基本情報処理演習

Report#1

提出日：2007年12月25日（火曜日）
所属　：工学部情報工学科
氏名　：津波古正輝
学籍番号：e075739A

1．次の排他的論理和に関する恒等式を、真理表を用いて証明せよ。

A EOR B = AB' + A'B = (A + B)(A' + B')

A	B	$\bar{A}$	$\bar{B}$	A+B	$\overline{A+B}$	A·B	$\overline{A \cdot B}$
0	0	1	1	0	1	0	1
0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	0
1	1	0	0	1	0	1	0

A⊕B	$A \cdot \bar{B}$	$\bar{A} \cdot B$	$A \cdot \bar{B} + \bar{A} \cdot B$	$\bar{A} + \bar{B}$
0	0	0	0	1
1	0	1	1	1
1	1	0	1	1
0	0	0	0	0

A	B	$(A+B)(\bar{A}+\bar{B})$
0	0	0
0	1	1
1	0	1
1	1	0

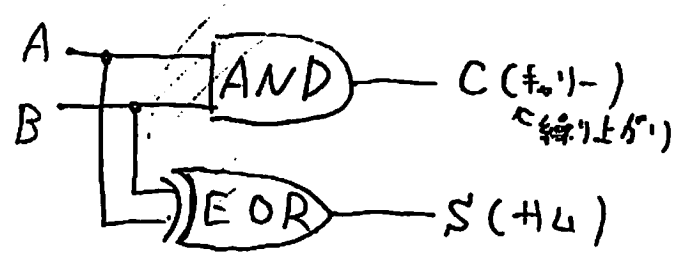
以上の図より、

A eor B = AB' + A'B = (A + B)(A' + B')

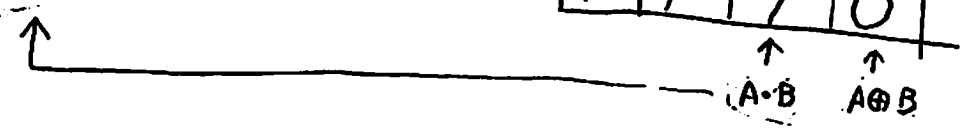
となる。

2.半加算器（ＨＡ）と全加算器（ＦＡ）を真理表と回路と用いて説明し、下記の４ビットの演算をＨＡとＦＡで実現せよ。

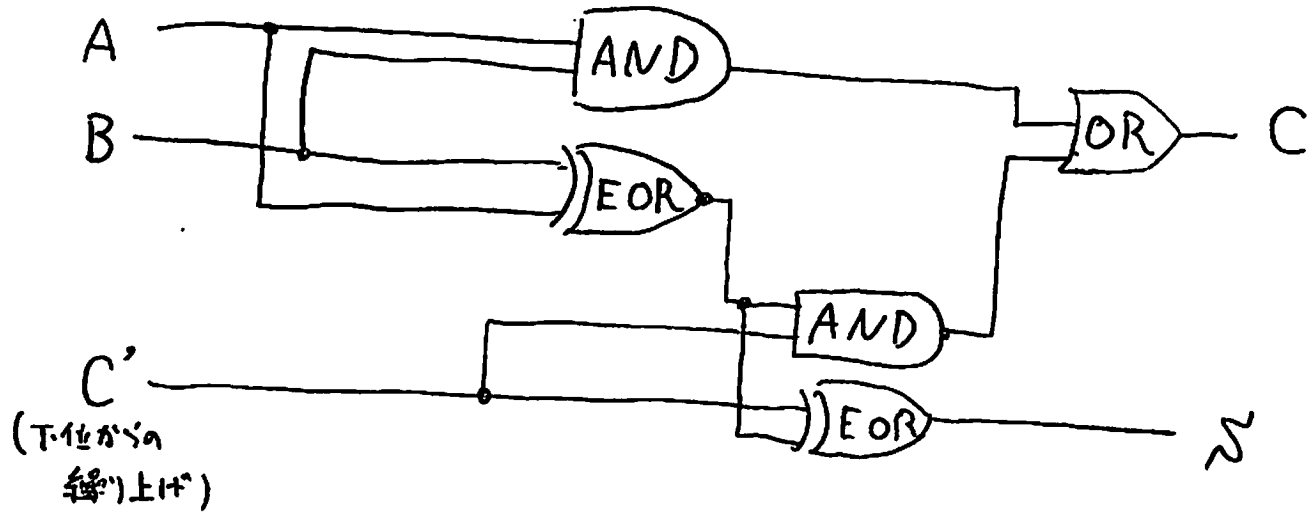
半加算器(HA)



A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



全加算器(FA)



例 $(1011)_2 + (1100)_2$

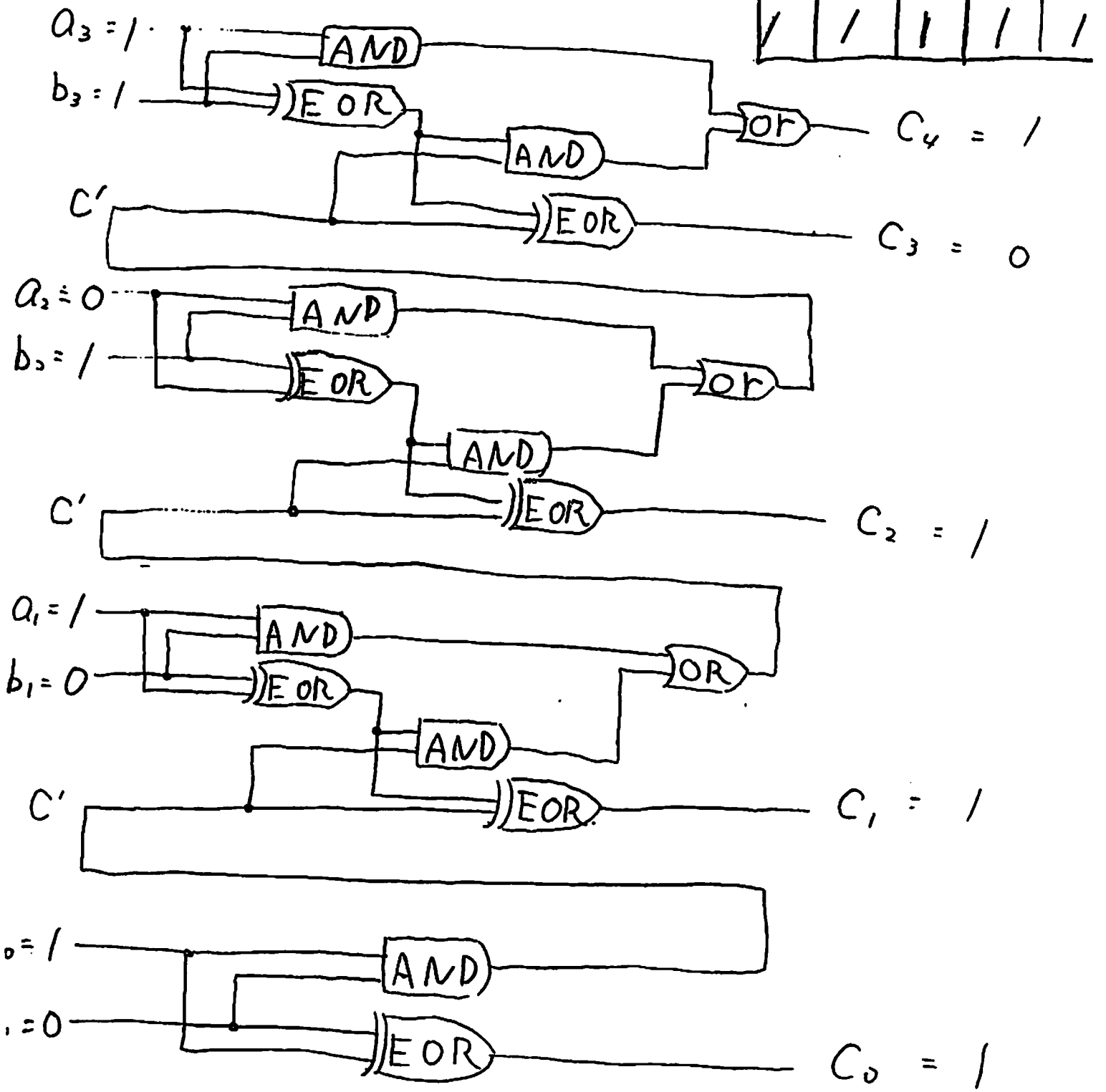
$= (10111)_2$

$a_3 a_2 a_1 a_0 = (1011)_2$

$b_3 b_2 b_1 b_0 = (1100)_2$

$C_4 C_3 C_2 C_1 C_0 = (10111)_2$

A	B	C'	C	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	C
1	0	0	0	
1	0	1	1	C
1	1	0	1	C
1	1	1	1	1



３.固定小数点と浮動小数点について説明せよ。

固定小数点…小数点の位置を決めてデータを表す方法。表したい数字が整数の時は、最下位ビットの右側に固定。表したい数字が小数のとき最上位ビットの右側に固定。つまり、10進数『1.05』を固定小数点の記法で2進数で表すと整数部は1、小数部は0.05となるので、基数変換をし、2進数『1.01』となる。

浮動小数点…高精度の計算をしたいときにする記法。具体的には大きい値や小さい値を計算するときに用いる。固定小数点と比べ、表現できる範囲の数値が広い。数値を、各桁の値の並びである「仮数部」と、小数点の位置を表わす「指数部」で表現。仮数部に、底を指数でべき乗した値をかけて実数を表現する。

例：10進数(76.5)を16進数の浮動小数点で表す。

76.5を16進数に基数変換＝2進数(1001100.1)＝16進数(4C.8)

正規化：4C.8=0.4C8×16の2乗

完成形:0（符号）2（指数部）4C8（仮数部）=024C8000

４.ＭＩＰＳとＭＦＬＯＰＳについて説明せよ。

MIPS…コンピュータの処理速度を表す単位。1MIPS=100万回の命令を処理できることを示す。英語で書くとMillion Instructions Per Second。100万回の命令/1秒間の、と訳せる。

MFLOPS…コンピュータの処理速度を表す単位。1秒間に100万回の計算が処理できることを示す。ここでFLOPSの説明をしておく。FLOPSもコンピュータの処理速度を表す単位で1FOPSは1秒間に1回の計算ができることを示す。MFLOPSの『M』はメガ=100万の意味なので、MFLOPSは「1秒間に100万回の処理を実行できる単位」ということになる。

5.補助単位についてまとめよ。

ある一つの単位だけで、データを示すことになると、データが大きくなったときに、桁数が増えて表現が非常に不便。なので、ある基準にきたら別の単位を使う。その「ある基準」が「1000倍ごと」である。

記号	読み方	指数表現
T	テラ	10^12
G	ギガ	10^9
M	メガ	10^6
k	キロ	10^3
m	ミリ	10^-3
μ	マイクロ	10^-6
n	ナノ	10^-9
p	ピコ	10^-12

6．2の補数表現について16ビットの数値を用いて2進数、10進数、16進数で説明せよ。

(0000000000010101)2=(21)10=(2B)16で考える。その前に、補数とは何?

