

応用情報技術者試験

テクノロジ系Ⅱ

1-2-2 アルゴリズム

優先度： **最強**

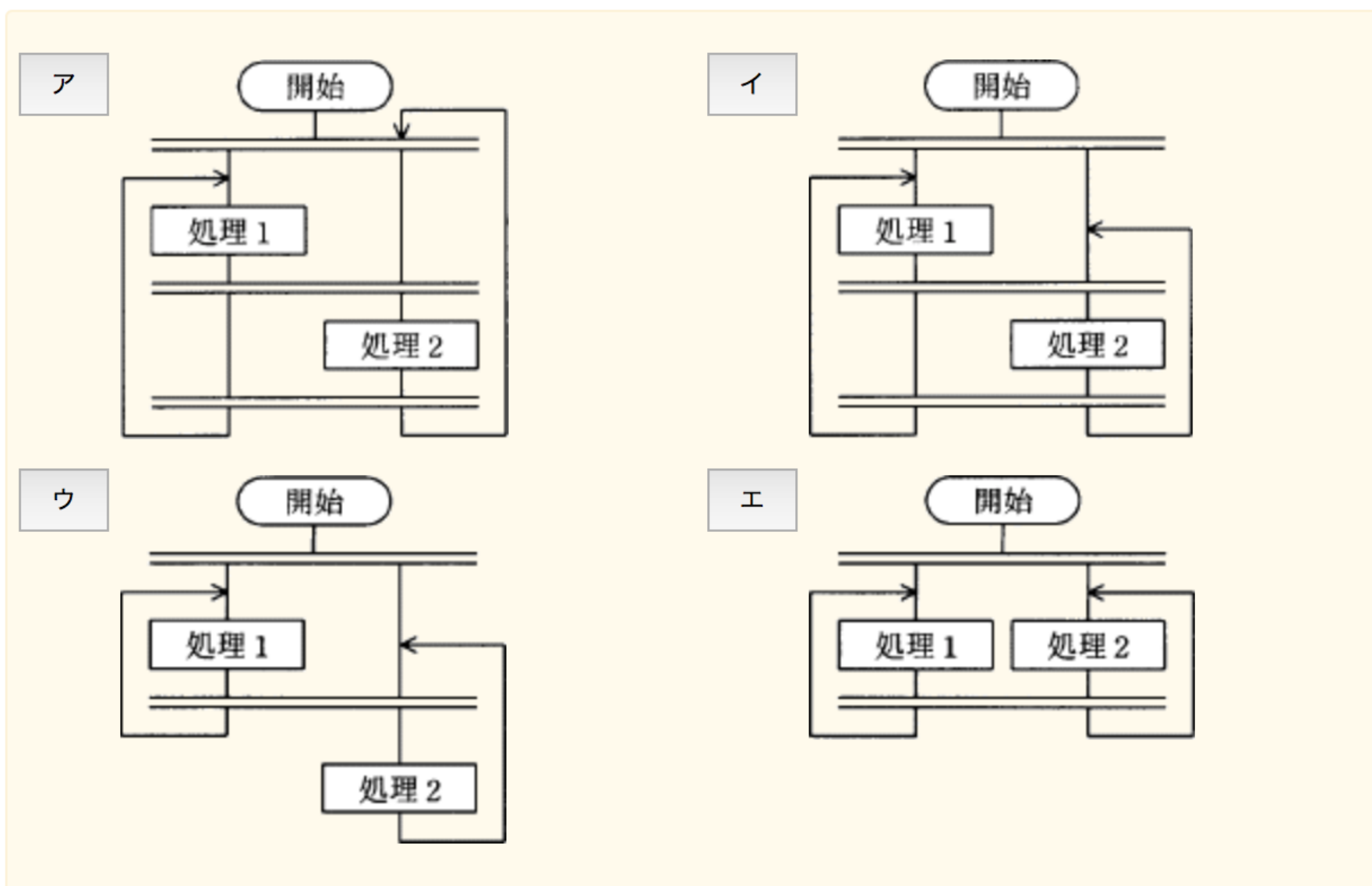
流れ図(フローチャート)



