ちょうど、机の上に重ねられた皿のように、最も上の皿が優先的に取り出されるのと同様です。

したがって、正解は**イ**です。

■ 平成24年度(2012年度)春期 午前 問6

十分な大きさの配列 A と初期値が 0 の変数 p に対して、関数 $f(x)$ と $g()$ が次のとおり定義されている。配列 A と変数 p は、関数 $f(x)$ と $g()$ だけでアクセス可能である。これらの関数が操作するデータ構造はどれか。

```
function f(x) {
    p = p + 1;
    A[p] = x;
    return None;
}

function g() {
    x = A[p];
    p = p - 1;
    return x;
}
```

- ア キュー イ スタック
ウ ハッシュ エ ヒープ

■ 解答：イ

関数 $f(x)$ は、 p の値を 1 増やした後に $A[p]$ に x を代入して返却する関数であり、関数 $g()$ は、 $A[p]$ の値を取り出した後に p の値を 1 減らした上で取り出した値を返す関数です。

データを入れる操作と取り出す操作の両方が、配列の末尾側で行われています。このことは、二つの関数の操作対象のデータ構造が**スタック** (stack) であることを示しています。

スタックでは、データが**後入れ先出し** = **LIFO** (last-in first-out) で蓄えられます。ちょうど、机の上に重ねられた皿のように、最も上の皿が優先的に取り出されるのと同様です。

データを入れる操作を**プッシュ** (push)、データを取り出す操作を**ポップ** (pop) といいます。

関数 f によって行われる操作がプッシュで、関数 g によって行われる操作がポップです。

正解は**イ**です。

■ 平成29年度(2017年度)秋期 午前 問5

A、B、C、D の順に到着するデータに対して、一つのスタックだけを用いて出力可能なデータ列はどれか。

- ア A、D、B、C イ B、D、A、C ウ C、B、D、A エ D、C、A、B

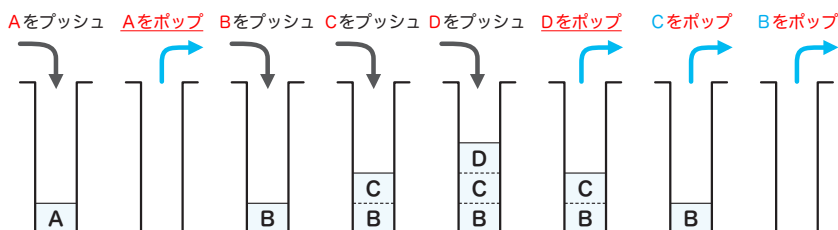
■ 解答：ウ

スタック (stack) では、データが**後入れ先出し** = **LIFO** (last-in first-out) で蓄えられます。ちょうど、机の上に重ねられた皿のように、最も上の皿が優先的に取り出されるのと同様です。

データを入れる操作を**プッシュ** (push)、データを取り出す操作を**ポップ** (pop) といいます。

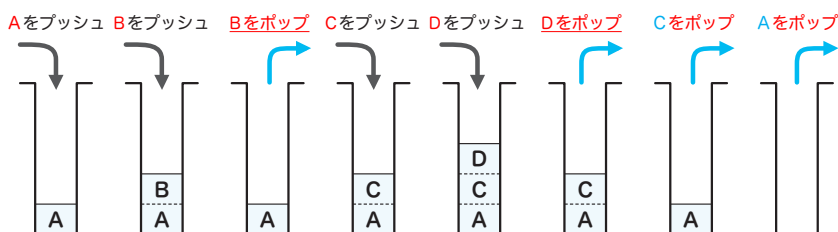
- ア A、D、B、C

最初にAが取り出され、2番目にDが取り出されることから、プッシュとポップは、右ページの図のようになります。Dを取り出した後は、C、Bと取り出されることとなりますので、A、D、C、Bでなければなりません。



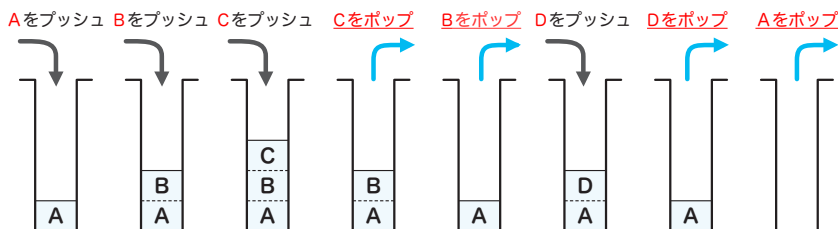
イ B、D、A、C

最初にBが取り出され、2番目にDが取り出されることから、プッシュとポップは、下図のようになります。Dを取り出した後は、C、Aと取り出されることとなりますので、B、D、C、Aでなければなりません。



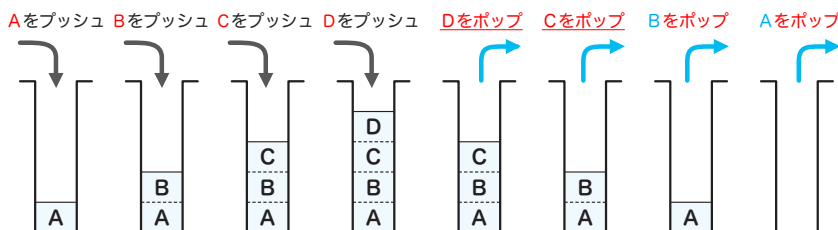
ウ C、B、D、A

プッシュとポップは、下図のように行われます。正解です。



エ D、C、A、B

最初に取り出されるのがDなので、プッシュとポップは、下図のようになります。Dを取り出した後は、C、B、Aと取り出されることとなりますので、D、C、B、Aでなければなりません。



平成30年度(2018年度)秋期 午前 問5

待ち行列に対する操作を、次のとおり定義する。

ENQ n : 待ち行列にデータ n を挿入する。

DEQ : 待ち行列からデータを取り出す。

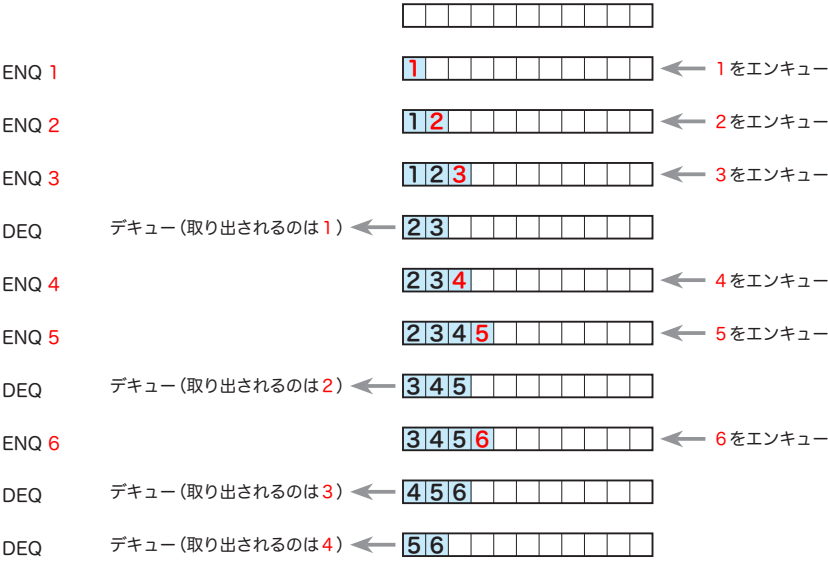
空の待ち行列に対し、ENQ 1, ENQ 2, ENQ 3, DEQ, ENQ 4, ENQ 5, DEQ, ENQ 6, DEQ, DEQ の操作を行った。次に DEQ を行ったとき、取り出される値はどれか。

- ア 1
- イ 2
- ウ 5
- エ 6

解答：ウ

キュー (queue) では、データが**先入れ先出し** = **FIFO** (first-in first-out) で蓄えられます。ちょうど、銀行の待ち行列のように、より早く到着して待っているお客さんから、優先的に手続きが行われるのと同様です。

本問では、キューに対して下図のように操作が行われます。



次にDEQの操作(デキュー)が行われた際に取り出されるのは、キューの先頭に位置する**5**ですから、正解は**ウ**です。

*

より簡単に正解を導く方法があります。

1～6が順にエンキューされるため、デキューでは、それらが順に取り出されます(1回目のデキューでは1、2回目のデキューでは2、…)。5回目にデキューされるのは**5**です。

■ 平成27年度(2015年度)春期 午前 問5

キューに関する記述として、最も適切なものはどれか。

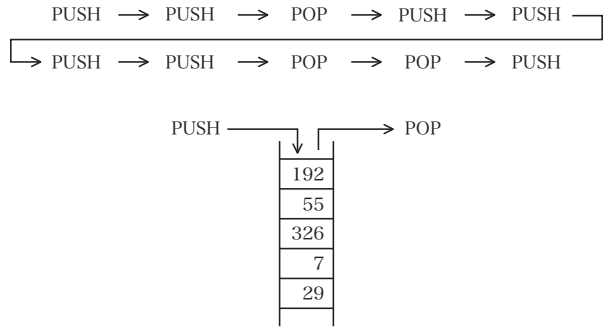
- ア 最後に格納されたデータが最初に取り出される。
- イ 最初に格納されたデータが最初に取り出される。
- ウ 添字を用いて特定のデータを参照する。
- エ 二つ以上のポインタを用いてデータの階層関係を表現する。

■ 解答：イ

- ア 最後に格納されたデータが最初に取り出される**後入れ先出し**= **LIFO** (last-in first-out) のデータ構造は、**スタック**です。
- イ 最初に格納されたデータが最初に取り出される**先入れ先出し**= **FIFO** (first-in first-out) のデータ構造が、**キュー** (queue) です。
ちょうど、銀行の待ち行列のように、より早く到着して待っているお客さんから、優先的に手続きが行われるのと同様です。
- ウ 添字 (インデックス) を用いて特定の位置のデータが参照されるデータ構造は、**配列** (array) です。
- エ 二つ以上のポインタを用いてデータの階層関係が表現されるデータ構造は、**木構造** (tree structure) です。

■ 平成17年度(2005年度)春期 午前 問13

PUSH 命令でスタックにデータを入れ、POP 命令でスタックからデータを取り出す。動作中のプログラムにおいて、ある状態から次の順で10個の命令を実行したとき、スタックの中のデータは図のようになった。1番目の PUSH 命令でスタックに入れたデータはどれか。



- ア 7 イ 29 ウ 55 エ 326

■ 解答：ア

本問では、ある状態のスタックに対して“10個の操作”が行われています。

最後に PUSH されたデータが 192 であることは明らかですから、最初に行われる9個の操作を検討しましょう。

① PUSH → ② PUSH → ③ POP → ④ PUSH → ⑤ PUSH → ⑥ PUSH → ⑦ PUSH → ⑧ POP → ⑨ POP

2箇所の赤文字の操作に着目します。いずれも、PUSHの直後にPOPが行われていますので、“たった今、入れたばかりのデータが取り出される”ことになります。入れられたデータが直後に消え去るため、これらの操作(②と③および⑦と⑧)は除去できます。