補数とは…

ある数Nの補数とは、基数（ある基準となる数）からそのNの値を引いた数のことです。

（補数）　＝　（基数）　－　N

2進数の場合、「2の補数」と「1の補数」があります。2の補数とは全ビットを反転させ、プラス1をした数字です。コンピュータの場合、マイナスの値を補数を用いて表現しています。

とあった。つまり、元の数字を、その数字の2の補数を加算すると、『0』になるということだ。表を作ってみると…

2進数 = 0000 0000 0001 0101
10進数 = 21
16進数 = 2B

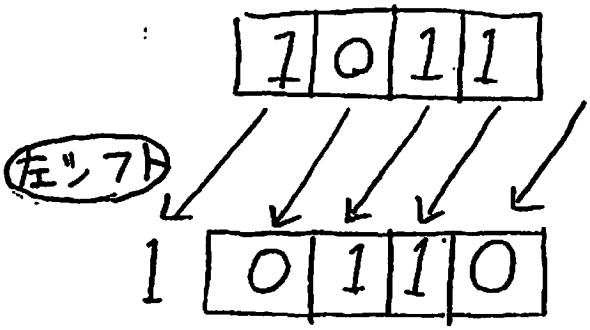
	2進数	10進数	16進数
	0000 0000 0001 0101	21	15
1の補数	1111 1111 1110 1010	22	FFDA
2の補数	1111 1111 1110 1011	-21	FFDB

このような結果になった。2進数、10進数、16進数のもとの数字と、2の補数を加算してみると、2進数と16進数は、きれいに桁が上がり、最上位ビットが1、それ以外が0になる。しかし、最上位ビットの桁上がりは無視するので、答えは『0』である。10進数は21+(-21)=0となる。よって、2の補数とは、元の数字と加算すると0になる、つまり、その数字にマイナスを付けた値になる事が分かる。

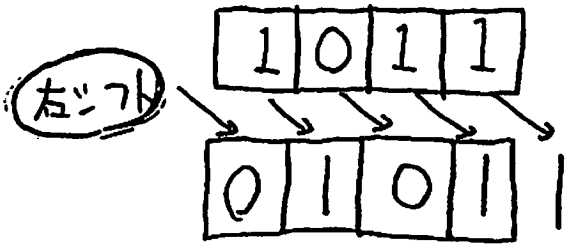
7. 論理シフトと算術シフトについてまとめよ。

例：1011

論理シフト

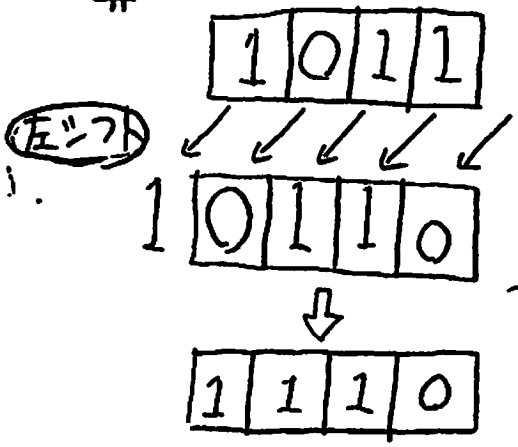


- あふれ出たビットは捨てる。
- 空きビットには0を入れる。

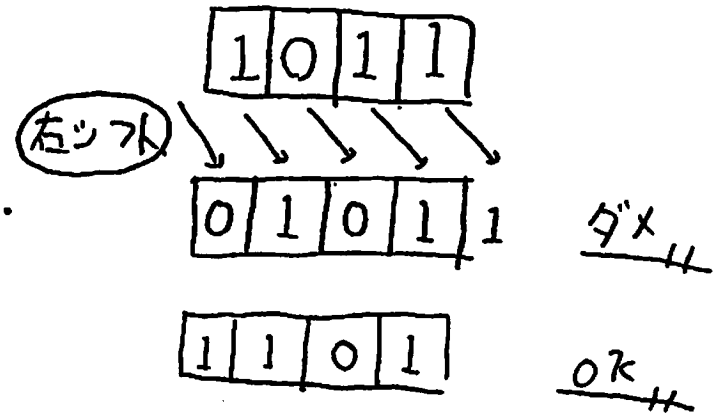


- あふれ出たビットは捨てる。
- 空きビットには0を入れる。

算術シフト



- あふれ出たビットは捨てる。
- 空きビットには0を入れる。
- 符号ビットはそのまま。



- あふれ出たビットは捨てる
- 空きビットには
  符号ビットと同じものを
  入れる。
- 符号ビットはそのまま。

まとめ：図を見てみると、論理シフトは右シフトも左シフトも、シフトして空きができてしまった場所には『0』が入る。算術シフトは符号に注意しする必要がある。算術シフトは右シフトと左シフトでは、空きビットに入るものがちがってくる。右シフトして空きができてしまった場所には『符号ビット』と同じものが入り、左シフトで空きができてしまった場所には『0』がはいる。論理、算術で共通なところは「溢れ出たビットは捨てる」ということだ。全体をまとめるとこうなる。

	論理シフトでの空きビット	算術シフトでの空きビット
左シフト（乗算）	0	0
右シフト（除算）	0	符号ビットと同じ数字

8．3種類の探索方法（線形探索、2分探索、ハッシュ探索）について、論ぜよ。

線形探索…最初から順番にデータを比較し、探索していく方法。

メリット

アルゴリズムが単純である。

データがどんな順番に並んでいたとしても探すことができる。つまり、データがソートされていなくても大丈夫。

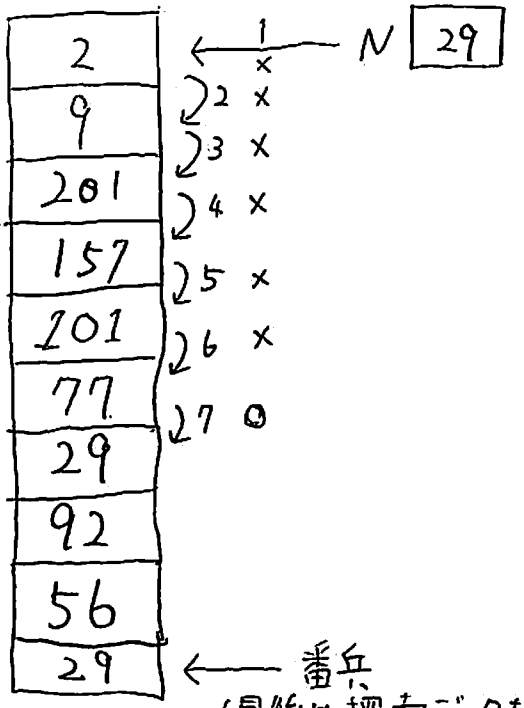
デメリット

最悪の場合、すべてのデータを調べなければならない。
与えられたデータの中に探索対象のデータがなくても、すべてのデータを調べてしまう。
与えられたデータの中に探索対象のデータが複数あったとしても、最初のデータしか見つけられない。

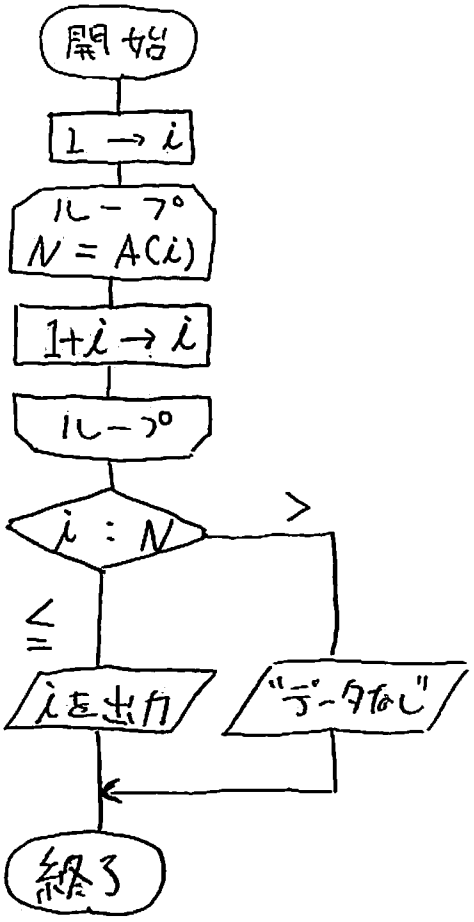
線形探索法の最大探索回数とくれば　データがN件の場合→N回
線形探索法の平均探索回数とくれば　データがN件の場合→（N＋1）÷2回

$N=29$ というデータを探すとき、

線形探索法



(最後は探索データを入れておく。これにより、データ件数を気にすることなく終了の判定が容易になる)



※ $i:N$ について

番兵があるかないか、ここで差がでる。

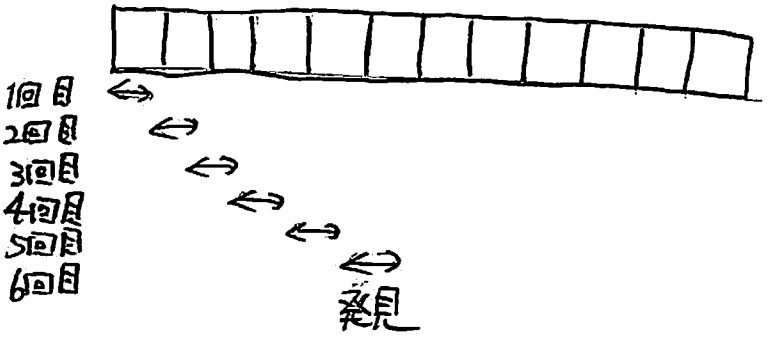
もし番兵があったのならば

$i = N+1$ となり、

$i > N$ となる。

$i>N$ という条件で発見された時は、「番兵で見つかった」ということになる。

線形探索法 イメージ



2分探索…データを二等分して探索範囲を調べていく方法。２分検索は、値が昇　順か（降順に）ソートされていることが必要だが、「ソートされたデータ」に対しては極めて強力な威力を発揮するアルゴリズム。