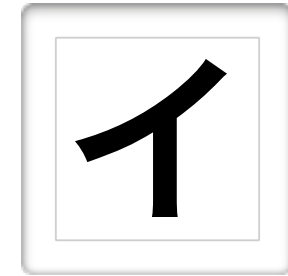
Level 1

次の流れ図のうちで、処理1と処理2が交互に繰り返し実行されるものはどれか。ここで、二重線は並列処理の同期を表す。

出典：平成17年春期 問15



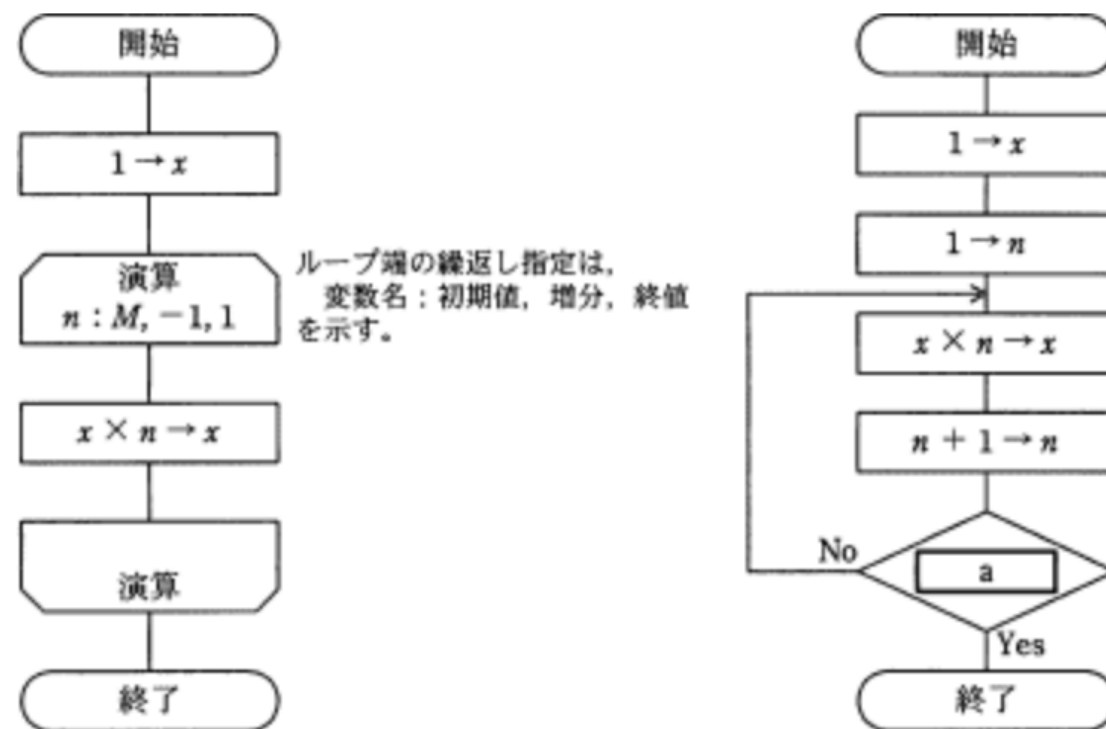
解答



- ア：処理 1 が一番上の同期部分に戻ることができない
- イ：正しい
- ウ：同期→処理 1 →処理 2 →同期→処理 2 となる場合がある
- エ：同期→処理 1 →処理 2 →同期→処理 2 となる場合がある

Level 2

正の整数 M に対して、次の二つの流れ図に示すアルゴリズムを実行したとき、結果 x の値が等しくなるようにしたい。 a に入れる条件として、適切なものはどれか。



出典：平成22年秋期 問7

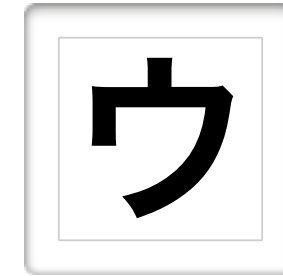
ア $n < M$

イ $n > M - 1$

ウ $n > M$

エ $n > M + 1$

解答



二つのアルゴリズムで共通の M は正の整数という条件が問題文に示されているので、 M に適当な数値を当てはめて解いてみることにします。

数字は何でもいいのですが今回は M に 2 を代入してみます。解き方としてはまず左のアルゴリズムを解き、その結果の x の値が右の式でも出力される条件はどれかを考えていくことになります。