すなわち、上の9個の操作は、次に示す5個の操作に置きかえ可能です。

① PUSH → ④ PUSH → ⑤ PUSH → ⑥ PUSH → ⑨ POP

青文字の操作に着目します。PUSHの直後にPOPが行われていますので、“たった今、入れたばかりのデータが取り出される”ことになります。入れられたデータが直後に消え去るため、この操作(⑥と⑨)も除去できます。

すなわち、上の5個の操作は、次に示す3個の操作に置きかえ可能です。

① PUSH → ④ PUSH → ⑤ PUSH

与えられた図と対比すると、⑤では55がPUSHされ、④では326がPUSHされ、①では7がPUSHされていることが分かります。正解はアです。

5

再帰的アルゴリズム

■ 平成29年度(2017年度)秋期 午前 問6

再帰呼出しの説明はどれか。

- ア あらかじめ決められた順番ではなく、起きた事象に応じた処理を行うこと
- イ 関数の中で自分自身を用いた処理を行うこと
- ウ 処理が終了した関数をメモリから消去せず、必要になったとき再び用いること
- エ 処理に失敗したときに、その処理を呼び出す直前の状態に戻すこと

■ 解答：イ

ア 事象駆動(イベントトリブン)

あらかじめ決められた順番ではなく、起きた事象(例：マウスがクリックされた、キーが押下された …)に応じた処理を行います。

イ 再帰呼出し

関数や手続きなどのモジュールが、自分自身のモジュールを呼び出すことです。

ウ 再使用可能(リユーズابل)

処理が終了した関数をメモリから消去せず、必要になったとき再び用います。一度実行した後で、ロードし直さずに再び実行を繰り返しても、正しい結果が得られます。

エ ロールバック

処理に失敗したときに、その処理を呼び出す直前の状態に戻します。

5

■ 平成16年度(2004年度)秋期 午前 問42

再帰的プログラムの特徴として、最も適切なものはどれか。

- ア 一度実行した後、ロードし直さずに再び実行を繰り返しても、正しい結果が得られる。
- イ 実行中に自分自身を呼び出すことができる。
- ウ 主記憶上のどこかのアドレスに配置しても、実行することができる。
- エ 同時に複数のタスクが共有して実行しても、正しい結果が得られる。

■ 解答：イ

ア 再使用可能(リユーザブル)プログラム

いったん主記憶にロードされたら、再ロードせずに実行できる性質のプログラムです。

イ 再帰的(リカーシブ)プログラム

関数や手続きなどのモジュールが、自分自身のモジュールを呼び出すことのできる性質のプログラムです。

ウ 再配置可能(リロケータブル)プログラム

任意のアドレスに配置できるようになっている性質のプログラムです。

エ 再入可能(リエントラント)プログラム

実行中であっても、他のタスクから呼び出すことのできる性質のプログラムです。リアルタイムシステムにおいて、複数のタスクから並行して呼び出される共用ライブラリのプログラムに要求されます。

■ 平成8年度(1996年度)秋期 午前 問17

問題を幾つかの互いに重ならない部分問題に分け、それぞれの解を得ることによって全体の解を求めようとする問題解決の方法はどれか。

- ア オブジェクト指向 イ 再帰呼出し ウ 動的計画法
- エ 二分探索法 オ 分割統治法

■ 解答：オ

問題をいくつかの互いに重ならない部分問題に分け、それぞれの解を得ることによって全体の解を求めようとする問題解決の方法は、**分割統治法**です。

8王妃問題、クイックソート、マージソートなどが、分割統治法の代表的なアルゴリズムです。

■ 令和元年度(2019年度)秋期 午前 問11

自然数 n に対して、次のとおり再帰的に定義される関数 $f(n)$ を考える。 $f(5)$ の値はどれか。

$f(n)$: if $n \leq 1$ then return 1 else return $n + f(n - 1)$

ア 6

イ 9

ウ 15

エ 25

■ 解答：ウ

再帰に慣れていれば、関数 $f(n)$ が、1 から n までの総和を求めることが一目で分かります。

$f(5)$ の値は **15** であり、正解は **ウ** です。

再帰が苦手であれば、トップダウンで計算しましょう。

$$\begin{aligned}
 f(5) &= 5 + f(4) \\
 &= 5 + (4 + f(3)) \\
 &= 5 + (4 + (3 + f(2))) \\
 &= 5 + (4 + (3 + (2 + f(1)))) \\
 &= 5 + (4 + (3 + (2 + 1))) \\
 &= \mathbf{15}
 \end{aligned}$$

■ 平成16年度(2004年度)春期 午前 問14

非負の整数 n に対して次のとおりに定義された関数 $F(n)$ 、 $G(n)$ がある。 $F(5)$ の値は幾らか。

$F(n)$: if $n \leq 1$ then return 1 else return $n \times G(n - 1)$

$G(n)$: if $n = 0$ then return 0 else return $n + F(n - 1)$

ア 50

イ 65

ウ 100

エ 120

■ 解答：イ

間接的な再帰アルゴリズムの問題です。トップダウンで計算しましょう。

$$\begin{aligned}
 F(5) &= 5 \times G(4) \\
 &= 5 \times (4 + F(3)) \\
 &= 5 \times (4 + (3 \times G(2))) \\
 &= 5 \times (4 + (3 \times (2 + F(1)))) \\
 &= 5 \times (4 + (3 \times (2 + 1))) \\
 &= \mathbf{65}
 \end{aligned}$$

$F(5)$ の値は **65** であり、正解は **イ** です。

■ 平成28年度(2016年度)秋期 午前 問7

整数 x, y ($x > y \geq 0$) に対して、次のように定義された関数 $F(x, y)$ がある。 $F(231, 15)$ の値は幾らか。ここで、 $x \bmod y$ は x を y で割った余りである。

$$F(x, y) = \begin{cases} x & (y = 0 \text{ のとき}) \\ F(y, x \bmod y) & (y > 0 \text{ のとき}) \end{cases}$$

ア 2

イ 3

ウ 5

エ 7

5

再帰的アルゴリズム

■ 解答：イ

関数 $F(x, y)$ は、**ユークリッドの互除法**によって整数 x と y の**最大公約数**を求める関数です。

二つの整数値が与えられたとき、大きいほうの値を小さいほうの値で割ります。割り切れて剰余が0となれば小さいほうの値が最大公約数です。割り切れない場合は、小さいほうの値と得られた剰余に対して、割り切れるまで、同じ手続きを再帰的に繰り返します。

次のように計算すると、正解が3すなわちイであることが分かります。

$$F(231, 15) = F(15, 6) \quad \text{※6は231を15で割ったあまり}$$

$$F(15, 6) = F(6, 3) \quad \text{※3は15を6で割ったあまり}$$

$$F(6, 3) = F(3, 0) \quad \text{※0は6を3で割ったあまり}$$

$$F(3, 0) = 3$$

■ 平成26年度(2014年度)秋期 午前 問7

次の関数 $f(n, k)$ がある。 $f(4, 2)$ の値は幾らか。

$$f(n, k) = \begin{cases} 1 & (k = 0), \\ f(n - 1, k - 1) + f(n - 1, k) & (0 < k < n), \\ 1 & (k = n). \end{cases}$$

ア 3

イ 4

ウ 5

エ 6

■ 解答：エ

関数 f は再帰的な関数です。 $f(4, 2)$ の値は、次のように求められます。

$$[A] \quad f(4, 2) = f(3, 1) + f(3, 2)$$

この式の右辺の $f(3, 1)$ と $f(3, 2)$ は、次のように求められます。

$$[B] \quad f(3, 1) = f(2, 0) + f(2, 1) = 1 + f(2, 1) \quad \text{※} k=0 \text{ なので } f(2, 0) \text{ は } 1$$

$$[C] \quad f(3, 2) = f(2, 1) + f(2, 2) = f(2, 1) + 1 \quad \text{※} k=n \text{ なので } f(2, 2) \text{ は } 1$$

ここで、[B] と [C] の両方に含まれる $f(2, 1)$ は、次のように求められます。

$$[D] \quad f(2, 1) = f(1, 0) + f(1, 1) = 1 + 1 = 2$$

※ $k=0$ なので $f(1, 0)$ は1、 $k=n$ なので $f(1, 1)$ は1

したがって、次のように求められます。

$$[C] \quad f(3, 2) = f(2, 1) + 1 = 2 + 1 = 3$$

$$[B] \quad f(3, 1) = 1 + f(2, 1) = 1 + 2 = 3$$

$$[A] \quad f(4, 2) = f(3, 1) + f(3, 2) = 3 + 3 = 6$$

よって、正解は**A**です。

5

■平成28年度(2016年度)春期 午前 問7

n の階乗を再帰的に計算する $F(n)$ の定義において、 a に入れるべき式はどれか。ここで、 n は非負の整数とする。

$$n > 0 \text{ のとき、} F(n) = \boxed{a}$$

$$n = 0 \text{ のとき、} F(n) = 1$$

$$\text{ア} \quad n + F(n - 1)$$

$$\text{イ} \quad n - 1 + F(n)$$

$$\text{ウ} \quad n \times F(n - 1)$$

$$\text{エ} \quad (n - 1) \times F(n)$$

■解答：ウ

たとえば、5の階乗である5!は、

$$5 \text{ の階乗} : 5 \times 4 \times 3 \times 2 \times 1 \quad \text{すなわち } 5 \times 4!$$

で求められますし、この式の右辺で使われている4!は、

$$4 \text{ の階乗} : 4 \times 3 \times 2 \times 1 \quad \text{すなわち } 4 \times 3!$$

で求められます。

問題文のように、 n の階乗を求める手続きを $F(n)$ とすると、上記の二つの式は、次のように表現されます。

$$5 \text{ の階乗} : 5 \times F(4)$$

$$4 \text{ の階乗} : 4 \times F(3)$$

すなわち、 n が0より大きいときの $F(n)$ は、 n と $F(n - 1)$ を掛け合わせた **$n \times F(n - 1)$** であり、正解は**ウ**です。

■ 平成9年度(1997年度)秋期 午前 問5

次に示す関数 $F(K)$ で、 $K = 7$ のときの関数値はどれか。

関数の定義

$$F(0) = 0, F(1) = 1,$$

$$F(K) = F(K - 1) + F(K - 2) \quad (K \geq 2)$$

ア 5

イ 8

ウ 13

エ 21

■ 解答：ウ

フィボナッチ数列に関する問題です。 $F(2)$ から順に求めていきましょう。

$$F(2) = F(1) + F(0) = 1 + 0 = 1$$

$$F(3) = F(2) + F(1) = 1 + 1 = 2$$

$$F(4) = F(3) + F(2) = 2 + 1 = 3$$

$$F(5) = F(4) + F(3) = 3 + 2 = 5$$

$$F(6) = F(5) + F(4) = 5 + 3 = 8$$

$$F(7) = F(6) + F(5) = 8 + 5 = 13$$

正解は**ウ**です。

フィボナッチ数列は、次のように定義された数列です（少しだけ書きかえていますが、問題文で与えられた式と本質的に同じです）。

$$f(0) = 0 \quad \cdots \text{第0項}$$

$$f(1) = 1 \quad \cdots \text{第1項}$$

$$f(k) = f(k - 2) + f(k - 1) \quad (k \geq 2) \quad \cdots \text{第} k \text{項} (k \text{は} 2 \text{以上})$$