メリット

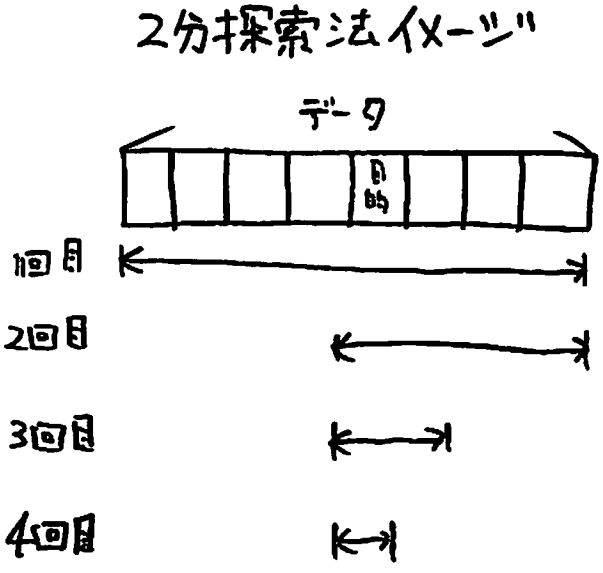
探索速度が速い。

デメリット

探索対象のデータをあらかじめソートしておかなければならない。

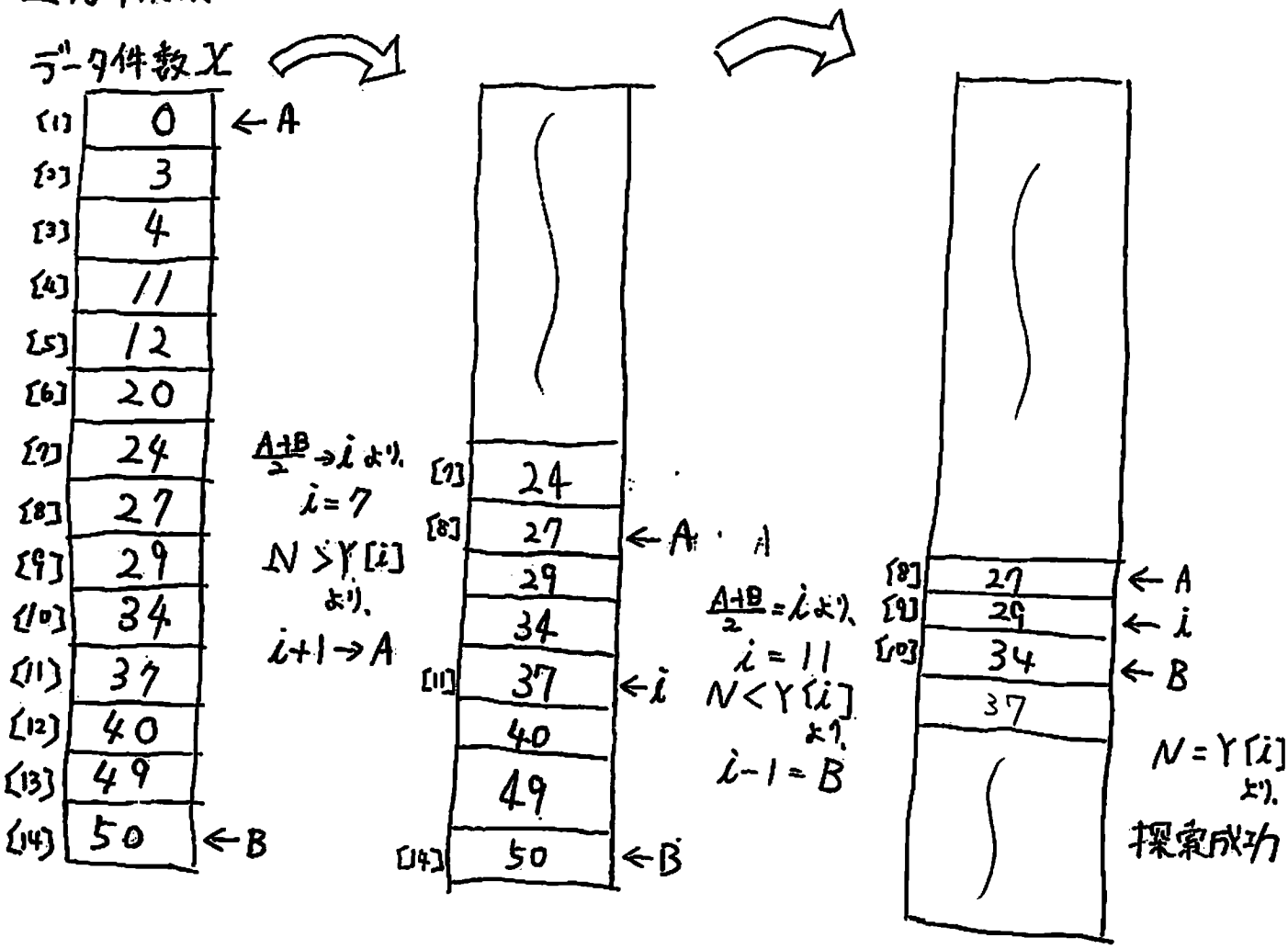
二分探索法の最大探索回数とくれば　データがN件の場合→log2N+1回

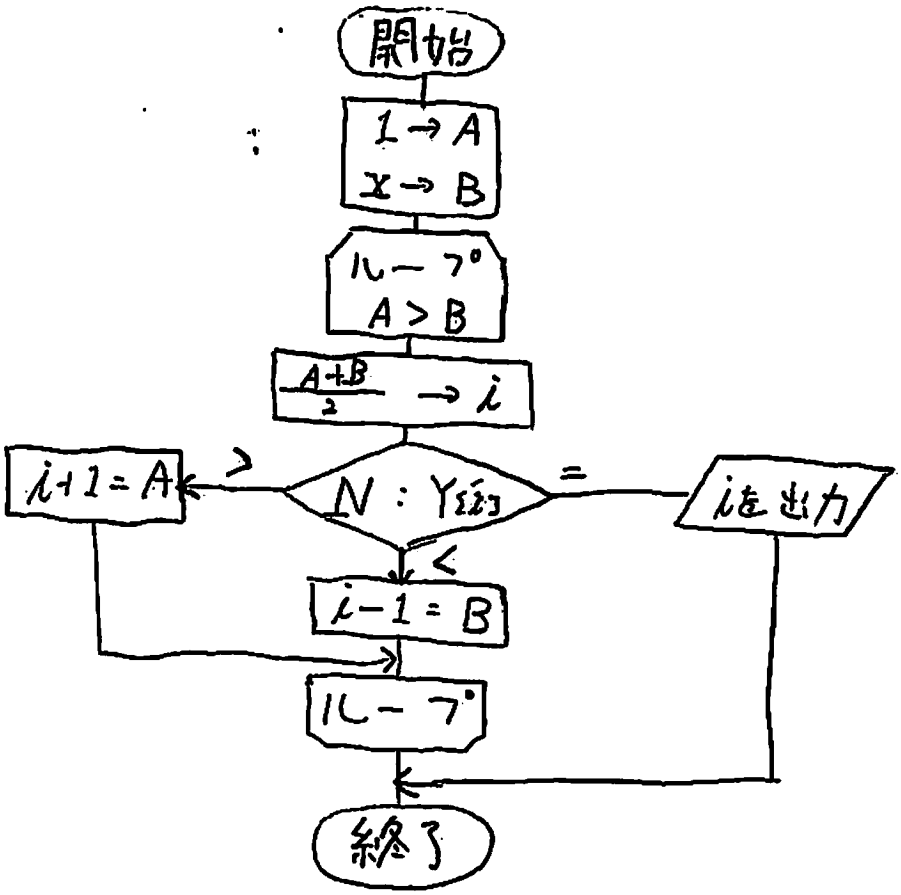
二分探索法の平均探索回数とくれば　データがN件の場合→log2N回



N=24のとき。

2分探索法





ハッシュ探索…ハッシュ関数を決めて、そのハッシュ関数で計算された値（ハッシュ値）を利用してデータの格納アドレスを決定する。しかし、ハッシュ関数で計算後のハッシュ値が同じになり、異なるデータに同じハッシュ値が割り当てられる事がある。このことをコリジョン（又はシノニム）といい、最初に格納されていたデータをホームレコード、後から格納されたデータをシノニムコードをいう。この解決方法は、

ハッシュ関数側に衝突困難な性質を持たせる（衝突を減らすためのアプローチ）。

単純にmodだけで考えてしまうと簡単に衝突が起きてしまう。

衝突が起こったときにデータ格納場所のずらし方に工夫する（衝突が起きたときの問題を解決するアプローチ）。例;オープンアドレス

空きが見つかるまでハッシュ値に1を加えていくのがオープンアドレス法である。1を加えたハッシュ値のことを再ハッシュ値をいい、再ハッシュ値=格納する配列の大きさ、になるならば、再ハッシュ関数=1となる。

が考えられる。

メリット

シノニムが発生しない場合、要素の数に関係なくハッシュ関数の計算が1回実行される。

デメリット

よいハッシュ関数を用いないとシノニムが発生してしまう。

9.

ソート（バブルソート、挿入ソート、選択ソート、シェルソート、ヒープソート、クィックソート、マージソート）について論ぜよ。

バブルソート：

一つ一つ確認、入れ替えしていき、確定していく方法。データの件数を n とすると、バブルソートは必ず n(n - 1)/2 回のスキャンが行われる。データ件数がnの時、1回目で1番目に軽い(小さい)値がリストの先頭に移動してゆき、2回目のスキャンで2番目に軽い値を浮かびあがらせ、3回目のスキャンで、という具合に n 回のスキャンで n 番目に軽い値を浮かびあがらせるのがバブルソートの特徴です。（条件を入れ換えると「重い(大きい)値を浮かびあがらせる」と述べることができる）

リストの個数が多くなればなるほど処理速度も遅くなりますが、シンプルなアルゴリズムなのでデータ量が少ないときには手軽に実装できる。

挿入ソート：

整列済みの配列要素とデータを比較して、挿入していく。1番目と2番目を比較し、順番が逆であれば入れ換える。次に、3番目の要素を、正しい順に並ぶように「挿入」する。（挿入する際、右側のデータを後ろに一つずつずらす。）この操作で、3番目までのデータが整列済みとなる（但し、データが挿入される可能性があるので確定ではない）。このあと、4番目以降の要素について、整列済みデータとの比較と適切な位置への挿入を繰り返す。

選択ソート：

最悪の場合の計算時間が長めだが、アルゴリズムが単純。昇順の場合は配列の中の最小値を、降順の場合は最大値を選択して、配列の先頭から順に値を入れ替えていく整列方法。これを改良したのが、ヒープソート。

シェルソート:

基本的に基本挿入法と変わらない。基本挿入法が１つの 数列に挿入していく方法であるのに対し、シェルソートは元の数列を仮想的に 複数の数列と考え、その１つ１つの数列に基本挿入法を用いて、徐々に数列の 数を減らし最終的に１つの数列にして、もう一度基本挿入法を行う方法である。挿入ソートは「ほとんど整列されたデータに対しては高速」という特長があるものの、「隣り合った要素同士しか交換しない」ため、あまり整列されていないデータに対しては低速。『ほとんど整列』の状態の時に挿入ソートを行う。この『ほとんど整列』の場合は高速。

ヒープソート:

まず配列をヒープ構造にしなければいけない。.ヒープ構造の特徴としては

1,根（配列の最初、N[1]に必ず最大値がくる
2,N[1]とN[2]とN[3],N[2]N[4]N[5]というようにN[a],N[2a],N[2a+1]のデータ
　をそれぞれ比較した場合、必ずN[a]のデータが一番大きくなる。
という2つの特徴がある.また、親子関係を比較しても必ず親が大きくなる

配列Nをヒープ構造にする過程

配列Nのデータ数がｎのとき、i=n/2、a=i として、配列Nの　データ
N[a],N[2a],N[2a+1]を比較し

- N[a]が最大の時、そのまま。i=i-1、a=i として、N[a],N[2a],N[2a+1]を比較。
- N[2a]が最大の時、N[a]とN[2a]のデータを交換(入れ替える)。a=2a として、N[a],N[2a],N[2a+1]を比較。
- N[2a+1]が最大の時、N[a]とN[2a+1]のデータを交換。a=2a+1 として、N[a],N[2a],N[2a+1]を比較。
- 葉(N[2a],N[2a+1]がない場合)の時は、そのまま。i=i-1、a=i として、N[a],N[2a],N[2a+1]を比較。

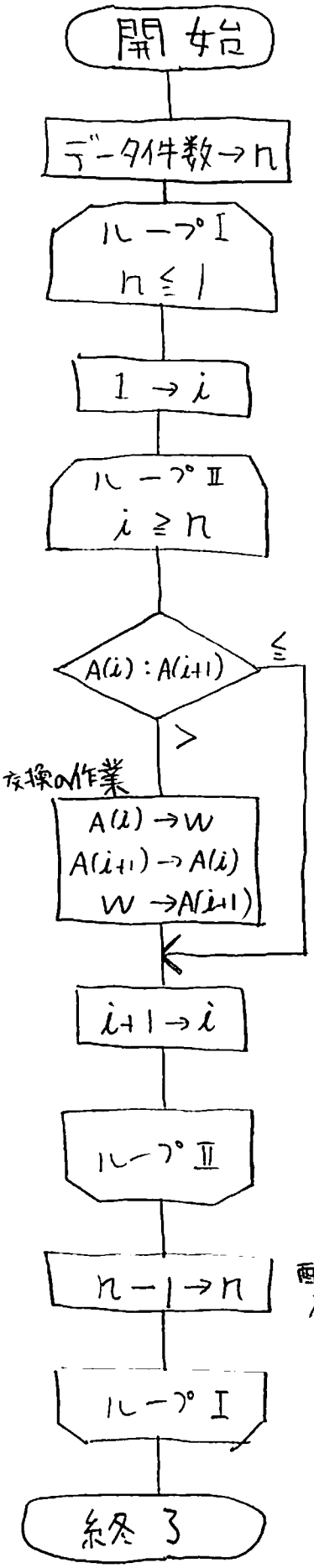
a=1のときの比較・交換をすれば終了。

マージソート：

あらかじめ整列してある 2 つの配列を合体させて、新しい、整列された配列を得るのは容易なことで、マージソートはこれに着目して、並べ替えたい配列を再帰的に分割していき、再び併合（マージ）していくことで、並び替えを実現しようとする、ソートアルゴリズム。もっとも小さい配列（要素数 1）まで分解されたら、それを順序良く併合していく。このとき、2 つの配列の先頭から小さい方を取り出して新しい配列を作成するようにすれば、取り出すだけで整列された配列を作成することが可能。