[左の流れ図]

```
<<開始>>
1 → x                      //x=1
ループ条件 n:2 増分 -1、n=1で終了
x×n → x (1×2 →x)          //x=2
ループ先頭に戻る n-1 → n    //n=1
n=1 になったのでループ終了
<<終了>>                    //x=2
```

左の流れ図をトレースしてみると、結果 x の値は 2 になることがわかります。

[右の流れ図]

```
<<開始>>
1→x                        //x=1
1 → n                      //n=1
x×n → x(1 × 1 →x) //x=1
n+1 → n (1+1 →n)//n=2
```

どの選択肢の条件でもここまでは同じです。後の部分は選択肢の条件ごとに結果 x を見ていきましょう。

ア : “ $n < M$ ”

(分岐条件 $n < M$)

$n < M$ ($2 < 2$) は No →ループ先頭へ戻る。以後ループ内で n の値は加算されていくだけなので ($n < M$)の条件を満たすことは永遠になく無限ループとなってしまいます。

イ : “ $n > M-1$ ”

(分岐条件 $n > M-1$)

$n > M-1$ ($2 > 2-1$) は Yes →流れ図は終了する。 結果 x の値は 1 で、左の流れ図とは異なる値となる。

ウ : “ $n > M$ ”

(分岐条件 $n > M$)

$n > M$ ($2 > 2$) は No →ループ先頭へ戻る。 //x=1

$x \times n \rightarrow x$ ($1 \times 2 \rightarrow x$) //x=2

$n+1 \rightarrow n$ ($2+1 \rightarrow n$) //n=3

$n > M$ ($3 > 2$) は Yes →流れ図は終了する。//x=2

結果 x の値は 2 で、**左の流れ図と同じ値**になる。

エ : “ $n > M+1$ ”

(分岐条件 $n > M+1$)