1 番目（第0項）の数は0で、2 番目（第1項）の数は1と決められています。

それ以降は、数列の“最後の2値を加えた値”が、次の数となります。たとえば、 $f(2)$ は $f(0)$ と $f(1)$ の和で、 $f(3)$ は $f(1)$ と $f(2)$ の和です。

与えられた式をもとにして再帰呼出しを用いたコーディングを行うと、効率が悪くなります。

たとえば、 $f(7)$ は $f(5)$ と $f(6)$ の和です。まず $f(5)$ を再帰呼出しで求めて、その後で $f(6)$ を再帰呼出しによって求めることになるわけですが、ほとんど同じ計算が（あまりにも数多く）重複してしまいます。

- ▶ $f(7)$ の値は、 $f(5)$ と $f(6)$ の値を求めた上で、それらの和として求めます。左辺 $f(5)$ の値を求めるには $f(3)$ と $f(4)$ の値を求める必要がありますし、右辺 $f(6)$ の値を求めるには（既に求めた） $f(5)$ と $f(4)$ の値を（再び）求める必要があります。

再帰呼出しを用いずに実現された右ページのプログラムを理解していきましょう。

```
// フィボナッチ数列を表示
```

```
//--- フィボナッチ数列を表示 ---//
```

```
void fibonacci(int k) {
    if (k < 0) return;
    int f0 = 0;
    int f1 = 1;
    switch (k) {
        case 0 -> IO.print(f0);
        case 1 -> IO.print(f1);
        default -> {
            IO.print(f0 + " " + f1);
            for (int i = 1; i < k; i++) {
                int f2 = f0 + f1;
                IO.print(" " + f2);
                f0 = f1;
                f1 = f2;
            }
        }
    }
    IO.println();
}

void main() {
    int k = Integer.parseInt(IO.readLine("整数 : "));
    fibonacci(k);
}
```

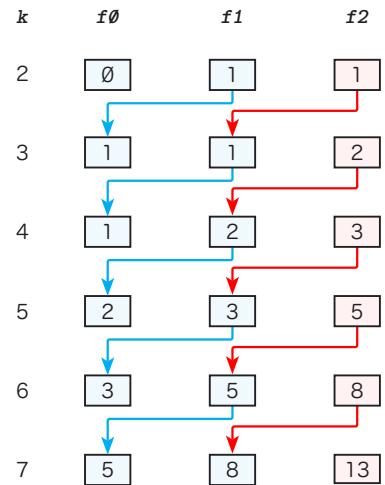
実行例

①	整数 : 0	0
②	整数 : 1	1
③	整数 : 2	0 1 1
④	整数 : 3	0 1 1 2
⑤	整数 : 4	0 1 1 2 3
⑥	整数 : 5	0 1 1 2 3 5
⑦	整数 : 6	0 1 1 2 3 5 8
⑧	整数 : 7	0 1 1 2 3 5 8 13

5

再帰的アルゴリズム

- 1 k が0未満であれば、何も行わずにメソッドを終了します。
- 2 変数 $f0$ を第0項の値 $f(0)$ すなわち0で、変数 $f1$ を第1項の値 $f(1)$ すなわち1で初期化します。
- 3 k が0もしくは1であれば、その値 (変数 $f0$ あるいは $f1$ のいずれかの値) のみを表示します。
- 4 ここに到達するのは k が2以上のときのみです。 $f0$ と $f1$ の値 (スペースで区切った0と1) を表示します。
- 5 $f(2)$ 以降の値を求めて表示する `for` 文です。変数 $f0$ と $f1$ の和によって新しい値を求めた後に、変数 $f1$ の値が $f0$ に代入され、変数 $f2$ の値が $f1$ に代入されています。



もともと第0項の値0で初期化されていた変数 $f0$ と、第1項の値1で初期化されていた変数 $f1$ が、それぞれ“2個前の値 $f(k-2)$ ”と“1個前の値 $f(k-1)$ ”として使われていることが分かるでしょう。図に示された変数の値の変化をトレースして、処理の手順を理解しましょう。

6

ソート

■ 平成13年度(2001年度)春期 午前 問13

n 個のデータをバブルソートを使って整列するとき、データ同士の比較回数は幾らか。

- ア $n \log n$ イ $n(n+1)/4$ ウ $n(n-1)/2$ エ n^2

■ 解答：ウ

バブルソートは**単純交換ソート**とも呼ばれる基本的なソートアルゴリズムです。

隣り合うデータ(要素)を比較して、必要であれば交換を行う処理を配列(の未ソート部)に対して行います(この一連の処理は**パス**と呼ばれます)。最初のパスでは配列全体が未ソート部であるため、要素の比較回数は $n - 1$ 回です。パスのたびに未ソート部の要素数が1個ずつ減っていき、比較回数も1ずつ減っていきます。未ソート部がなくなるまで処理が行われるため、合計 $n - 1$ 回のパスが必要です。

ソート全体でのデータ(要素)の比較回数は、次のようになりますので、正解は**ウ**です。

$$(n - 1) + (n - 2) + \cdots 1 = n(n - 1) / 2$$

■ 平成19年度(2007年度)春期 午前 問14

配列 $A[i]$ ($i = 1, 2, \dots, n$) を、次のアルゴリズムによって整列する。行2～3の処理が初めて終了したとき、必ず実現されている配列の状態はどれか。

[アルゴリズム]

行番号

- 1 i を1から $n - 1$ まで1ずつ増やしながら行2～3を繰り返す
- 2 j を n から $i + 1$ まで減らしながら行3を繰り返す
- 3 もし $A[j] < A[j - 1]$ ならば、 $A[j]$ と $A[j - 1]$ を交換する

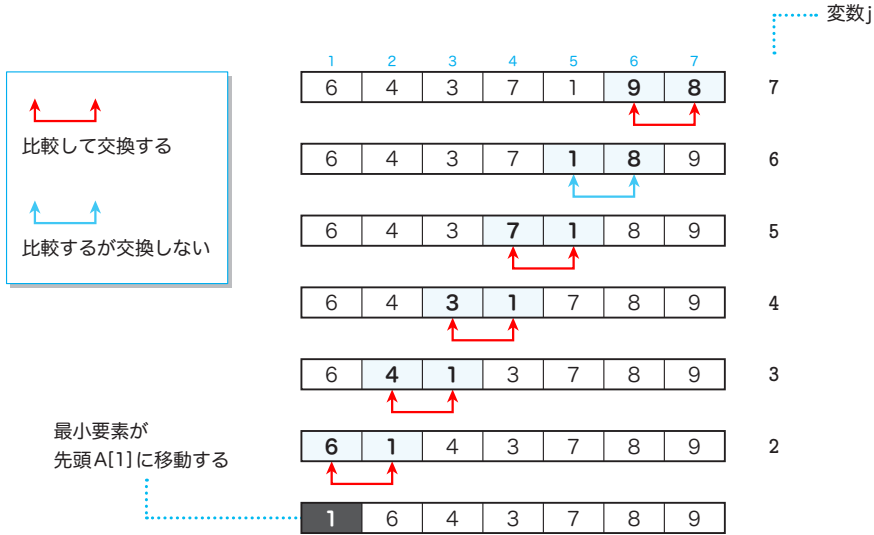
- ア $A[1]$ が最小値になる。 イ $A[1]$ が最大値になる。
 ウ $A[n]$ が最小値になる。 エ $A[n]$ が最大値になる。

■ 解答：ア

与えられたアルゴリズムは、**バブルソート(単純交換ソート)**です。隣り合うデータ(要素)を比較して、必要であれば交換を行うという一連の処理=**パス**が $n - 1$ 回行われます。

本問で問われているのは、整列作業を最後まで行うことではありません。変数 i の値が1であるとき、すなわち“最初の1パス目”が完了した時点でのデータの並びです。

以下に示すのは、 n が7である場合の具体例です（変数 i の値は1です）。



1パス目が終了した時点で、最小値が配列の先頭A[1]に位置しました。正解は**ア**です。

平成14年度(2002年度)秋期 午前 問13

未整列の配列 $A[i]$ ($i = 1, 2, \dots, n$) を、次のアルゴリズムで整列する。要素同士の比較回数のオーダを表す式はどれか。

〔アルゴリズム〕

- (1) $A[1] \sim A[n]$ の中から最小の要素を探し、それを $A[1]$ と交換する。
- (2) $A[2] \sim A[n]$ の中から最小の要素を探し、それを $A[2]$ と交換する。
- (3) 同様に、範囲を狭めながら処理を繰り返す。

ア $O(\log_2 n)$

イ $O(n)$

ウ $O(n \log_2 n)$

エ $O(n^2)$

解答：エ

本問で示されているアルゴリズムは**単純選択ソート**です。配列（の未ソート部）の最小要素を選択した上で、先頭要素と交換する作業を繰り返します。

最小要素を見つけるための比較回数は、(1)では $n - 1$ 回、(2)では $n - 2$ 回と減っていき、最後は1回です。そのため、全体の比較回数は $n(n - 1) / 2$ 回すなわち、 $n^2/2 - n/2$ 回となります。

計算量のオーダは **n^2** ですから、正解は**エ**です。

■ 平成14年度(2002年度)春期 午前 問14

四つの数の並び (4, 1, 3, 2) を、ある整列アルゴリズムに従って昇順に並べ替えたところ、数の入替えは右のとおり行われた。この整列アルゴリズムはどれか。

(1, 4, 3, 2)

(1, 3, 4, 2)

(1, 2, 3, 4)

ア クイックソート イ 選択ソート ウ 挿入ソート エ バブルソート

■ 解答：ウ

次のようにソートが行われています。

(4, 1, 3, 2)

↓ 2番目の要素1に着目し、それより左側の適切な位置(1の前)に**挿入**する。

(1, 4, 3, 2)

↓ 3番目の要素3に着目し、それより左側の適切な位置(1と4のあいだ)に**挿入**する。

(1, 3, 4, 2)

↓ 4番目の要素2に着目し、それより左側の適切な位置(1と3のあいだ)に**挿入**する。

(1, 2, 3, 4)

配列の2番目以降の要素に着目して、その要素より前の適切な位置に**挿入**する処理を繰り返すことによってソートが行われています。

このアルゴリズムは**単純挿入ソート**ですから、正解は**ウ**です。

■ 平成12年度(2000年度)秋期 午前 問13

次の手順はシェルソートによる整列を示している。データ列 “7, 2, 8, 3, 1, 9, 4, 5, 6” を手順 (1) ～ (4) に従って整列すると、手順 (3) を何回繰り返して完了するか。ここで、[] は小数点以下を切り捨てる。