それぞれのソート法のフローチャートと、具体例を書く。

バブルソート

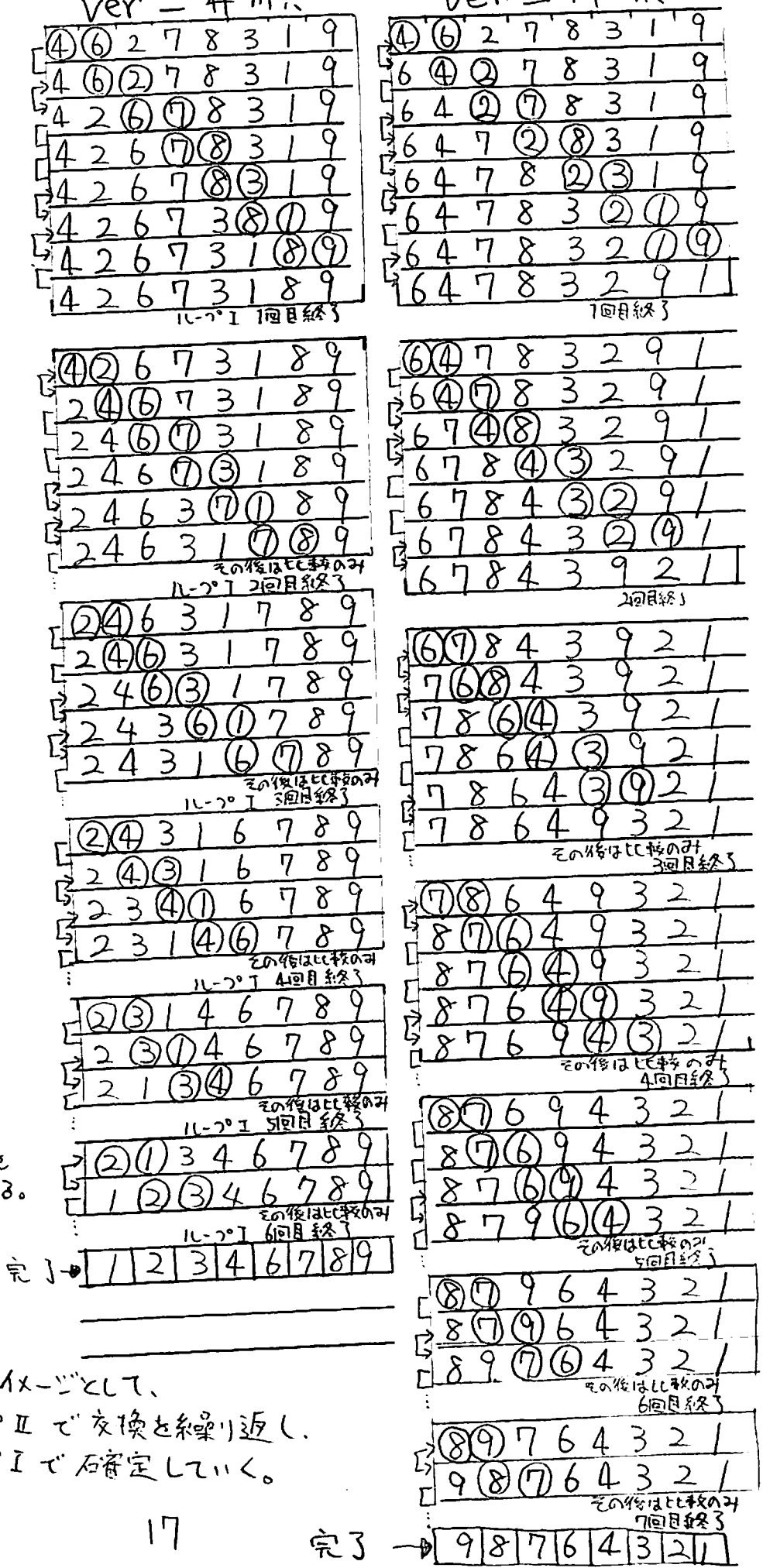


配列の最後を
右確定させている。

全体のイメージとして、
ループIIで交換を繰り返し、
ループIで右確定していく。

⟷ 交換 ， —— 比較のみ ， ○ 対象
（上から見た下の関係を示す）

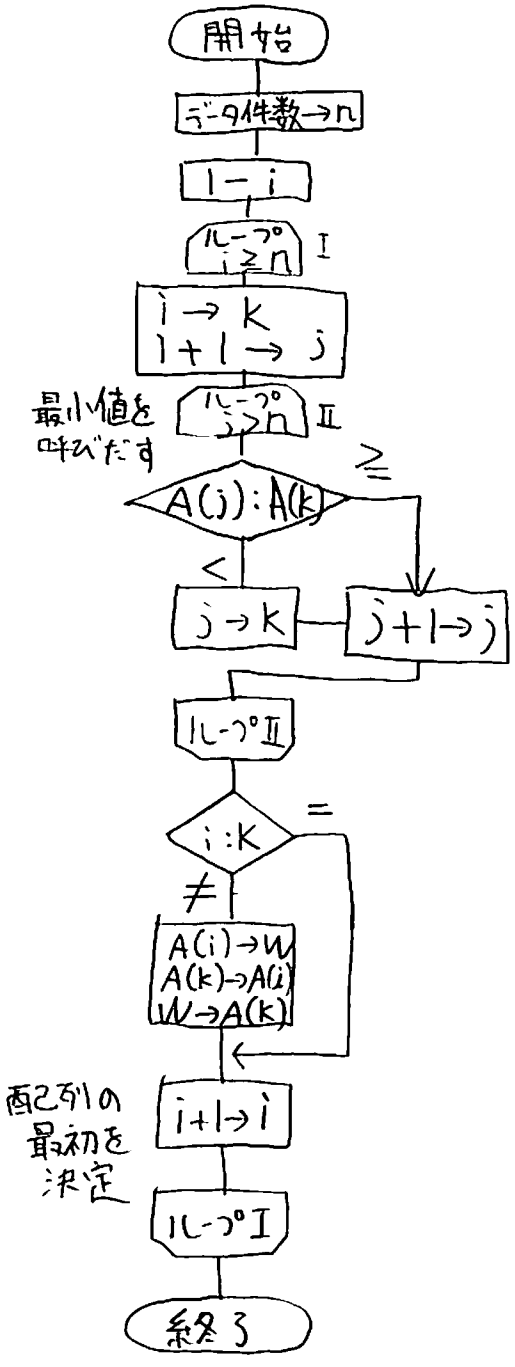
n=8の場合



選択ソート

(選出, ↔ 交換, ○対象

(上から見た下の関係を示す)



Ver_昇順

4 6 2 7 8 3 ① 9
④ 6 2 7 8 3 ① 9
1 6 2 7 8 3 4 9

ループI 1回目終了

1 6 ② 7 8 3 4 9
1 ⑥ ② 7 8 3 4 9
1 2 6 7 8 3 4 9

ループI 2回目終了

1 2 6 7 8 ③ 4 9
1 2 ⑥ 7 8 ③ 4 9
1 2 3 7 8 6 4 9

ループI 3回目終了

1 2 3 7 8 6 ④ 9
1 2 3 ⑦ 8 6 ④ 9
1 2 3 4 8 6 7 9

ループI 4回目終了

1 2 3 4 8 ⑥ 7 9
1 2 3 4 ⑧ ⑥ 7 9
1 2 3 4 6 8 7 9

ループI 5回目終了

1 2 3 4 6 8 ⑦ 9
1 2 3 4 6 ⑧ ⑦ 9
1 2 3 4 6 7 8 9

ループI 6回目終了

1 2 3 4 6 7 ⑧ 9
1 2 3 4 6 7 8 9

ループI 7回目終了

1 2 3 4 6 7 8 ⑨
1 2 3 4 6 7 8 9

ループI 8回目終了

完了 | 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 |

Ver_降順

4 6 2 7 8 3 1 ⑨
④ 6 2 7 8 3 1 ⑨
9 6 2 7 8 3 1 4

ループI 1回目終了

9 6 2 7 ⑧ 3 1 4
9 ⑥ 2 7 ⑧ 3 1 4
9 8 2 7 6 3 1 4

ループI 2回目終了

9 8 2 ⑦ 6 3 1 4
9 8 ② ⑦ 6 3 1 4
9 8 7 2 6 3 1 4

ループI 3回目終了

9 8 7 2 ⑥ 3 1 4
9 8 7 ② ⑥ 3 1 4
9 8 7 6 2 3 1 4

ループI 4回目終了

9 8 7 6 2 3 1 ④
9 8 7 6 ② 3 1 ④
9 8 7 6 4 3 1 2

ループI 5回目終了

9 8 7 6 4 ③ 1 2
9 8 7 6 4 3 1 2

ループI 6回目終了

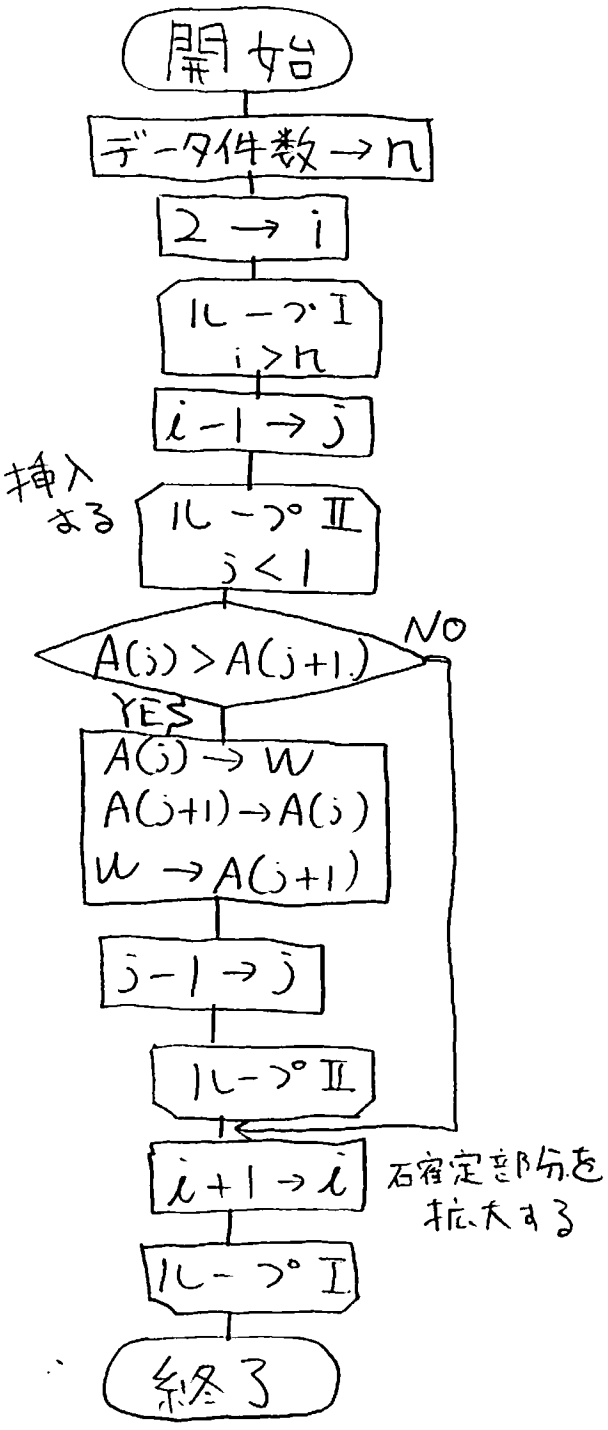
9 8 7 6 4 3 1 ②
9 8 7 6 4 3 ① ②
9 8 7 6 4 3 2 1