$n > M+1$ ($2 > 2+1$) は No →ループ先頭へ戻る。 //x=1

$x \times n \rightarrow x$ ($1 \times 2 \rightarrow x$) //x=2

$n+1 \rightarrow n$ ($2+1 \rightarrow n$) //n=3

$n > M+1$ ($3 > 2+1$) は No →ループ先頭へ戻る。 //x=2

$x \times n \rightarrow x$ ($2 \times 3 \rightarrow x$) //x=6

$n+1 \rightarrow n$ ($3+1 \rightarrow n$) //n=4

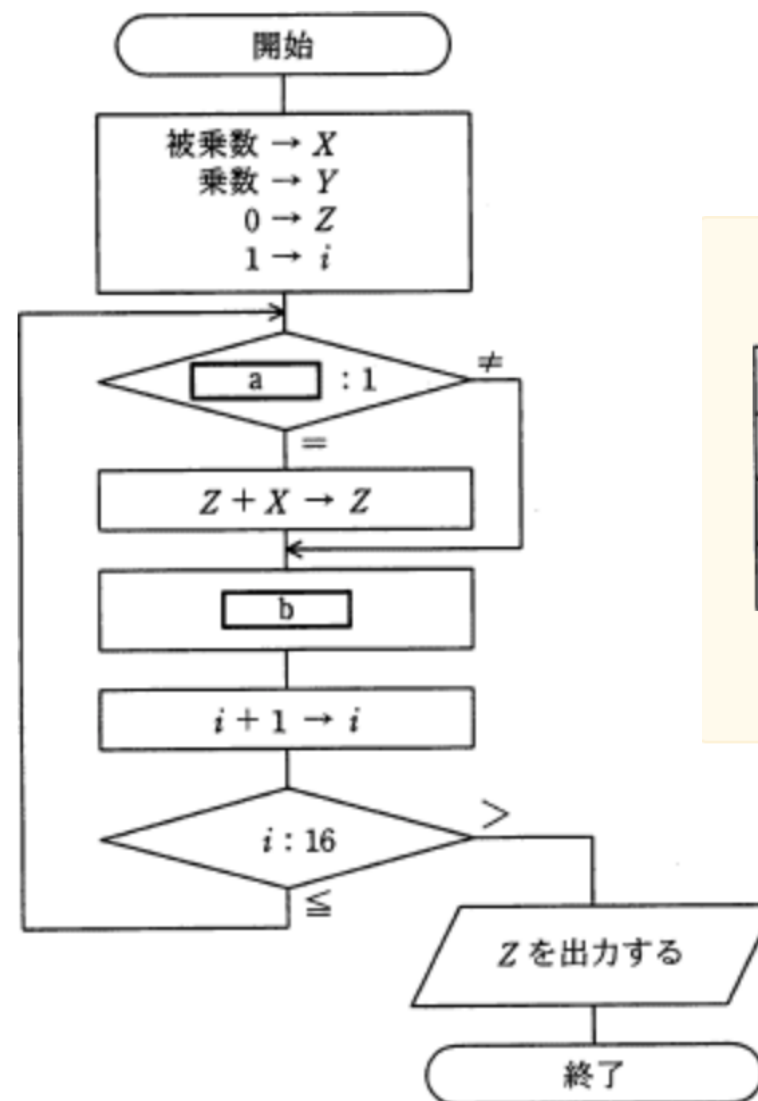
$n > M+1$ ($4 > 2+1$) は Yes →流れ図は終了する。//x=6

結果 x の値は 6 で、左の流れ図とは異なる結果となる。

以上の結果から二つのアルゴリズムの結果を同じにする条件として「ウ」が適切であるということになります。

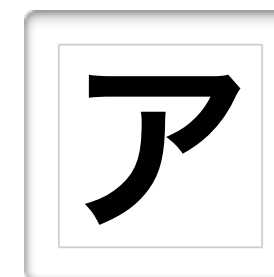
Level 3

次の流れ図は、シフト演算と加算の繰り返しによって2進数の乗算を行う手段を表したものである。この流れ図の中のa, bの処理の組合せとして、正しいものはどれか。ここで、乗数と被乗数は符号なしの16ビットで表される。X, Y, Zは32ビットのレジスタであり、けた送りには論理シフトを用いる。最下位ビットを第0ビットを記す。



	a	b
ア	Yの第0ビット	Xを1ビット左シフト, Yを1ビット右シフト
イ	Yの第0ビット	Xを1ビット右シフト, Yを1ビット左シフト
ウ	Yの第15ビット	Xを1ビット左シフト, Yを1ビット右シフト
エ	Yの第15ビット	Xを1ビット右シフト, Yを1ビット左シフト

解答



XおよびYにトレースが簡単な数値を設定して、出力Zが適切な値かどうかで正しい組合せを導きます。ここでは $X=3$ 、 $Y=3$ とすることにします。X,Yともにビット表記では011です。またあるビット列を左に1ビットシフトすると2倍、右に1ビットシフトすると $1/2$ したのと同じになります。(ビットあふれは無視します)