[手順]

- (1) $[データ数 \div 3] \rightarrow H$ とする。
- (2) データ列を互いに H 要素分だけ離れた要素の集まりからなる部分列とし、それぞれの部分列を挿入法を用いて整列する。
- (3) $[H \div 3] \rightarrow H$ とする。
- (4) H が 0 であればデータ列の整列は完了し、0 でなければ (2) に戻る。

ア 2

イ 3

ウ 4

エ 5

■ 解答：ア

シェルソートは、ある間隔で要素を取り出した部分列を整列し、さらに間隔をつめた部分列を取り出して整列する方法です。

本問では、データ数を3で割っていきますので、

$[9 \div 3] \rightarrow 3$ 3ソート (3要素分だけ離れた要素の集まりをソート)

$[3 \div 3] \rightarrow 1$ 1ソート (1要素分だけ離れた要素の集まり=配列全体をソート)

と、3ソートと1ソートの**2回**で全体のソートが完了します。正解は**ア**です。

► 具体的には、次のように行われます。

配列 7, 2, 8, 3, 1, 9, 4, 5, 6

⇓ 3ソート…3要素分離れた要素の集まりをソート (赤、青、緑のそれぞれをソート)

配列 3, 1, 6, 4, 2, 8, 7, 5, 9

⇓ 1ソート…1要素分離れた要素の集まりをソート (配列全体をソート)

配列 1, 2, 3, 4, 5, 6, 7, 8, 9

平成7年度(1995年度)春期 午前 問16

データの整列と併合に関する次の記述中の [] に入れるべき適切な字句の組合せはどれか。

キーの値の小さなものから大きなものへデータを並べることを、[a] に [b] するという。対象とするデータ列が補助記憶装置にある場合、この操作を [c] と呼ぶ。

また、一定の順序に [b] された二つ以上のファイルを一つのファイルに統合することを [d] という。

	a	b	c	d
ア	降順	整列	外部整列	併合
イ	昇順	併合	外部併合	整列
ウ	降順	併合	内部併合	整列
エ	昇順	整列	外部整列	併合
オ	昇順	併合	内部併合	整列

解答：エ

整列／ソート (sorting) は、キー値の大小関係に基づいてデータを並べ替える作業です。
キー値の小さいものから大きいものへと並べるのが**昇順ソート**、キー値の大きいものから小さいものへと並べるのが**降順ソート**です。

ソート対象の全データが主記憶上の配列に格納できる場合は、**内部整列／内部ソート** (internal sorting) と呼ばれるアルゴリズムで実現できます。一方、主記憶に全データを格納できない場合は、ディスクなどの補助記憶装置上のファイルを利用するなどの、複雑なアルゴリズムが必要です。このような整列は、**外部整列／外部ソート** (external sorting) と呼ばれます。

なお、整列済みのデータ群を統合する操作は**併合／マージ** (merging) です (併合／マージの高速性を活用して行われる代表的なソートアルゴリズムが、**マージソート**です)。

したがって正解は**エ**です。

平成9年度(1997年度)秋期 午前 問9

データ全体をある値より大きいデータと小さいか等しいデータに二分する。次に二分されたそれぞれのデータの集まりにこの操作を適用する。これを繰り返してデータ全体を大きさの順に並べ替える整列法はどれか。

- ア クイックソート
- イ バブルソート
- ウ ヒープソート
- エ マージソート

■ 解答：ア

C.A.R.Hoareが考案した**クイックソート** (quick sort) は、その名のとおり、高速なソートアルゴリズムの一つです。

このアルゴリズムは、データ群を、中間的な基準値となる**枢軸** (pivot) より大きいデータ群と小さいデータ群とに分割し、それぞれのデータ群に対して要素が一つになるまで、同じ操作を再帰的に適用するものです。

正解は**ア**です。

■ 平成17年度(2005年度)春期 午前 問14

データの整列方法に関する記述のうち、適切なものはどれか。

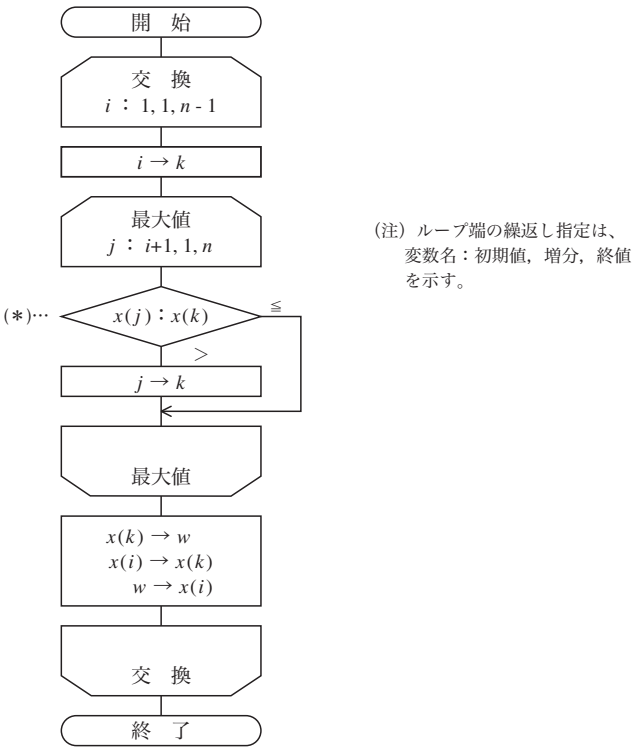
- ア クイックソートでは、ある一定間隔おきに取り出した要素から成る部分列をそれぞれ整列し、更に間隔を詰めて同様の操作を行い、間隔が1になるまでこれを繰り返す。
- イ シェルソートでは、隣り合う要素を比較して、大小の順が逆であれば、それらの要素を入れ替えるという操作を繰り返して行う。
- ウ バブルソートでは、中間的な基準値を決めて、それよりも大きな値を集めた区分と小さな値を集めた区分に要素を振り分ける。次に、それぞれの区分の中で同様な処理を繰り返す。
- エ ヒープソートでは、未整列の部分を順序木に構成し、そこから最大値又は最小値を取り出して既整列の部分に移す。これらの操作を繰り返して、未整列部分を縮めていく。

■ 解答：エ

- ア ある間隔で要素を取り出した部分列を整列し、更に間隔を詰めた部分列を取り出して整列する方法は、クイックソートではなく**シェルソート**です (取り出された部分列の整列に対しては単純挿入ソートのアルゴリズムが適用されます)。
- イ 隣り合う要素を比較して、大小の順が逆であれば、それらの要素を入れ替えるという操作を繰り返して行う方法は、シェルソートではなく**バブルソート** (**単純交換ソート**) です。
- ウ 中間的な基準値を決めて、それより大きな値の要素を集めた区分と小さな値の要素を集めた区分とに振り分け、次にそれぞれの区分の中で同様な処理を繰り返す方法は、バブルソートではなく**クイックソート**です。
- エ **ヒープソート**は、未整列の部分を順序木で表し、そこから最大値もしくは最小値を取り出して既整列の部分に移す操作を繰り返して、未整列部分を縮めていく方法です。

平成14年度(2002年度)春期 午前 問13

次の流れ図は、最大値選択法によって値を大きい順に整列するものである。*印の処理（比較）が実行される回数を表す式はどれか。



- ア $n - 1$ イ $\frac{n(n-1)}{2}$ ウ $\frac{n(n+1)}{2}$ エ n^2

■ 解答：イ

最大値選択法（単純選択ソート）のアルゴリズムに関する問題です。

本アルゴリズムは、“交換”ループの中に、“最大値”ループが入る**2重のループ構造**となっています。問われている*印の処理は、内側の“最大値”ループが繰り返されるたびに（1回ずつ）実行されます。

外側の“交換”ループでは、変数*i*の値が1から*n* - 1まで増えていきますので、繰返しは*n* - 1回です。

内側の“最大値”ループは、変数*j*が*i* + 1から*n*まで増えていきますので、実行される回数は次のようになります。

$i = 1$ のとき $\cdots n - 1$ 回

$i = 2$ のとき $\cdots n - 2$ 回

$\cdots$ 中略 $\cdots$

$i = n - 2$ のとき $\cdots 2$ 回

$i = n - 1$ のとき $\cdots 1$ 回

したがって、合計の回数は次のとおりです。

$$1 + 2 + \cdots + (n - 2) + (n - 1) = n(n - 1) / 2$$

正解は**イ**です。

■平成30年度(2018年度)秋期 午前 問6

クイックソートの処理方法を説明したものはどれか。

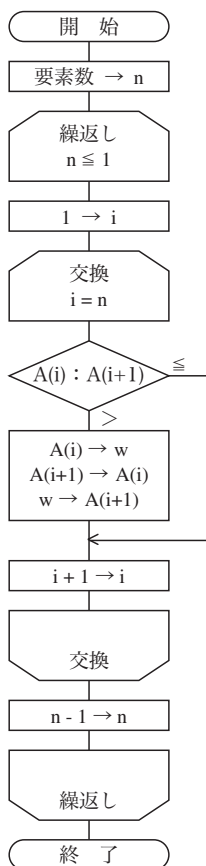
- ア 既に整列済みのデータ列の正しい位置に、データを追加する操作を繰り返していく方法である。
- イ データ中の最小値を求め、次にそれを除いた部分の中から最小値を求める。この操作を繰り返していく方法である。
- ウ 適当な基準値を選び、それよりも小さな値のグループと大きな値のグループにデータを分割する。同様にして、グループの中で基準値を選び、それぞれのグループを分割する。この操作を繰り返していく方法である。
- エ 隣り合ったデータの比較と入替えを繰り返すことによって、小さな値のデータを次第に端の方に移していく方法である。

■解答：ウ

- ア 整列済みのデータ列の正しい位置に、データを追加（挿入）する操作を繰り返していく方法は、**挿入ソート**です。
- イ データ中の最小値を求め、次にそれを除いた部分の中から最小値を求める操作を繰り返していく方法は、**選択ソート**です。
- ウ 適当な基準値を選び、それよりも小さな値のグループと大きな値のグループにデータを分割します。同様にして、グループの中で基準値を選び、それぞれのグループを分割します。この操作を繰り返していく方法が、**クイックソート**です。
- エ 隣り合ったデータの比較と入替えを繰り返すことによって、小さな値のデータを次第に端の方に移していく方法は、**バブルソート（単純交換ソート）**です。

平成8年度(1996年度)秋期 午前 問8

次の流れ図が表す整列アルゴリズムはどれか。



ア クイックソート

イ シェーカーソート

ウ シェルソート

エ 挿入ソート

オ バブルソート

■ 解答：オ

与えられたフローチャートでは、配列内の隣り合う2要素 $A(i)$ と $A(i + 1)$ の大小関係の判定結果に基づいて、(必要な場合に) それらの値が交換されています。