ループI 7回目終了

完了 | 9 | 8 | 7 | 6 | 4 | 3 | 2 | 1 |

全体のイメージとして、
最小(又は最大)の値を
見つけ出し、
端(決定されている場所は
除き)に格納。

挿入ソート



全体のイメージとして

バブルソートの

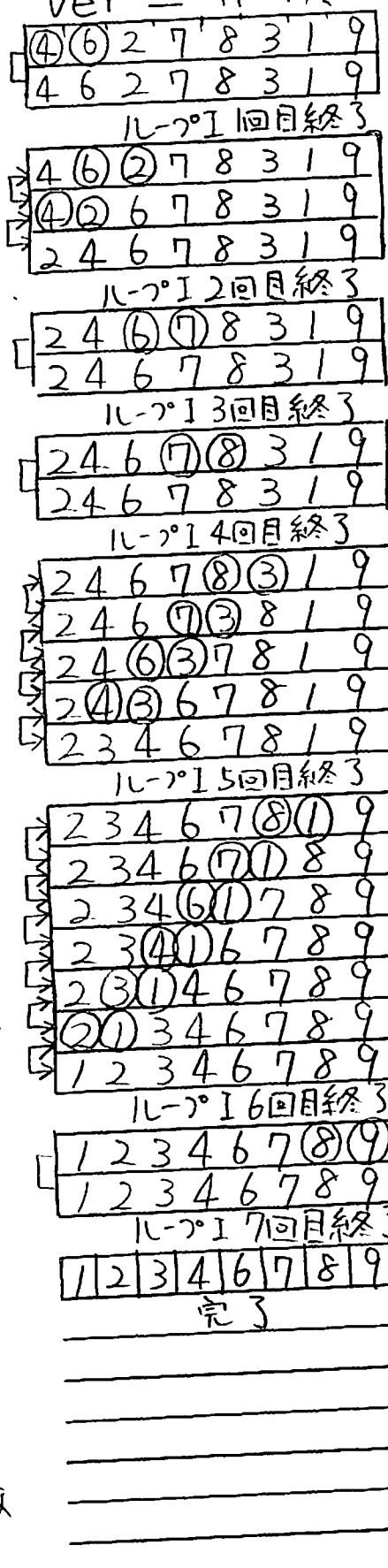
逆バージョンの様な

アルゴリズム

↔ 交換，— 比較，○ 対象

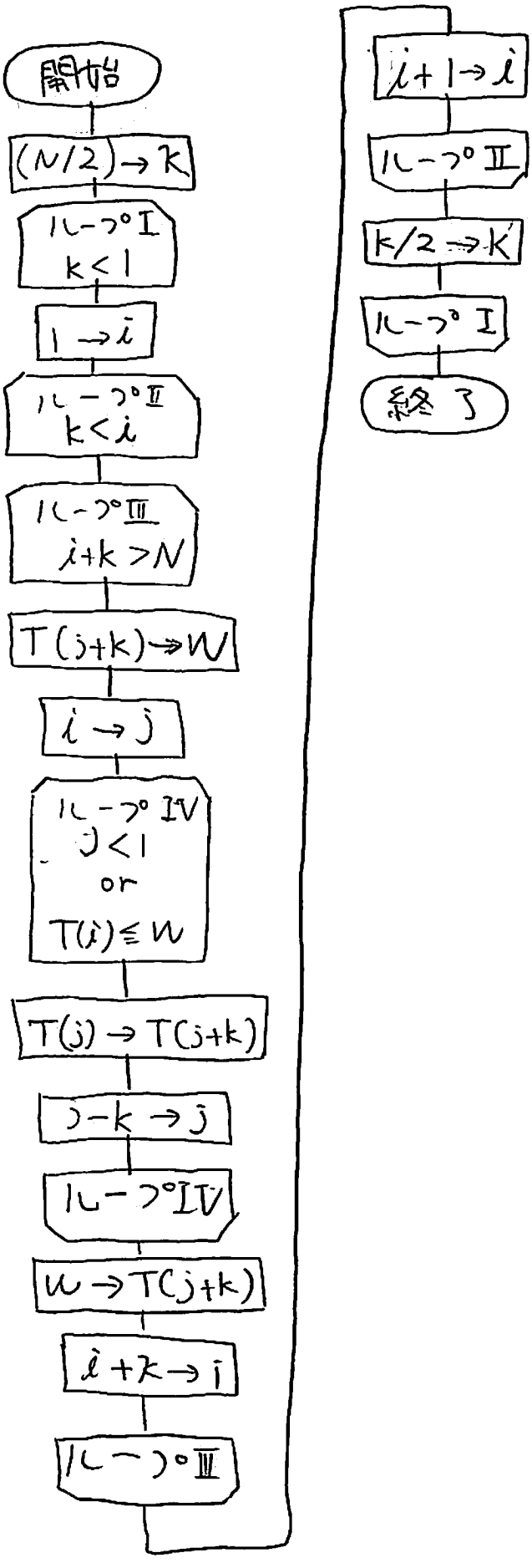
（上から見た下の関係を示す）

Ver_昇順

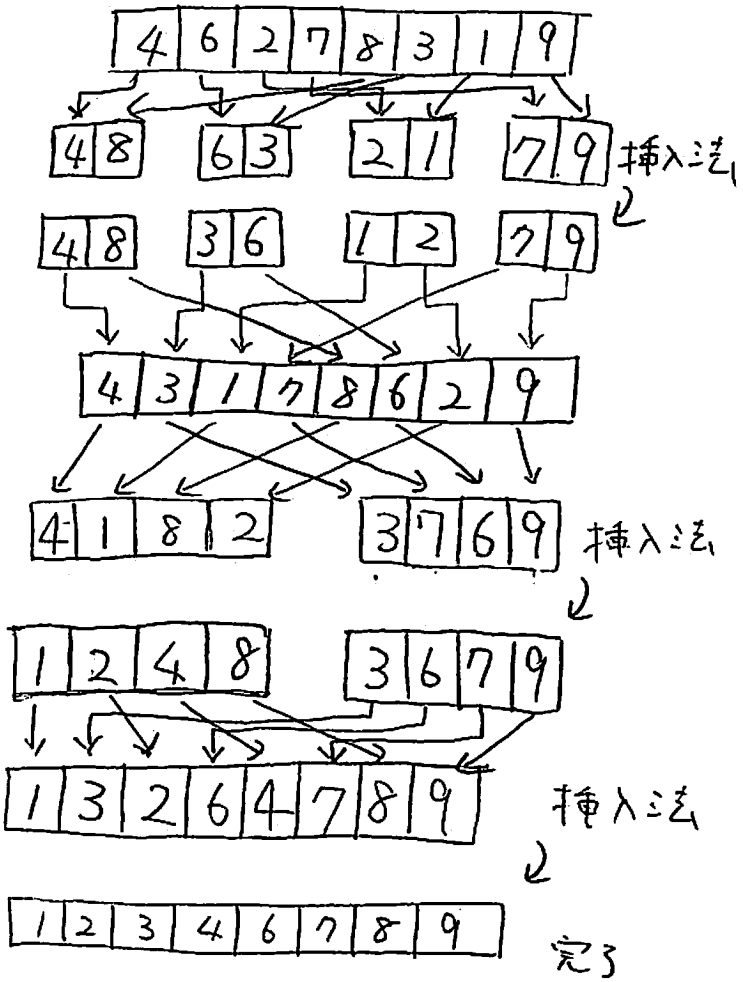


Ver_降順

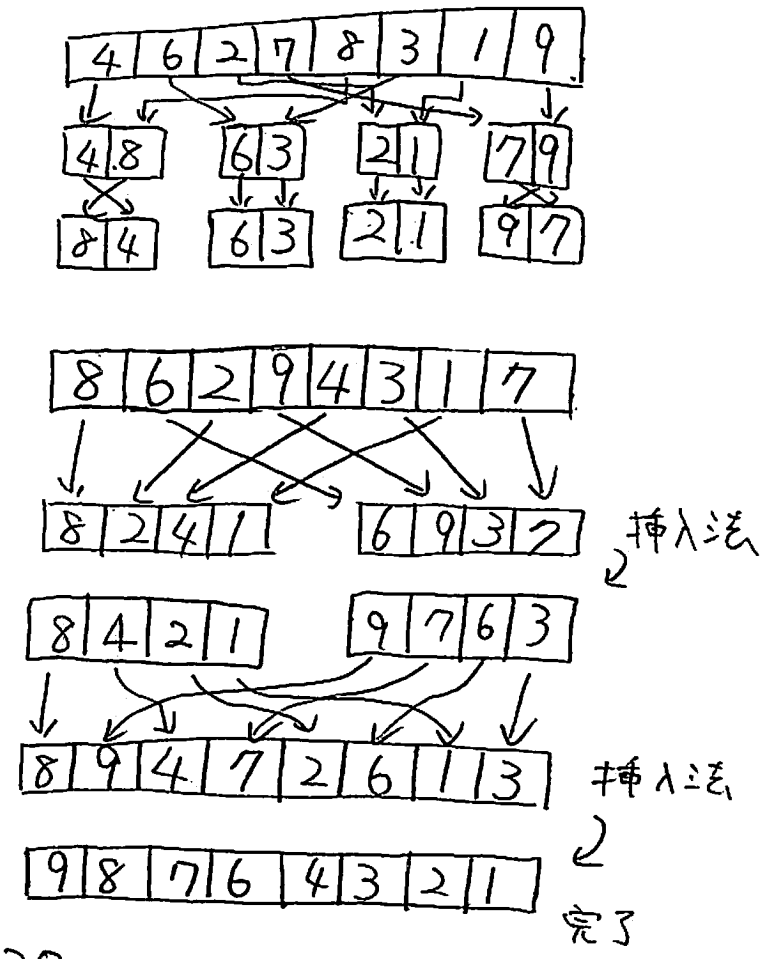
シェルソート



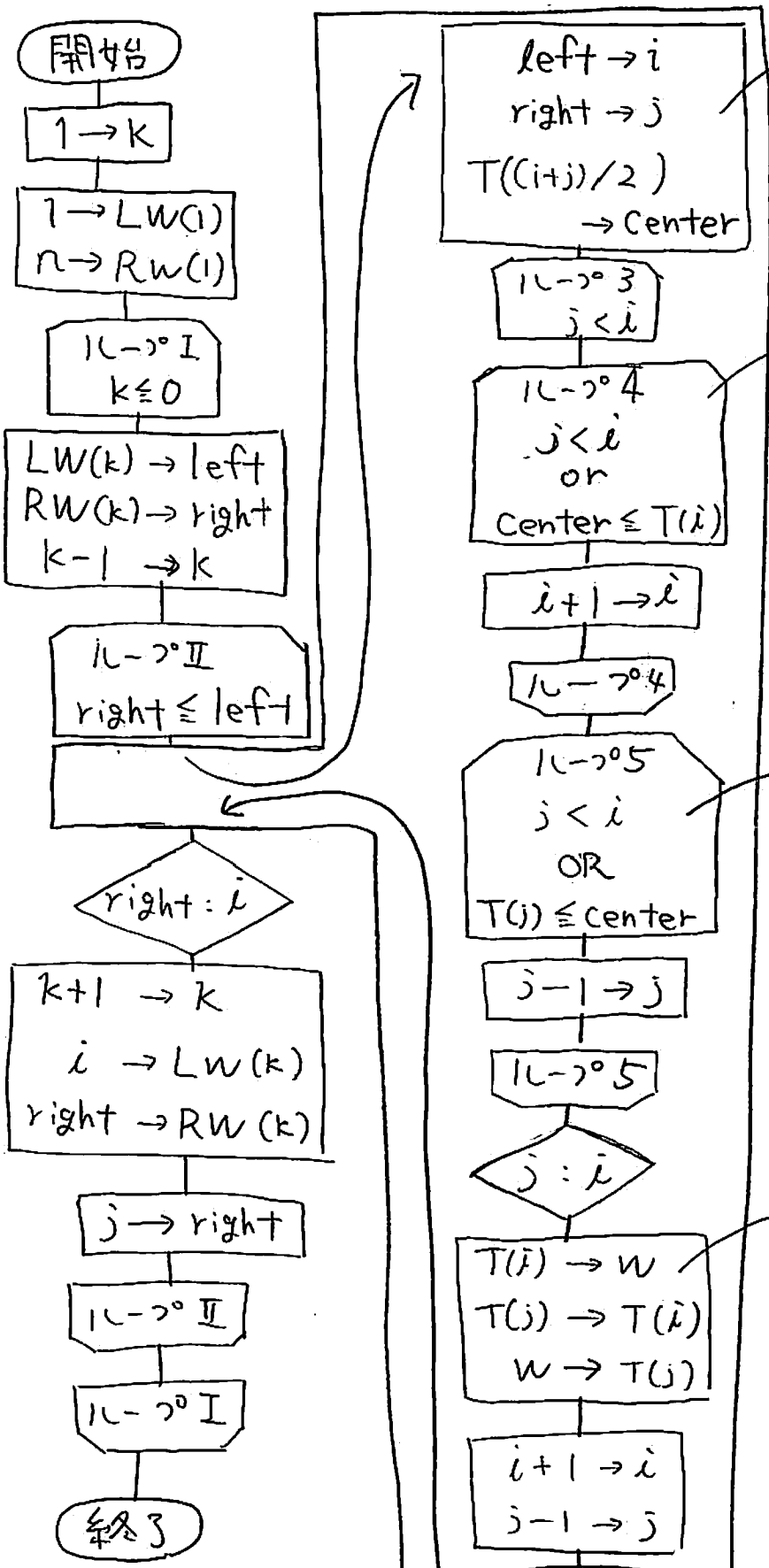
Ver_昇順



Ver_降順



クイックソート