ア1: Yの0ビット目は1なので、 $Z \leftarrow (Z+X)$ //Z=3

2. Xを1ビット左シフト //X=6

3. Yを1ビット右シフト //Y=1

4. $i \leftarrow (i+1)$ //i=1

5. Yの第0ビットは1なので、 $Z \leftarrow (Z+X)$ //Z=9

6. Xを1ビット左シフト //X=12

7. Yを1ビット右シフト //Y=0

8. 以後、Yの第0ビットは $i>16$ になるまで0なので、加算は行われずにループ終了

9. Zを出力する //Z=9

イ1: Yの第0ビットは1なので、 $Z \leftarrow (Z+X)$ //Z=3

2. Xを1ビット右シフト //X=1

3. Yを1ビット左シフト //Y=6

4. $i \leftarrow (i+1)$ //i=1

5. Yを左にシフトすると第0ビットに0が挿入される。以後、Yの第0ビットは $i>16$ になるまで常に0なので、加算は行われずにループ終了

6. Zを出力する //Z=3

ウ1: Yの第15ビットは0なので加算はなし

2. Xを1ビット左シフト //X=6

3. Yを1ビット右シフト //Y=1

4. $i \leftarrow (i+1)$ //i=1

5. Yを右にシフトすると第15ビットには0が挿入される。以後、Yの第15ビットは $i>16$ になるまで常に0なので、加算は行われずにループ終了

6. Zを出力する //Z=0

エ1: Yの第15ビットは0なので加算はなし

2. Xを1ビット右シフト //X=1

3. Yを1ビット左シフト //Y=6

4. $i \leftarrow (i+1)$ //i=1

5. Yの第15ビットは0なので加算はなし

6. Xを1ビット右シフト //X=0

7. Yを1ビット左シフト //Y=12

8. Xが0になったので、加算処理が行われたとしてもZは0のままでループ終了

9. Zを出力する //Z=0

唯一、正しい結果を得られる「ア」の組合せが適切とわかります。

このアルゴリズムを一言で言うならば、Yの第nビットが1であれば、Zに $X \times 2^n$ を加算する処理を繰り返すものということになります。

探索アルゴリズム

- 1 : 線形探索
- 2 : 2分探索
- 3 : ハッシュ表探索



線形探索

- データを先頭から順に探索していく単純なアルゴリズム
- 平均すると大体真ん中くらいで見つかると考えられる
- n 個のデータで探索を行う場合
 - * 平均探索回数は $n / 2$ 回, 計算量は $O(n)$ となる

2分探索

- データをあらかじめ整列させておき, 最初に真ん中のデータと探索するデータを比較する
- 2つのデータの関係から前後どちらのグループに目的のデータがあるかを予測し, そのグループの真ん中のデータと比較
- 計算量は $O(\log n)$

探索されるデータ

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

探索するデータ

3

①	$5 > 3$
②	$3 > 2$
③	$3 = 3$

ハッシュ表探索

- ハッシュ関数を利用し, データからハッシュ値を求めることによって探索する

例

ハッシュ関数 $h(x) = x \% 5$
データの格納場所を5つ用意

探索するデータが「7」の時
 $h(7) = 7 \% 5 = 2$
ハッシュ値が「2」の場所を見ると
データ「7」が見つかる

データ量に関係なく計算量は $O(1)$
ただし違うデータでハッシュ値が重なる**シノニム**
という問題が発生することがある

ハッシュ値	データ
0	25
1	11
2	7
3	13
4	4

探索の問題

探索表の構成法を例とともに a～c に示す。探索の平均計算量が最も小さい探索手法の組合せはどれか。ここで、探索表のコードの空欄は表の空きを示す。

a コード順に格納した探索表

コード	データ
120380	
120381	
120520	
140140	

b コードの使用頻度順に格納した探索表

コード	データ
120381	
140140	
120520	
120380	

c コードから一意に決まる場所に格納した探索表

コード	データ
120381	
120520	
140140	
120380	

	a	b	c
ア	2分探索	線形探索	ハッシュ表探索
イ	2分探索	ハッシュ表探索	線形探索
ウ	線形探索	2分探索	ハッシュ表探索
エ	線形探索	ハッシュ表探索	2分探索

ア

イ

ウ

エ

解答

ア

[a コード順に格納した探索表]

探索表はコード順に整列済みなので、線形探索または2分探索を使用可能です。各探索法の平均探索回数は、線形探索が $(N+1)/2$ 回、2分探索法が $\lceil \log_2 N \rceil$ 回ですので、探索表の要素数が同じならば平均探索回数は2分探索のほうが少なくて済みます。したがってa表には**2分探索**が適切です。(*[n]はnを超えない最大の整数を表します)

[b コードの使用頻度順に格納した探索表]

使用頻度が高いデータほど探索表の先頭のほうに位置することになります。線形探索では探索表の先頭から順番に探索していくので、このような探索表に対しては効率的に探索できます。

2分探索法は、データが整列されていないと使えないという制限がありますし、この表はハッシュ法に対応していないので、b表に対しては**線形探索**が唯一使用できる方法となります。

[c コードから一意に決まる場所に格納した探索表]