これは**バブルソート**(**単純交換ソート**)の特徴ですから、正解は**オ**です。

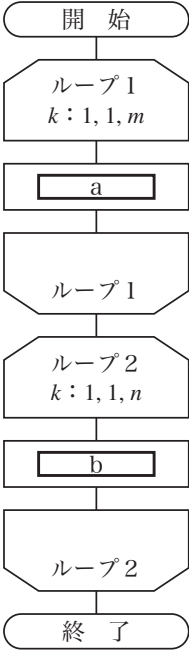
7

文字列探索

平成26年度(2014年度)春期 午前 問8

長さ m 、 n の文字列をそれぞれ格納した配列 X 、 Y がある。図は、配列 X に格納した文字列の後ろに、配列 Y に格納した文字列を連結したものを、配列 Z に格納するアルゴリズムを表す流れ図である。図中の a 、 b に入れる処理として、適切なものはどれか。ここで、1 文字が一つの配列要素に格納されるものとする。

7
文字列探索



(注) ループ端の繰返し指定は、
変数名：初期値，増分，終値
を示す。

	a	b
ア	$X(k) \rightarrow Z(k)$	$Y(k) \rightarrow Z(m + k)$
イ	$X(k) \rightarrow Z(k)$	$Y(k) \rightarrow Z(n + k)$
ウ	$Y(k) \rightarrow Z(k)$	$X(k) \rightarrow Z(m + k)$
エ	$Y(k) \rightarrow Z(k)$	$X(k) \rightarrow Z(n + k)$

■ 解答：ア

文字列を連結するアルゴリズムを問う問題です。ここでは、

配列 X … 文字列 "ABCD" 長さ (文字数) m は 4

配列 Y … 文字列 "123" 長さ (文字数) n は 3

を連結して、配列 Z に文字列 "ABCD123" を格納する具体例で考えていきます。

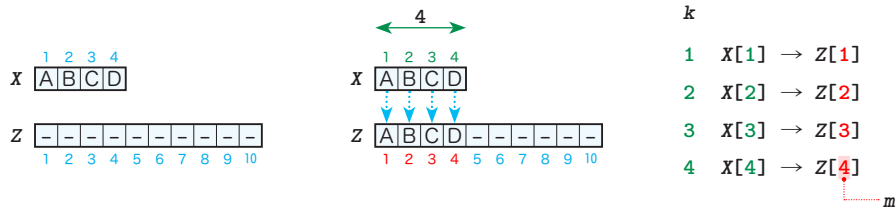
処理は2つのループで行われています。

■ ループ1：配列 Z に文字列 X をコピー

変数 k の値を 1 から m までインクリメントする繰返しです。繰返しの回数が配列 X の長さ (文字数) であることから、下図に示すように、配列 $X(1) \sim X(4)$ が $Z(1) \sim Z(4)$ にコピーされることが分かります。

すなわち、 k が 1 から m すなわち 4 まで増やされながら、 $X(k)$ が $Z(k)$ に代入されますので、

a は " $X(k) \rightarrow Z(k)$ " です。

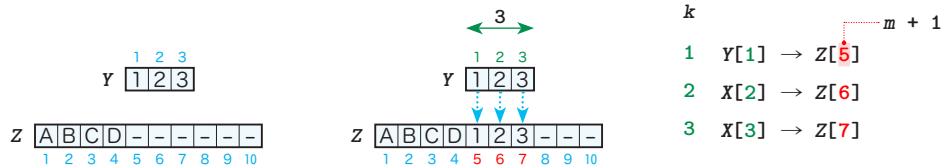


■ ループ2：配列 Z に格納されている文字列に文字列 Y を追加

このループでは、下図に示すように、配列 $Y(1) \sim Y(3)$ が $Z(5) \sim Z(7)$ にコピーされます。

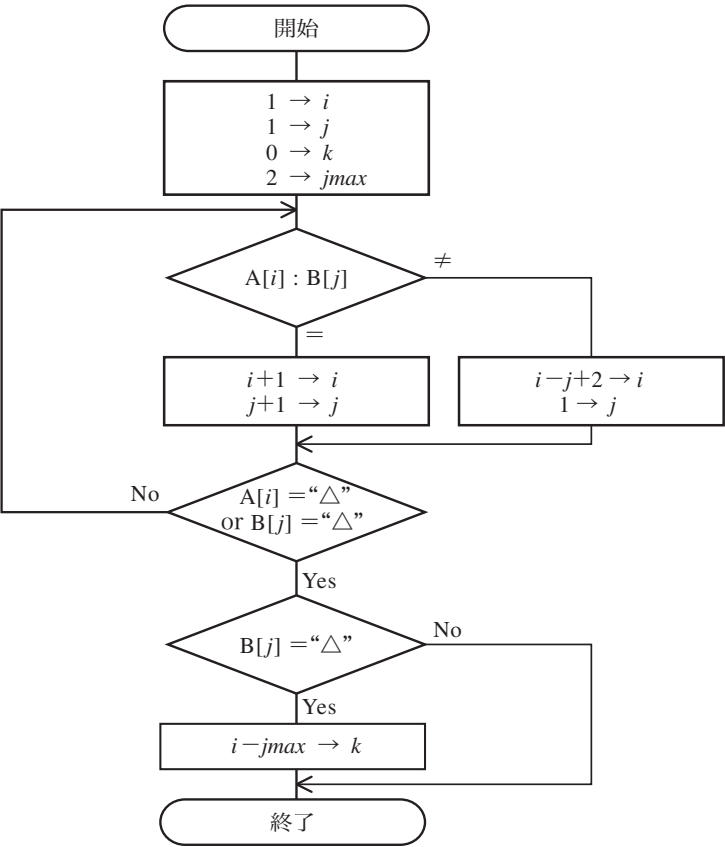
すなわち、 k が 1 から n すなわち 3 まで増やされながら、 $Y(k)$ が $Z(m + k)$ に代入されますので、

b は " $Y(k) \rightarrow Z(m + k)$ " です。



平成19年度(2007年度)春期 午前 問13

文字列 A が “aababx△”、文字列 B が “ab△” であるとき、流れ図の終了時点の k は幾らか。ここで、文字列の先頭の文字を 1 番目と数えるものとし、 $A[i]$ は A の i 番目の文字を、 $B[j]$ は B の j 番目の文字を、“△” は終端を示す文字を表す。



ア 0 イ 1 ウ 2 エ 4

■ 解答：ウ

与えられたフローチャートは、テキスト用文字列 A の中に含まれる、パターン用文字列 B を **力まかせ法** (単純法／素朴法 = brute force method) で探索するアルゴリズムです。

各変数の役割は、次のとおりです。

- 変数 i … テキスト用文字列 A の走査で着目中の文字の添字 (右ページの図の●)。
- 変数 j … パターン用文字列 B の走査で着目中の文字の添字 (右ページの図の●)。
- 変数 k … 探索成功時の文字列 A 側の添字 (失敗時は 0)。
- 変数 $jmax$ … テキスト用文字列 B の長さ (文字数／△は含まない)。

処理の流れを追っていきましょう。

- a** パターン用文字列Bの①の文字を、テキスト用文字列Aの①の文字と重ねて、先頭文字から順に照合していきます。

- 図1 まず着目文字A[1]とB[1]とを比較します。これらの文字は一致しますので、**i**と**j**の両方に1を加えることによって2に更新して、次の文字に着目する準備をします。

- 図2 着目文字A[2]とB[2]とを比較します。これらの文字は一致しません。

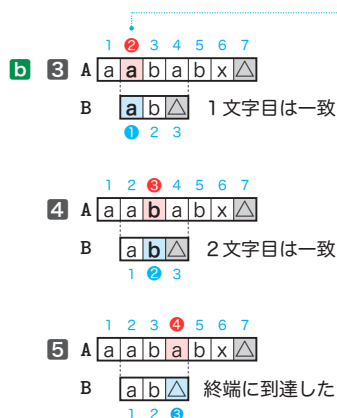
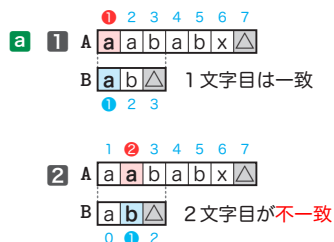
- b** **i**を**i - j + 2**すなわち2に更新して、**j**を1に更新します。その結果、パターンが1文字文後方にずれ、パターンの①の文字が、テキストの②の文字に重ねられました。

- 図3 着目文字A[2]とB[1]とを比較します。これらの文字は一致しますので、**i**と**j**の両方に1を加えることによって、**i**を3に、**j**を2に更新して、次の文字に着目する準備をします。

- 図4 着目文字A[3]とB[2]とを比較します。これらの文字は一致しますので、**i**と**j**の両方に1を加えることによって、**i**を4に、**j**を3に更新して次の文字に着目する準備をします。

- 図5 着目文字B[3]が文字列の終端を表す“△”であるため、繰返しを終了します。
変数**k**に**i - jmax**すなわち4 - 2で得られる2が代入されます。正解は**ウ**です。

※問題に示されているのは、探索に成功する例です。探索に失敗する場合は、 $B[j] = \text{“}\triangle\text{”}$ が成立しないため、変数**k**の値は0になります。



テキスト側の先頭位置のインデックス2は最後の図5で照合した位置である④から文字列の長さ2を引くことで得られる

8

線形リスト

■ 平成21年度(2009年度)春期 午前 問6

配列と比較した場合の連結リストの特徴に関する記述として、適切なものはどれか。

- ア 要素を更新する場合、ポインタを順番にたどるだけなので、処理時間は短い。
- イ 要素を削除する場合、削除した要素から後ろにあるすべての要素を前に移動するので、処理時間は長い。
- ウ 要素を参照する場合、ランダムにアクセスできるので、処理時間は短い。
- エ 要素を挿入する場合、数個のポインタを書き換えるだけなので、処理時間は短い。

■ 解答：エ

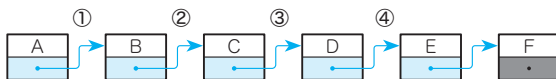
連結リスト（単方向の線形リスト）は、各ノードが、**後続ポインタ**（後続ノードを指すポインタ）をもつデータ構造です。先頭ノードからポインタをたぐることによって、リスト上に登録されているすべてのデータを取り出すことができますようになっています。