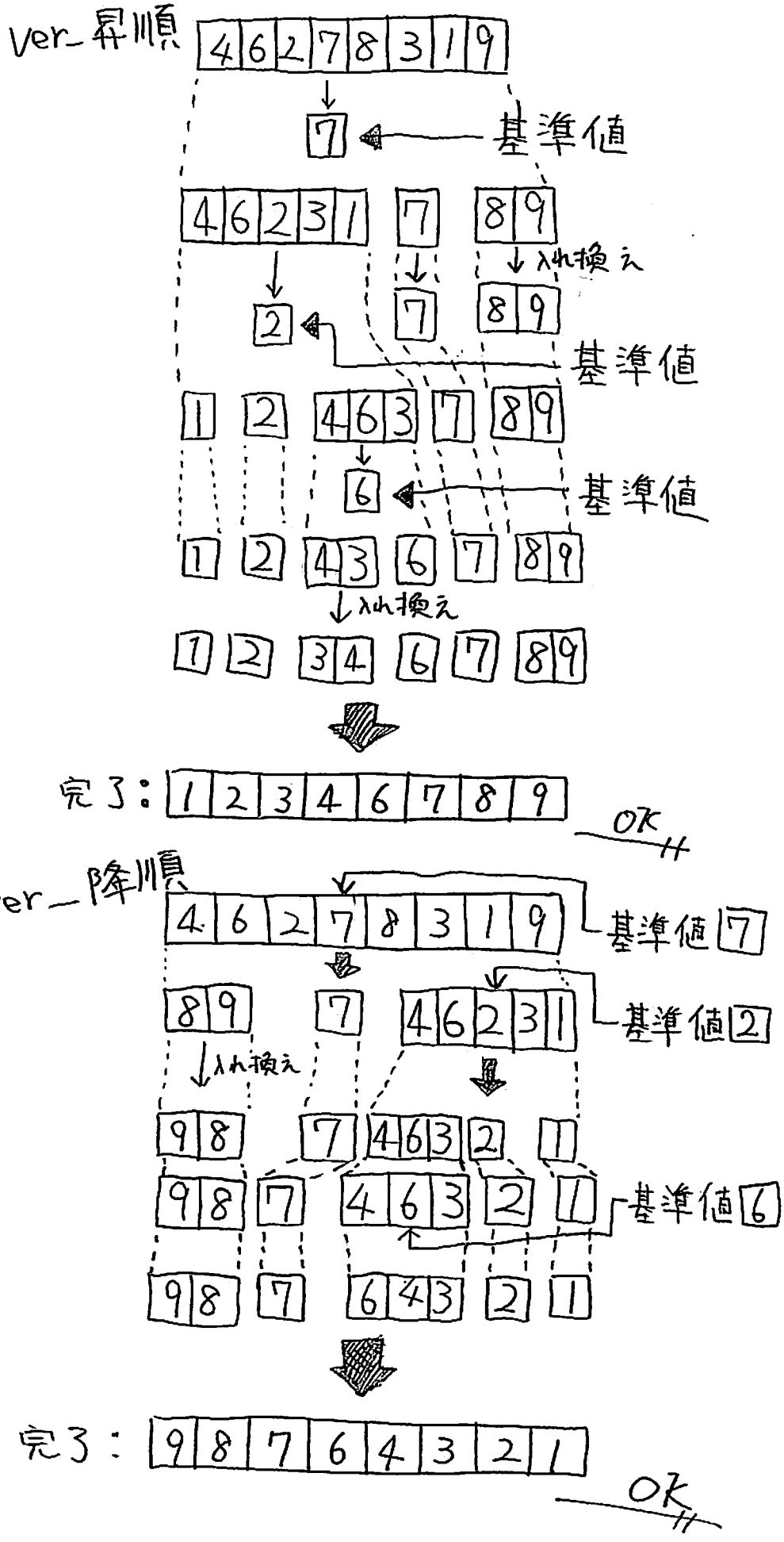
データを振りわけるブロックの基準となる中央位置のセット

ループ4
左データ列で基準値よりも大きなものをサーチ

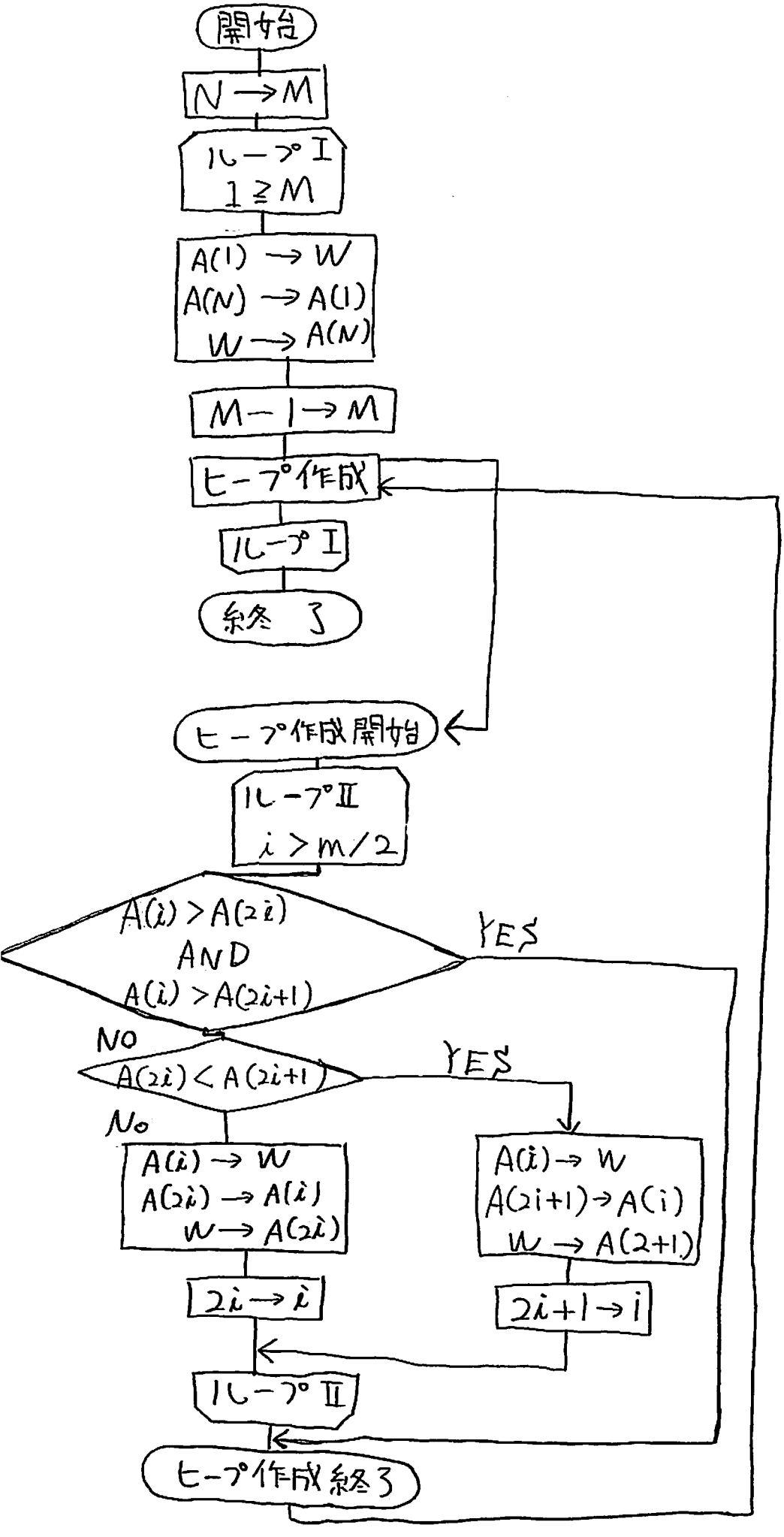
ループ5
右データ列で基準値よりも小さなものをサーチ

交換
サーチによって見つかった基準値の左右で逆順となっているデータを入れ替え

クイックソート
具体例

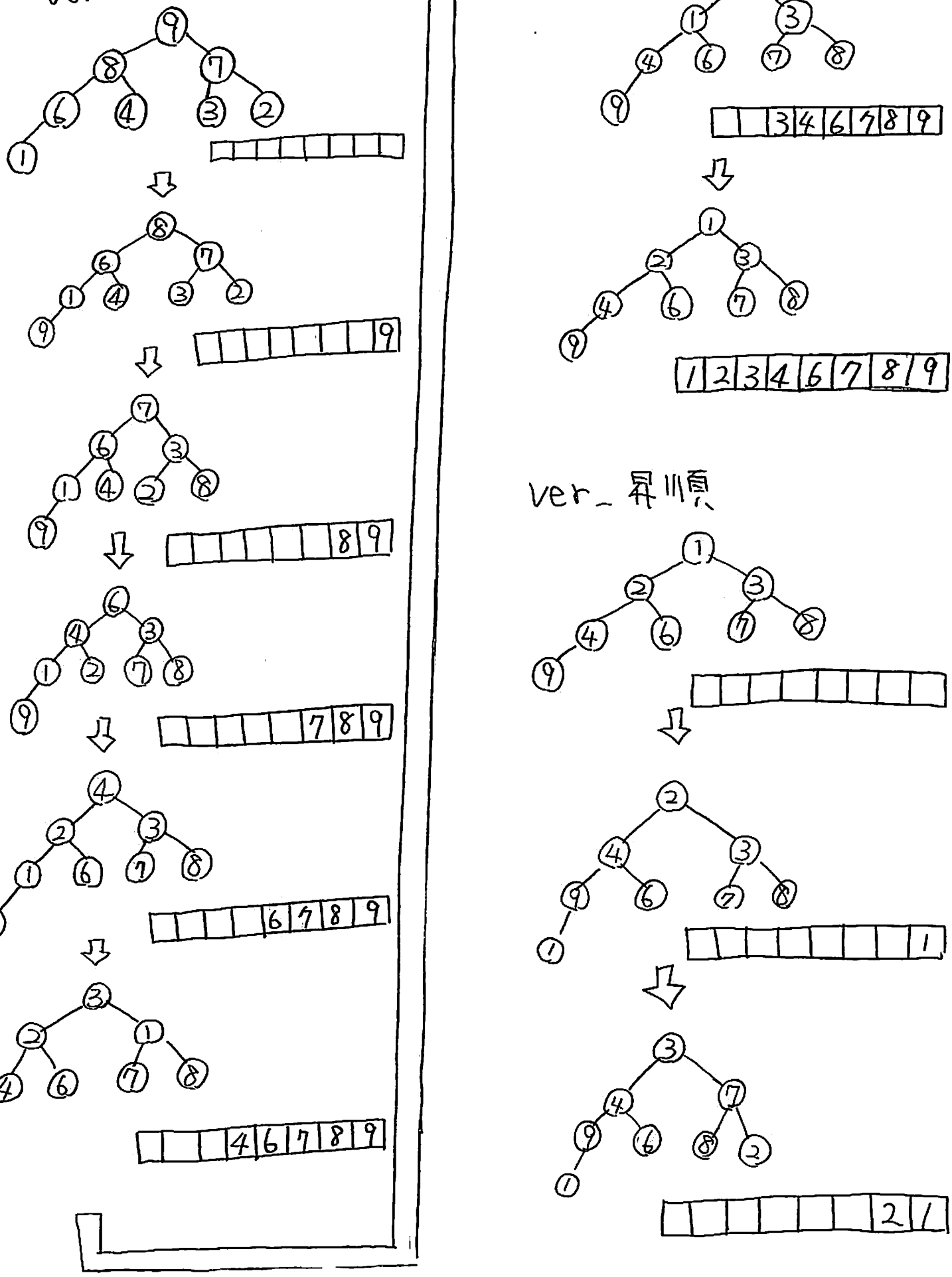

Ver_昇順
基準値
入れ換え
基準値
基準値
入れ換え
完了：
OK
Ver_降順
基準値 7
基準値 2
入れ換え
基準値 6
完了：
OK

ヒープソート


開始
N→M
ループI
1≦M
A(1)→W
A(N)→A(1)
W→A(N)
M−1→M
ヒープ作成
ループI
終了
ヒープ作成開始
ループII
i>m/2
A(i)>A(2i)
AND
A(i)>A(2i+1)