ハッシュ法は、探索データのキー値から、そのデータの格納場所(アドレス)を直接計算する方法で、(シノニムが発生しなければ)1回の計算で一意に目的のデータにたどりつけます。

1回の探索でいいので、このc表に対して最も計算量が少なくなる探索法は**ハッシュ表探索**です。

したがって、**a=2分探索**、**b=線形探索**、**c=ハッシュ表探索**の組合せが正解です。

補足：シノニムの問題

自然数をキーとするデータを，ハッシュ表を用いて管理する。キー x のハッシュ関数 $h(x)$ を

$$h(x) = x \bmod n$$

とする。ここで， n はハッシュ表の大きさであり， $x \bmod n$ は x を n で割った余りを表す。

キー a と b が衝突する条件はどれか。

出典：平成19年秋期 問12

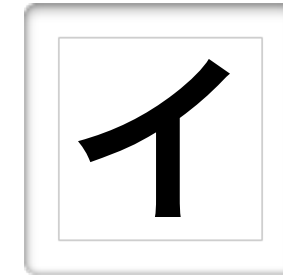
ア $a + b$ が n の倍数

イ $a - b$ が n の倍数

ウ n が $a + b$ の倍数

エ n が $a - b$ の倍数

解答



キーaのハッシュ関数は $a \bmod n$ ，キーbのハッシュ関数は $b \bmod n$ です。この2つが一致する場合を方程式で表すと

$$a \bmod n = b \bmod n$$

$$(a \bmod n) - (b \bmod n) = 0$$

$$(a - b) \bmod n = 0$$

$a - b$ が n で割りきれるときにこの等式が成り立つことがわかるので、衝突する条件は「 $a - b$ が n の倍数」ということになります。

整列アルゴリズム

- 1 : バブルソート
- 2 : 挿入ソート
- 3 : 選択ソート
- 4 : クイックソート
- 5 : シェルソート
- 6 : ヒープソート
- 7 : マージソート



整列アルゴリズム

- バブルソート

- * 隣り合う要素を比較して、大小の順が逆であれば、その要素を入れ替える操作を繰り返すアルゴリズム (計算量は $O(n^2)$)

- 挿入ソート

- * 整列された列に、新たに要素を1つずつ適切な位置に挿入する操作を繰り返すアルゴリズム (計算量は $O(n^2)$)

- 選択ソート

- * 未整列の部分列から最大値 (または最小値) を検索し、それを繰り返すことで整列させていくアルゴリズム (計算量は $O(n^2)$)

- クイックソート

- * 最初に中間的な基準値を決めて、それよりも大きな値を集めた部分列と小さな値を集めた部分列に要素を振り分け、部分列の中で基準値を決めて同様の操作をするアルゴリズム (計算量は $O(n \log n)$)

整列アルゴリズム

- シェルソート

- * ある一定間隔おきに取り出した要素からなる部分列をそれぞれ整列させ，さらに間隔を狭めて同様の操作を繰り返し，最後に間隔を1にして完全に整列させるアルゴリズム(計算量は $O(n \log n)$)

- ヒープソート

- * 未整列部分でヒープを構成し，その根から最大値 (または最小値)を取り出して整列済みの列に移すという操作を繰り返して，未整列部分をなくしていくというアルゴリズム(計算量は $O(n \log n)$)

- マージソート

- * 未整列のデータ列を前半と後半に分ける分割操作を，これ以上分割できない，大きさが1の列になるところまで繰り返す。その後，分割した前半と後半をマージ (併合)して，整列済みのデータ列を作成することを繰り返し，最終的に全体をマージするアルゴリズム (計算量は $O(n \log n)$) : メモリを多く消費するのが欠点

Level 1

データ列が整列の過程で図のように上から下に推移する整列方法はどれか。ここで、図中のデータ列中の縦の区切り線は、その左右でデータ列が分割されていることを示す。

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

出典：平成26年秋期 問6

ア クイックソート

イ シェルソート

ウ ヒープソート

エ マージソート

解答

エ

図の整列過程を見ると、大きさが1の部分文字列の併合を繰り返して最終的な整列済み文字列にしていることがわかります。このように整列対象を大きさが1の部分文字列に分割した後、隣り合う要素ごとに整列と併合を繰り返しながら整列を行う手法がマージソートです。