ア 要素の更新は、次の手順で行われます。

- ① 更新すべき要素を見つける（先頭から順に後続ポインタを一つずつたぐっていく）。
- ② 見つけた要素の値を更新する。

更新すべき要素を見つけるための①が必要であるため、**処理時間は長くなります**（ランダムに要素をアクセスできる配列であれば、処理時間は短くなります）。

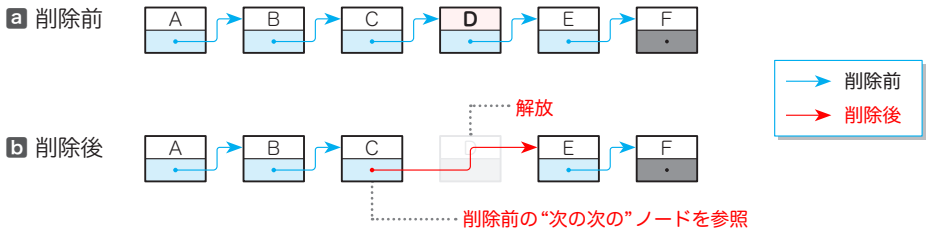
下の図の例でノードEのデータを更新する場合を考えましょう。まず、先頭ノードAに含まれる後続ポインタをたぐってノードBに着目します。次に、ノードBに含まれる後続ポインタをたぐってノードBに着目します。…といった具合で、後続ポインタを4回たぐることによって、ようやくEに到達します。到達した後に、Eのデータの更新作業を行います。



イ 要素を削除する処理は、1個から数個のポインタを書きかえるだけで実現できるため、**処理時間は短くなります**（処理時間が長くなるのは、連結リストではなく配列です。削除した要素から後ろにあるすべての要素を前に移動する必要があるからです）。

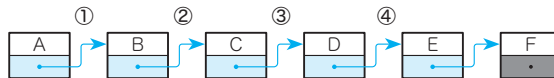
右ページの図の例でノードDを削除する場合を考えましょう。ノードCに含まれる後続ポインタがノードEを指すように更新するだけで、削除処理は終了します。

※どこからも指されなくなったDは、リストから切り離されます。



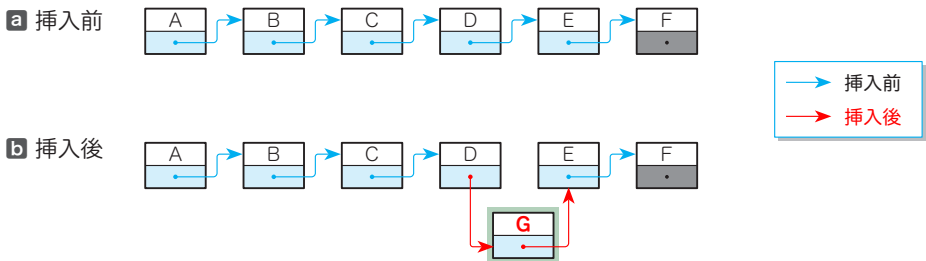
ウ 要素を参照する場合、ポインタを順番にたどる処理が必要であるため、**処理時間は長くなります**（要素を参照する際にランダムなアクセスが可能であって処理が短いのは、連結リストではなく**配列**です）。

下の図の例でノードEのデータを参照する場合を考えましょう。まず、先頭ノードAに含まれる後続ポインタをたぐってノードBに着目します。次に、ノードBに含まれる後続ポインタをたぐってノードCに着目します。…といった具合で、後続ポインタを4回たぐることによって、ようやくEに到達します。到達した後に、Eのデータを参照を行います。



エ 要素を挿入する処理は、数個のポインタを書きかえるだけで実現できるため、**処理時間は短くなります**（**配列**では、挿入する要素から後ろにあるすべての要素を後ろに移動するため処理時間が長くなります）。正解です。

下の図の例でノードDとノードEのあいだにノードGを挿入する場合を考えましょう。新しく挿入されるノードGに含まれる後続ポインタの参照先がノードEとなるように設定した上で、ノードDに含まれる後続ポインタがノードGを参照するように更新します。



平成10年度(1998年度)秋期 午前 問13

図は単方向リストを表している。“東京”がリストの先頭であり、そのポインタには次のデータのアドレスが入っている。また、“名古屋”はリストの最後であり、そのポインタには0が入っている。

アドレス 150 に置かれた“静岡”を、“熱海”と“浜松”の間に挿入する処理として正しいものはどれか。

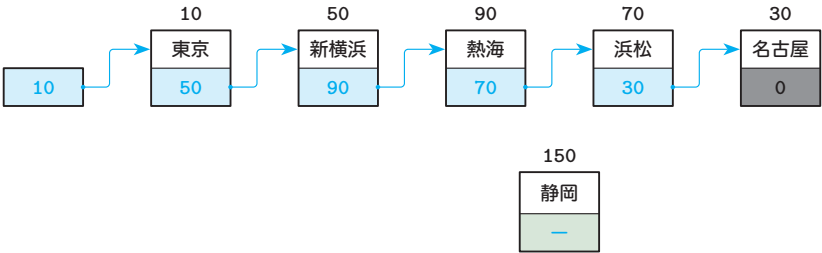
先頭データへのポインタ	アドレス	データ	ポインタ
10	10	東京	50
	30	名古屋	0
	50	新横浜	90
	70	浜松	30
	90	熱海	70
	150	静岡	

- ア 静岡のポインタを 50 とし、浜松のポインタを 150 とする。
- イ 静岡のポインタを 70 とし、熱海のポインタを 150 とする。
- ウ 静岡のポインタを 90 とし、浜松のポインタを 150 とする。
- エ 静岡のポインタを 150 とし、熱海のポインタを 90 とする。

解答：イ

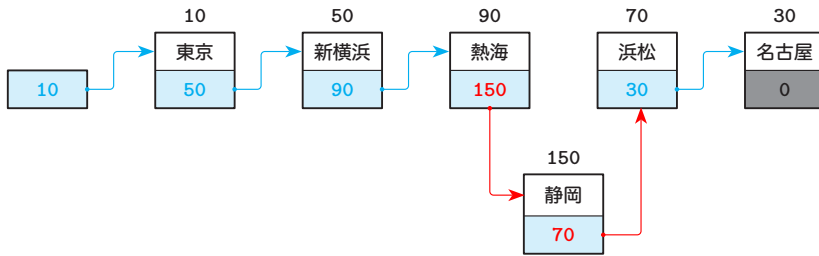
単方向リスト (単方向の線形リスト) は、各ノードが、後続ノードへのポインタをもつデータ構造です。先頭ノードからポインタをたぐることによって、リスト上に登録されているすべてのデータを走査して取り出すことができますようになっています。

本問では、リストのデータとポインタが、表の形式で配列に格納されています。与えられたリストのノード間のつながりを表したのが、次の図です。



先頭ノードは“東京”であって、その後ろに、“新横浜”、“熱海”、“浜松”、“名古屋”の順で並んでいます。なお、“静岡”は表の中に登録されているものの、リストには含まれていません。

与えられたリストの“熱海”と“浜松”のあいだに“静岡”を挿入する処理は、次の図のように行われることになります。



具体的な手続きは、次のとおりです。

- “静岡”のポインタは“浜松”を指さなければなりませんから、**70**に更新します。
- “熱海”のポインタは“静岡”を指さなければなりませんから、**150**に更新します。

正解は**イ**です。

平成8年度(1996年度)秋期 午前 問12

図のような単方向リストがある。“ナリタ” がリストの先頭であり、そのポインタには次に続くデータのアドレスが入っている。また、“ミラノ” はリストの最後であり、そのポインタには0が入っている。

“ロンドン” を“パリ” に置き換える場合の適切な処理はどれか。

先頭データへのポインタ	アドレス	データ部分	ポインタ
120	100	ウィーン	160
	120	ナリタ	180
	140	パリ	999
	160	ミラノ	0
	180	ロンドン	100

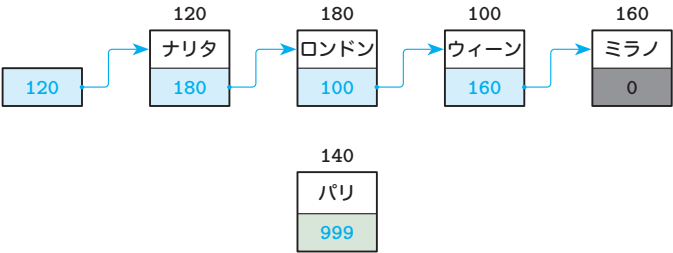
8
線形リスト

- ア パリのポインタを 100 とし、ナリタのポインタを 140 とする。
- イ パリのポインタを 100 とし、ロンドンのポインタを 0 とする。
- ウ パリのポインタを 100 とし、ロンドンのポインタを 140 とする。
- エ パリのポインタを 180 とし、ナリタのポインタを 140 とする。
- オ パリのポインタを 180 とし、ロンドンのポインタを 140 とする。

■ 解答：ア

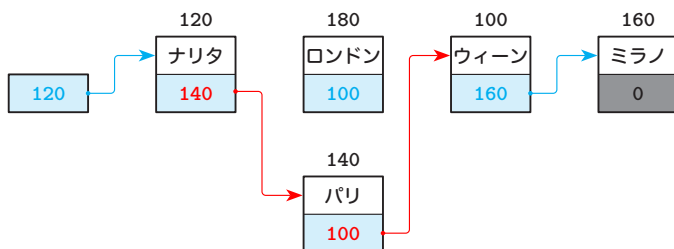
単方向リスト (単方向の線形リスト) は、各ノードが、後続ノードへのポインタをもつデータ構造です。先頭ノードからポインタをたぐることによって、リスト上に登録されているすべてのデータを走査して取り出すことができるようになっています。

本問では、リストのデータとポインタが、表の形式で配列に格納されています。与えられたリストのノード間のつながりを表したのが、次の図です。



先頭ノードは“ナリタ”であって、その後ろに、“ロンドン”、“ウィーン”、“ミラノ”の順で並んでいます。なお、“パリ”は表の中に登録されているものの、リストには含まれていません。

与えられたリストの“ロンドン” から“パリ” への置換の処理は、次の図のように行われることになります。



具体的な手続きは、次のとおりです。

- “パリ”のポインタは“ウィーン”を指さなければなりませんから、**100**に更新します。
- “ナリタ”のポインタは“パリ”を指さなければなりませんから、**140**に更新します。

正解は**A**です。

※ なお、“ロンドン” のポインタを変更する必要はありません。というのも、どこからも指されなくなった“ロンドン”は、リストから（論理的に）切り離されるからです。

平成18年度(2006年度)秋期 午前 問13

表は、配列を用いた連結セルによるリストの内部表現であり、リスト [東京, 品川, 名古屋, 新大阪] を表している。このリストを [東京, 新横浜, 名古屋, 新大阪] に変化させる操作はどれか。ここで、 $A(i, j)$ は表の第 i 行第 j 列の要素を表す。例えば、 $A(3, 1) =$ “名古屋” であり、 $A(3, 2) = 4$ である。また、 $\rightarrow$ は代入を表す。

		列	
行	A	1	2
1		“東京”	2
2		“品川”	3
3		“名古屋”	4
4		“新大阪”	0
5		“新横浜”	

	第 1 の操作	第 2 の操作
ア	$5 \rightarrow A(1, 2)$	$A(A(1, 2), 2) \rightarrow A(5, 2)$
イ	$5 \rightarrow A(1, 2)$	$A(A(2, 2), 2) \rightarrow A(5, 2)$
ウ	$A(A(1, 2), 2) \rightarrow A(5, 2)$	$5 \rightarrow A(1, 2)$
エ	$A(A(2, 2), 2) \rightarrow A(5, 2)$	$5 \rightarrow A(1, 2)$