YES
NO
A(2i)<A(2i+1)
YES
No
A(i)→W
A(2i)→A(i)
W→A(2i)
A(i)→W
A(2i+1)→A(i)
W→A(2+1)
2i→i
2i+1→i
ループII
ヒープ作成終了

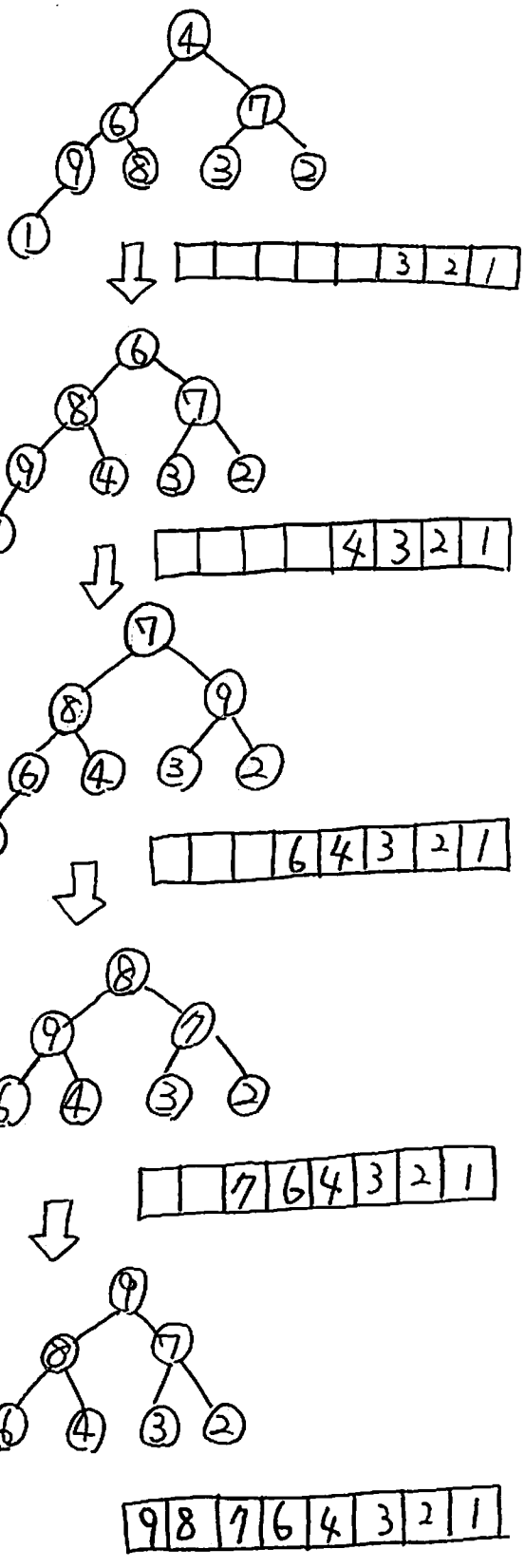
ヒープソート

ver_降順

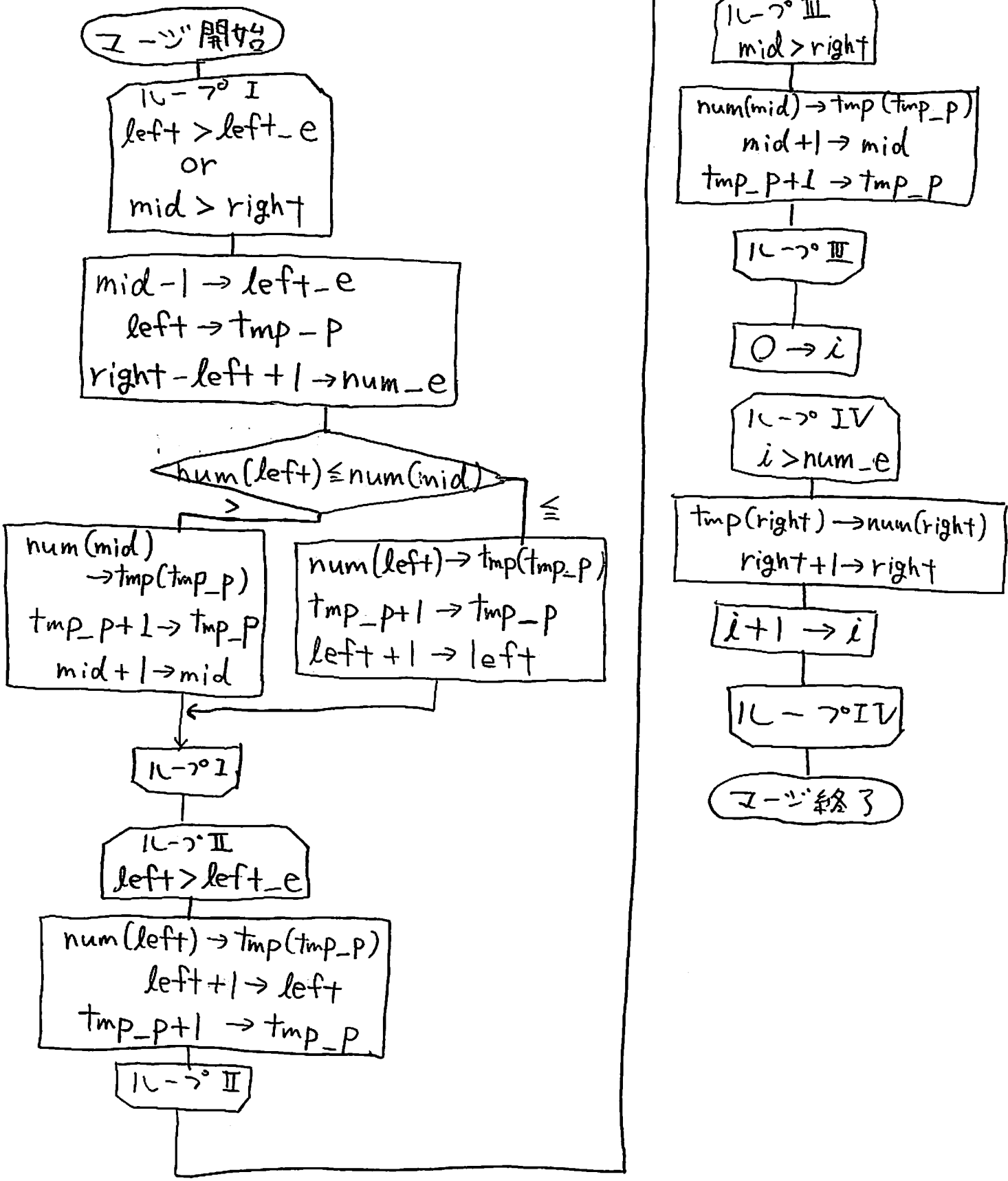


ver_昇順

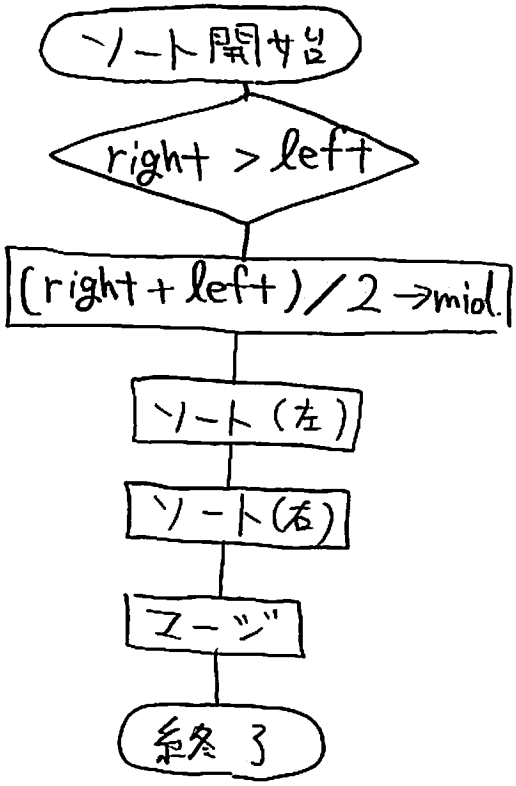
ヒープソート　続き…

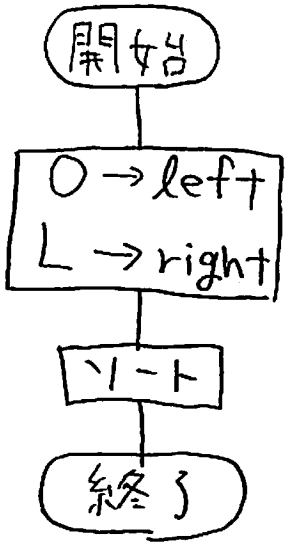

4
6
7
9
8
3
2
1
3 2 1
6
8
7
9
4
3
2
1
4 3 2 1
7
8
9
6
4
3
2
1
6 4 3 2 1
8
9
7
6
4
3
2
1
7 6 4 3 2 1
9
8
7
6
4
3
2
1
9 8 7 6 4 3 2 1