ア：“クイックソート”

クイックソートは、適当な基準値を選び、それより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整列を行う手法です。

イ：“シェルソート”

シェルソートは、ある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整列手法です。

ウ：“ヒープソート”

ヒープソートは、未整列の部分を順序木にし、そこから最小値(または最大値)を取り出して整列済の部分に移す。この操作を繰り返して、未整列の部分を縮めていく整列手法です。

エ：“マージソート”

正しい。

Level 2

n個のデータを整列するとき、比較回数が最悪の場合で $O(n^2)$ 、最良の場合で $O(n)$ となるものはどれか。

出典：平成19年春期 問11

ア クイックソート

イ 単純選択法

ウ 単純挿入法

エ ヒープソート

解答

ウ

オーダ記法は、アルゴリズムの計算量が実行時に処理するデータ量によってどのように増加するかやアルゴリズムの実行時間の長さを示します。時間計算量を表すオーダ記法によってアルゴリズムの複雑さがわかり、アルゴリズムの理論的比較することができます。

それぞれの整列法の比較回数は次のようになっています。

	最良	平均	最悪
クイックソート	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
単純選択法	$O(n^2)$	$O(n^2)$	$O(n^2)$
単純挿入法	$O(n)$	$O(n^2)$	$O(n^2)$
ヒープソート	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

したがって最悪の場合で $O(n^2)$ 、最良で $O(n)$ となるアルゴリズムは「単純挿入法」が適切です。

Level 3

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き、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た結果を表す。

〔手順〕

(1) : $[\text{データ数} \div 3] \rightarrow H$ 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。

出典：平成24年春期 問7

ア 2

イ 3

ウ 4

エ 5

解答

ア

手順に沿って整列を行うと次のようになります。

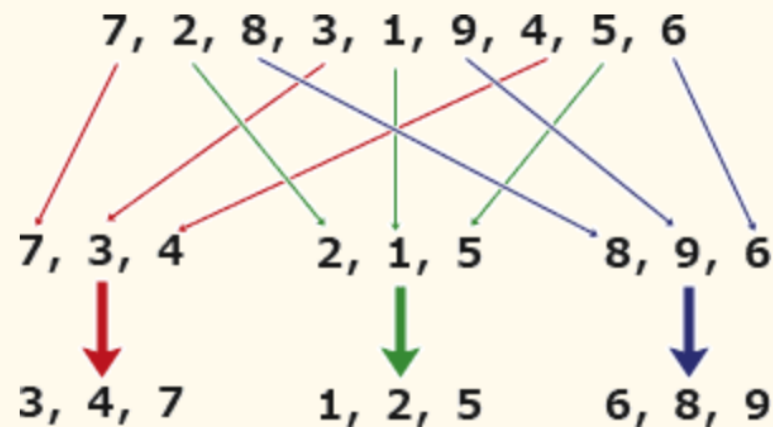
(1) : $H \leftarrow 9 \div 3 = 3$

(2) : 要素ごとが互いに3つずつ離れた要素から成る3つの部分文字列に分解し、それぞれ整列を行います。

[部分列1] 7, 3, 4 $\rightarrow$ (整列) $\rightarrow$ 3, 4, 7

[部分列2] 2, 1, 5 $\rightarrow$ (整列) $\rightarrow$ 1, 2, 5

[部分列3] 8, 9, 6 $\rightarrow$ (整列) $\rightarrow$ 6, 8, 9



(3) : $H \leftarrow 3 \div 3 = 1$ //1回目

(4) : $H \neq 0$ なので2に戻る

(5) : 要素ごとが互いに1つずつ離れた要素から成る3, 4, 7, 1, 2, 5, 6, 8, 9を整列し、1, 2, 3, 4, 5, 6, 7, 8, 9 とします。

(6) : $H \leftarrow 1 \div 3 = 0$ (小数点未満切り捨て) //2回目

この時点でHが0になるためデータ列の整列は完了します。したがって(3)の処理が繰り返される回数は2回です。