解答：ウ

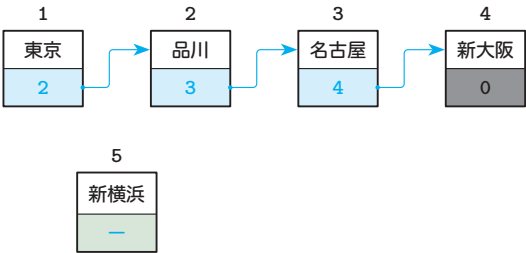
与えられた表の各列には、次のデータが格納されています。

1 列目：駅名。

2 列目：次の駅が格納されている行番号（**後続ポインタ**と呼びます）。

たとえば、“東京”の後続ポインタは2ですから、次の駅は、第2行の“品川”です。なお、“新大阪”の後続ポインタ0は、後続ノードが存在しない終着駅を意味しています。

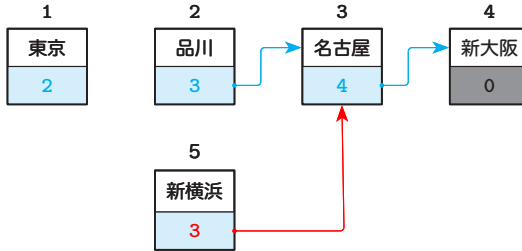
リストのノード間のつながりを表したのが、次の図です。第5行の“新横浜”は、リストに含まれていないことが分かります。



本問では、リスト中の2番目に存在する“品川”を“新横浜”に入れかえる操作が問われています。正解は選択肢②であって、次の2段階で処理が行われます。

■ Step 1 “新横浜”の次の駅を名古屋とする

リストに新しく入れられる“新横浜”の次の駅を“名古屋”にしなければなりません。そのためには、まず、除去される“品川”の後続ポインタ(“品川”の次の駅である“名古屋”の行番号3)を、“新横浜”の後続ポインタにコピーする必要があります。



具体的には、次のようになります。

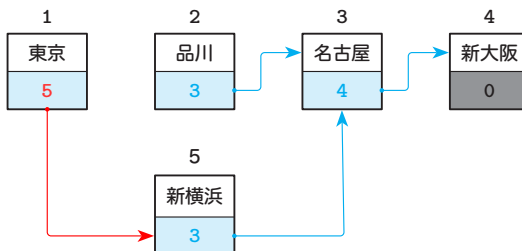
“品川”は“東京”の次の駅ですから、その行番号はA(1, 2)で得られます(値は2です)。

その“品川”の後続ポインタA(A(1, 2), 2)すなわちA(2, 2)を、“新横浜”の後続ポインタA(5, 2)にコピーします。A(5, 2)に3が代入される結果、“新横浜”の後続ポインタ3は、第3行目の“名古屋”を指すことになります。

この操作は、“A(A(1, 2), 2) → A(5, 2)”です。

■ Step 2 “東京”の次の駅を“新横浜”にする

“東京”の次の駅を“新横浜”にするために、“東京”の後続ポインタA(1, 2)に、“新横浜”の行番号5を代入します。



この操作は、“5 → A(1, 2)”です。

※ どこからも指されない(どのノードからもたどることのできない)“品川”は、リストから(論理的に)切り離されることになります。

平成17年度(2005年度)秋期 午前 問13

データ構造に関する記述のうち、適切なものはどれか。

- ア 2分木は、データ間の関係を階層的に表現する木構造の一種であり、すべての節が二つの子をもつデータ構造である。
- イ スタックは、最初に格納したデータを最初に取り出す先入れ先出しのデータ構造である。
- ウ 線形リストは、データ部と次のデータの格納先を指すポインタ部から構成されるデータ構造である。
- エ 配列は、ポインタの付替えだけでデータの挿入・削除ができるデータ構造である。

■ 解答：ウ

- ア **2分木**は、データ間の関係を階層的に表現する木構造の一種です。各節の子の数は最大で二つです(子がゼロあるいは一つのノードが存在し得ます)。
- イ **スタック**は、最後に格納したデータを最初に取り出す後入れ先出しのデータ構造です。
- ウ **線形リスト** (に含まれる個々のノード＝セル) は、データ部と、次のデータの格納先を指すポインタ部とから構成されるデータ構造です。
- エ **配列**は、添字＝インデックスによって要素を特定するデータ構造です。ポインタの付替えだけでデータの挿入・削除ができるデータ構造は**線形リスト**です。

平成22年度(2010年度)春期 午前 問5

双方向のポインタをもつリスト構造のデータを表に示す。この表において新たな社員Gを社員Aと社員Kの間に追加する。追加後の表のポインタa～fの中で追加前と比べて値が変わるポインタだけをすべて列記したものはどれか。

表

アドレス	社員名	次ポインタ	前ポインタ
100	社員A	300	0
200	社員T	0	300
300	社員K	200	100

追加後の表

アドレス	社員名	次ポインタ	前ポインタ
100	社員A	a	b
200	社員T	c	d
300	社員K	e	f
400	社員G	x	y

- ア a, b, e, f イ a, e, f ウ a, f エ b, e

■ 解答：ウ

双方向リスト（重連結リスト）の各ノードには、次の二つのポインタが与えられます。

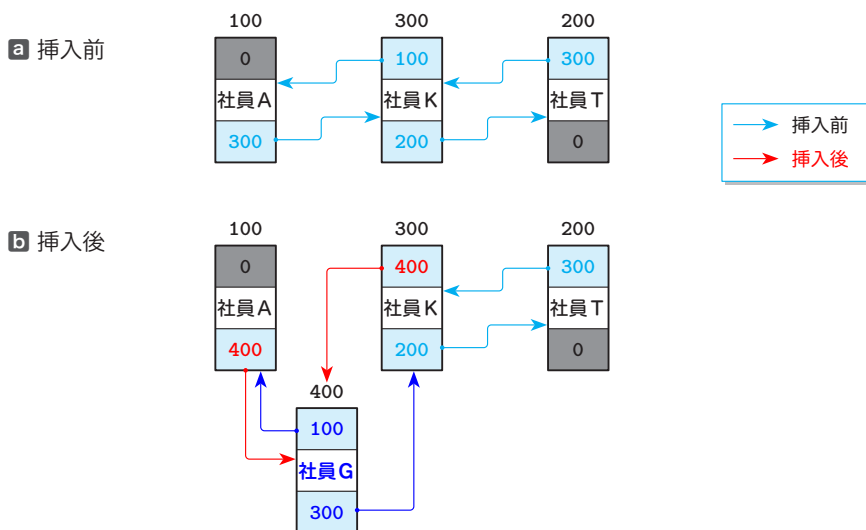
- 後続ノードを指す**次ポインタ**（後続ポインタ） ※後続ノードの格納場所のアドレス
- 先行ノードを指す**前ポインタ**（先行ポインタ） ※先行ノードの格納場所のアドレス

これらのポインタをたぐることによって、後方のデータ、あるいは前方のデータをアクセスできる**データ構造**です。

表の1行目に着目します。社員Aのデータは100番地に格納されていて、その後続ノードである“次の社員”の格納場所は300番地です（すなわち社員Kです）。また、先行ノードである“前の社員”の格納場所は0番地です。これはAよりも前に社員がいないこと、すなわち社員Aが先頭ノードであることを示しています。

社員Kは2番目のノードであり、社員Tは3番目のノードであることが分かります。なお、社員Tの“次の社員”の格納場所は0番地です。これはTよりも後ろに社員がいないこと、すなわち社員Tが末尾ノードであることを示しています。

本問で与えられた双方向リストのノード間のつながりを表したのが、次の図**a**です。



社員Gを社員Aと社員Kのあいだに挿入（追加）した後の状態が図**b**です。二つのポインタが変更されています。

■ 社員Aの次ポインタ

社員Kではなく社員Gを指すように変更されています（300から400に更新されています）。

■ 社員Kの前ポインタ

社員Aではなく社員Gを指すように変更されています（100から400に更新されています）。

正解が**ウ**であることが分かりました。

9

木構造と2分探索木

■平成16年度(2004年度)春期 午前 問43

データ構造の一つである木構造に関する記述として、適切なものはどれか。

- ア 階層の上位から下位に節点をたどることによって、データを取り出すことができる構造である。
- イ 格納した順序でデータを取り出すことができる構造である。
- ウ 格納した順序とは逆の順序でデータを取り出すことができる構造である。
- エ データ部と一つのポインタ部で構成されるセルをたどることによって、データを取り出すことができる構造である。

9

木構造と2分探索木

■解答：ア

- ア **木構造**は、階層の上位から下位に節点をたどることによって、データを取り出すことができる構造です。

節点のたどり方には、複数の種類があります。

■ **横型探索／幅優先探索** (breadth-first search)

レベルの低いノードから始めて左側から右側へと進みます。進め終わると、次のレベルへと下る、という処理を繰り返します。

■ **縦型探索／深さ優先探索** (depth-first search)

葉に到達するまで下流にくだるのを優先する方法です。葉に到達して行き止まりとなった際は、いったん親に戻って、それから次のノードをたどります。

走査にあたっては、各ノードを何度も通過します。どのタイミングで“立ち寄り”のかによって次の3種類の走査法に分類されます。

行きがけ順／前順／先行順／前置順 (preorder)

通りがけ順／間順／中間順／中置順 (inorder)

帰りがけ順／後順／後行順／後置順 (postorder)

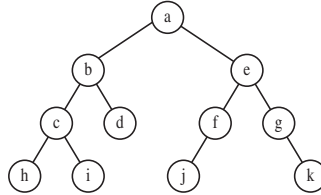
- イ 格納した順序でデータを取り出すことができるのは、データを**先入れ先出し** = FIFO (first-in first-out) で蓄える**キュー** (queue) です。
- ウ 格納した順序とは逆の順序でデータを取り出すことができるのは、データを**後入れ先出し** = LIFO (last-in first-out) で蓄える**スタック** (stack) です。
- エ データ部と一つのポインタ部で構成されるセル (ノード) をたどることによって、データを取り出すことができるのは、**単方向の線形リスト**です。

■ 平成15年度(2003年度)秋期 午前 問12

2分木の走査の方法には、その順序によって次の三つがある。

- (1) 前順：節点、左部分木、右部分木の順に走査する。
- (2) 間順：左部分木、節点、右部分木の順に走査する。
- (3) 後順：左部分木、右部分木、節点の順に走査する。