マージソート

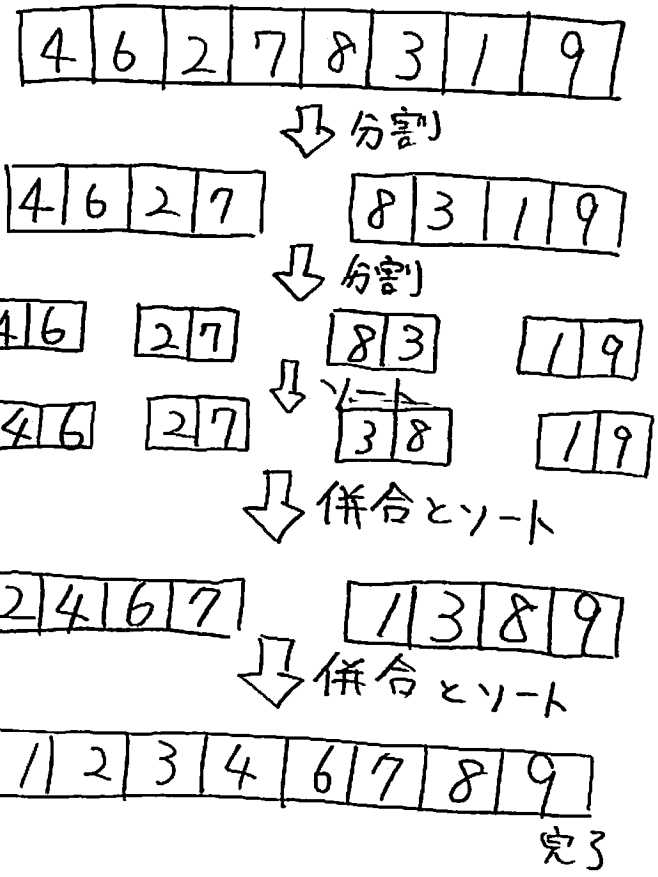

マージ開始
ループ I
left > left_e
or
mid > right
mid-1 → left_e
left → tmp_p
right-left+1 → num_e
num(left) ≦ num(mid)
>
≦
num(mid) → tmp(tmp_p)
tmp_p+1 → tmp_p
mid+1 → mid
num(left) → tmp(tmp_p)
tmp_p+1 → tmp_p
left+1 → left
ループ I
ループ II
left > left_e
num(left) → tmp(tmp_p)
left+1 → left
tmp_p+1 → tmp_p
ループ II
ループ III
mid > right
num(mid) → tmp(tmp_p)
mid+1 → mid
tmp_p+1 → tmp_p
ループ III
0 → i
ループ IV
i > num_e
tmp(right) → num(right)
right+1 → right
i+1 → i
ループ IV
マージ終了

マージソート

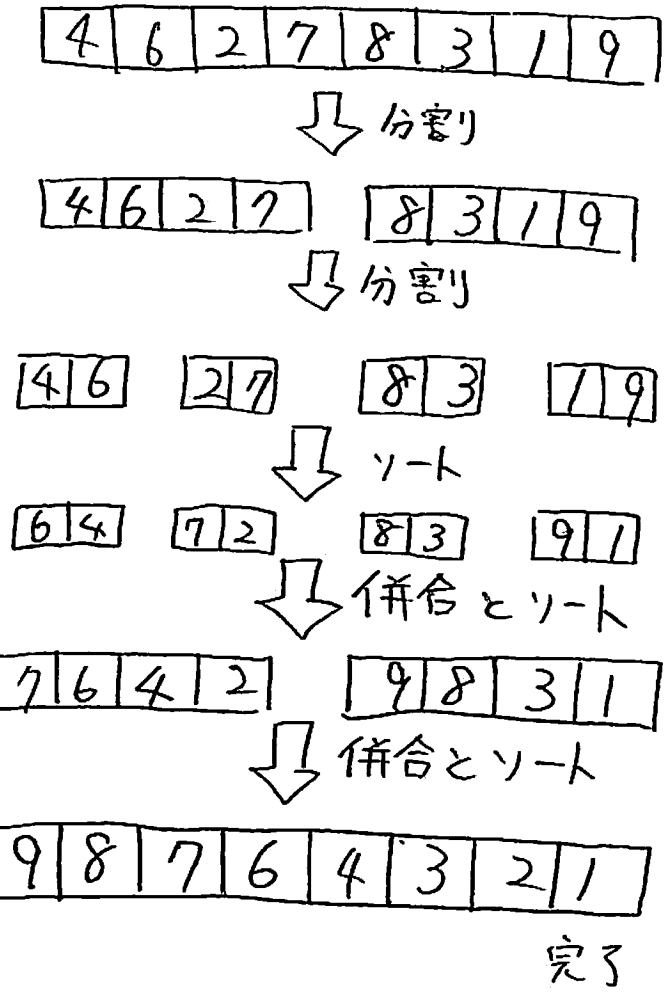




Ver_昇順



Ver_降順



10.
講義で解けなかった問題を解きましょう。問題文は書く必要はないて゛す。答えを見ないて゛解答した答えとなせ゛間違えたかも書きましょう。答えた゛けの場合、再提出にします。

第1章 演習2 並列処理

設問：1.a=カ、b=ア、c=イ…全部間違え。

まず800ドット掛ける600を行い、40万で除算した。結果、1.2となり、何かの単位があると考え『カ』に決定。Bは、コンピュータが4台あると記述されていたので、aの答えを4で割った。結果、『ア』の300を選んだ。Cは、浮動小数点計算に関するものだったので、40万をどうにかして変化させると考えた。そこで、またもや、コンピュータが4台という点に注目し、4で割った。結果『イ』の10を選んだ。

自分で見ても悲惨だ。この問題は二回目なのだが、最初の時と全く一緒の結果だ。その頃は、「まだ習い始めだから大丈夫か」という安心感みたいなものがあったが、さすがに後期の中盤にさしかかってもこの調子ではまずい。

自分は、この文章を読んだとき、理解が全くできなかった。よって時間をかけてもう一回よんだ。しかし、わからない。理由はきっとまだ自分が『ドット』や『浮動小数点』、『ミリ秒』等、今まであまり使わなかった言葉に対応しきれていないからだと思う。

本の解答:aは、プログラムが一回実行される時間を求めればいいと言っているのだから、1,画像データの読み込み、2,1行についての計算部、3,出力結果の合計実行時間をしらべればいい。よって、不明な2の1行についての計算部の実行時間を調べる。
コンピュータの性能を表すMFLOPSと、行数、1行あたりのかかる処理時間の三つから答えをだす。
X=400000×600行÷(200×1000000)
X=1.2(秒)=1200ミリ秒
1200+40+120=1360ミリ秒となる

bは往復300ミリ秒+(2の実行時間÷4)+No.1のコンピュータの1,画像データの読み込み、3,出力結果の合計=答えである。よって
X=300+300+40+120=760となる

第2章 演習2 手続き呼び出し

設問1：a=ウ、b=イ
設問2：ア、イ、ウ…間違い

設問2の『リンク時に』という問題文の意味がよく理解できていなかった。

第3章 演習7

演習7

設問１：a=ア、b=ア…間違い
設問２：c=ア、d=イ、e=カ、f=ウ、g=ウ…c と f が間違え

設問１：
a: Sp と Mp が 1 づつ加算されていることから、繰り返しの条件のナカに Sp または Mp が使用されていると予想。Page[Gp][Mp]←Text[Sp]から文章に関するものだと思い、よって、Dan もしくは Moji のどちらかが使用されていると予想。ここから分からなくなり、『ア』を選択。
理解までの過程：Ep とは何なのかがずっと考えていて分からなかったが、解答を読んでいるときに、隣のページに記載されている事に気づく。
ここで、Page[Gp][Mp]←Text[Sp]の関係を図で考えた。要するに、Ep は Text の行末で、Sp がどんどん加算されてき、（この間に Page[Gp][Mp]に格納）、最後は Ep に到着する。よって考えられるしきは、Ep>=Sp。答えは『カ』となる。

b: Ep が右に一つずれる時の条件で、『Ep が行末じゃないかつ~~~』。~~~の部分は Ep-Sp+1>Moji ならばいいので、移項して Ep<(Sp+Moji-1)、『ウ』となる。

設問２：
c: ただ単純に 2 で割るだけだと思っていたが、空白があるのを忘れていた。空白が二つあるので、Moji-2。二つにわかれているので割る２。よって、(Moji-2)/2 となる
d: 一段目の行の始めを示す。余白を考慮するのを忘れない。よってMp←Yohaku+1
e: 一段目の文字数は Dan に格納されているから空白２つ分、さらに＋1 で、Mp ←Yohaku+Dan+3 の『カ』となる
f: Dp＝１という状態は一段目を出力し、これは変化前を一緒より、『イ』の Gp-1 がはいる
g: Gp と Gyo の関係に気をつける。Gyo は 1 ページの最大行数まである。すると、二段目になっているということは、一段目は Gyo まで使っているということである。よって『ウ』の Gyo がはいる。

演習 8（これは簡単だった）

設問１：a=ウ、b=イ
設問２：

設問１：条件と表を確実に見ていけば分かる。
設問２：これもよくみていけばわかるが、図を書くのが一番いいと思った。

演習 9

設問 1：イのみ…間違え
設問 2：
設問 3：ウ、キ、…最後が分からなかった。

設問 1:オーダ量つまり、"行数に関係なく計算量を一定にする"ということ。ここでは、『付け加える』と『消す』という操作は、その前後関係も考慮しないといけないので、オーダ量が違ってくる。よって答えは、GET0と LAST0。『イ』と『エ』

設問 2：行数の記述がない。よって、LAST は最初から辿っていかないと行けないので、オーダ量に変化がある。DELETE に関して、ポインタを繋ぎ合わせる操作は常に一定なので、OK。GET は設問 1 と同じく OK。INSERT は DELETE と同じ要領で、追加した前後に元のポインタを格納すればよい。(追加位置のポインタは分かっているものとする)

設問 3：
リストを配列で表現した時の操作として、リストを削除する時は、削除されたリストの前後関係のポインタを変更すればよい。この問題では 8 番の要素がなくなるので要素番号 2 と 9 の連結関係が途切れてしまう。よって、要素 2 番目の NEXT を 8 から 9 へ、要素 9 番目の PREV を 8 から 2 へ変更すれば良い。では、NEXT[8]（消された要素）はなくなるのか…なくならない。今度は、空きリストを示すポインタとして作動する。要素番号 8 が削除され、空きリストの先頭になるので、EMPTY が 8 となり、削除したの NEXT が削除する前の空きリストの先頭である 3 を指す事になる。

参考文献：
http://www2.osk.3web.ne.jp/~a0mediac/Argoa/B96a0809/b96a08rf.htm
http://akademeia.info/index.php?%A5%C7%A1%BC%A5%BF%C3%B5%BA%F7

2007 秋基本情報技術者午後問題集
基本情報技術者合格教本