図に示す2分木に対して前順に走査を行い、節の値を出力した結果はどれか。



ア abchidefjgk

イ abechidfjgk

ウ hcibdajfegk

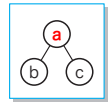
エ hicdbjfkgea

■ 解答：ア

本問で問われてい走査は、**縦型探索(深さ優先探索)**の一つである**前順(行きがけ順／先行順／前置順)**です。

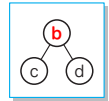
まず、根aと、その二つの子bとcに着目します。

走査の順が『a ⇨ bを根とする部分木 ⇨ eを根とする部分木』ですから、最初に出力されるのが**a**であることが分かります。



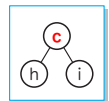
それでは、左側のbを根とする部分木に着目します。

走査の順が『b ⇨ cを根とする部分木 ⇨ dを根とする部分木』ですから、2番目に出力されるのが**b**であることが分かります。



次に、cを根とする部分木に着目します。

走査の順が『c ⇨ hを根とする部分木 ⇨ iを根とする部分木』ですから、3番目に出力されるのが**c**であることが分かります。

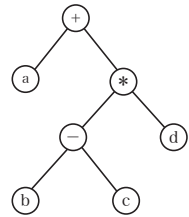


ここまで**a b c**と出力されたのですから、正解は**ア**です。

■平成19年度(2007年度)秋期 午前 問12

2 分木の各ノードがもつ記号を出力する再帰的なプログラム Proc(ノード n) は、次のように定義される。このプログラムを、図の 2 分木の根 (最上位のノード) に適用したときの出力はどれか。

```
Proc(ノード  $n$ ) {  
     $n$  に左の子  $l$  があれば Proc( $l$ ) を呼び出す  
     $n$  に右の子  $r$  があれば Proc( $r$ ) を呼び出す  
     $n$  に書かれた記号を出力する  
}
```



- ア $b - c * d + a$ イ $+ a * - bcd$ ウ $a + b - c * d$ エ $abc - d * +$

9

木構造と2分探索木

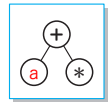
■解答：エ

プログラム Proc(ノード n) では、“左部分木⇒右部分木⇒ノード n ” という縦型探索 (深さ優先探索) の一つである後順 (行きがけ順／後行順／後置順) での走査が行われます。

根に対してプログラム Proc(ノード n) を適用すると、どのように表示が行われるのかを検討しましょう。

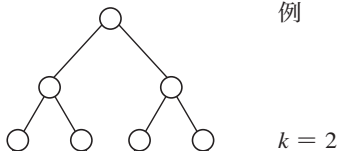
走査の順は “ a を根とする部分木 ⇒ $*$ を根とする部分木 ⇒ $+$ ” ですから、最後に出力されるのは $+$ です。

したがって、最後に $+$ が出力される選択肢 **エ** が正解です (これ以上の検討は不要です)。



■平成17年度(2005年度)秋期 午前 問12

すべての葉が同じ深さをもち、葉以外のすべての節点が二つの子をもつ 2 分木に関して、節点数と深さの関係を表す式はどれか。ここで、 n は節点数、 k は根から葉までの深さを表す。例に示す 2 分木の深さ k は 2 である。



- ア $n = k(k + 1) + 1$ イ $n = 2^k + 3$
ウ $n = 2^{k+1} - 1$ エ $n = (k - 1)(k + 1) + 4$

■解答：ウ

最下流の葉を除くすべてのノード (節点) が二つの子をもつ 2 分木である、**全 2 分木** (full binary tree) に関する問題です。

ノードの個数は、深さの増加に伴って次のように増えていきます。

深さ1：1+2

= 2

深さ2：1+2+4

= 7

深さ3：1+2+4+8

= 15

深さ4：1+2+4+8+16

= 31

木の深さが k の全2分木がもつことのできるノードは、 $2^{k+1}-1$ であることが分かります。したがって、正解はウです。

平成10年度(1998年度)春期 午前 問14

図1の二分木を配列で表現したものが図2である。aに入る値はどれか。

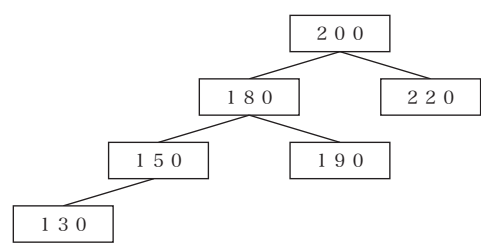


図1 二分木

添字	値	ポインタ1	ポインタ2
1	200	3	2
2	220	0	0
3	180	5	a
4	190	0	0
5	150	6	0
6	130	0	0

図2 二分木の配列表現

- ア 2
- イ 3
- ウ 4
- エ 5

解答：ウ

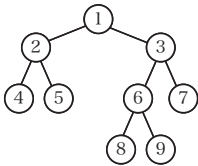
図2の表で示されている“ポインタ1”と“ポインタ2”は、それぞれ、左の子ノードが格納されている添字と右の子ノードが格納されている添字です。

問われているのは、180の右の子ノードの格納されている添字です。右の子ノード190は、添字が4の位置に格納されています。したがって、正解はウです。

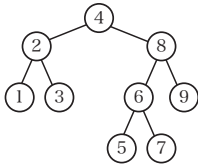
平成17年度(2005年度)春期 午前 問12

2分探索木として適切なものはどれか。ここで、1～9の数字は、各ノード（節）の値を表す。

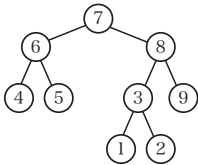
ア



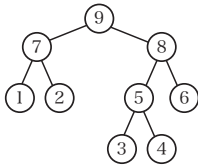
イ



ウ



エ



9

木構造と2分探索木

■ 解答：イ

2分探索木は、どのノードに着目しても、

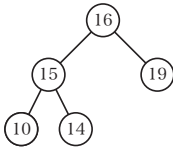
左部分木のノードのキーの最大値 < キー値 < 右部分木のノードのキーの最小値

が成立する2分木です。この条件を満たしているのはイです。

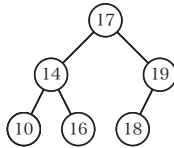
平成28年度(2016年度)秋期 午前 問6

2分探索木になっている2分木はどれか。

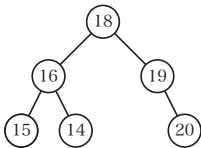
ア



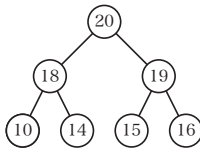
イ



ウ



エ



■ 解答：イ

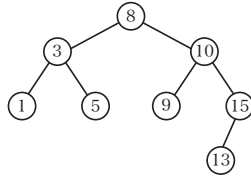
2分探索木は、どのノードに着目しても、

左部分木のノードのキーの最大値 < キー値 < 右部分木のノードのキーの最小値

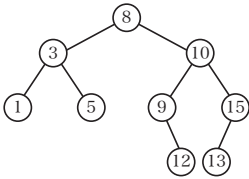
が成立する2分木です。この条件を満たしているのはイです。

■ 平成19年度(2007年度)春期 午前 問12

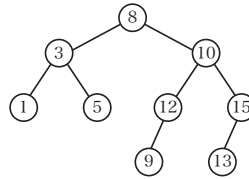
次の2分探索木に 12 を追加したとき、追加された節 12 の位置を正しく表している図はどれか。



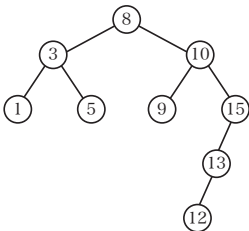
ア



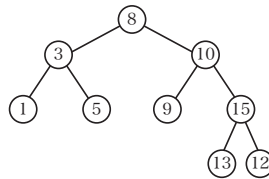
イ



ウ



エ



■ 解答：ウ

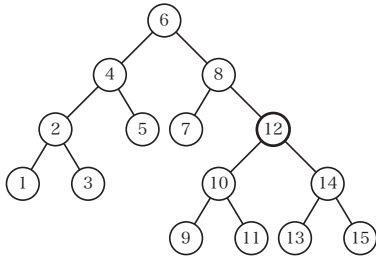
2分探索木は、どのノードに着目しても、

左部分木のノードのキーの最大値 < キー値 < 右部分木のノードのキーの最小値

が成立する2分木です。この条件を満たしているのはウです。

平成25年度(2013年度)春期 午前 問5

次の2分探索木から要素12を削除したとき、その位置に別の要素を移動するだけで2分探索木を再構成するには、削除された要素の位置にどの要素を移動すればよいか。



9

木構造と2分探索木

- ア 9 イ 10 ウ 13 エ 14

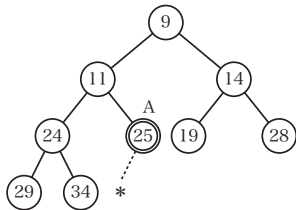
■ 解答：ウ

2分探索木は、どのノードに着目しても、
左部分木のノードのキーの最大値 < キー値 < 右部分木のノードのキーの最小値

が成立する2分木です。
削除される12の位置に、11もしくは13を移動すれば2分探索木の要件が維持されます。選択枝に含まれるのは13ですから、正解はウです。

平成20年度(2008年度)秋期 午前 問12

親の節の値が子の節の値より小さいヒープがある。このヒープへの挿入は、要素を最後部に追加し、その要素が親よりも小さい間、親と子を交換することを繰り返せばよい。次のヒープの*の位置に要素7を追加したとき、Aの位置に来る要素はどれか。

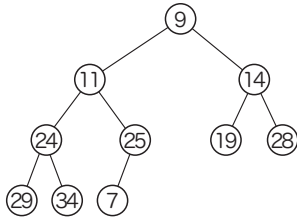


- ア 7 イ 11 ウ 24 エ 25

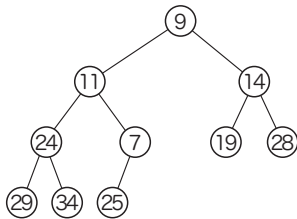
■ 解答：イ

ヒープ (heap) は、親の値が子の値以上であるという条件を満たす完全2分木です。

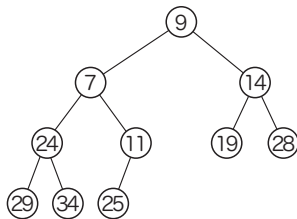
*の位置に7を挿入すると、次のようになります。



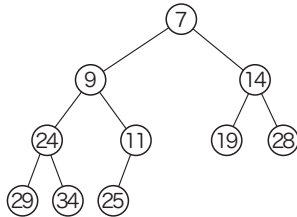
挿入された7が親の25より小さいため、これらを交換します。次のようになります。



交換された7は、親の11より小さいため、これらを交換します。次のようになります。



交換された7は、親の9より小さいため、これらを交換します。次のようになります。



Aの位置には11が位置しています。正解はイです。

制 作 … 柴田 望洋
装 丁 … 柴田 望洋
編 集 … 福井 荘介

新・明解Javaで学ぶアルゴリズムとデータ構造 第3版

読者特典PDF書籍②

基本情報技術者試験過去問67問の徹底学習

2026年9月10日 初版発行

著 者 … 柴田 望洋
発行者 … 出井 貴完
発行所 … S Bクリエイティブ株式会社
〒105-0001 東京都港区虎ノ門2-2-1
<https://www.sbcr.jp/>