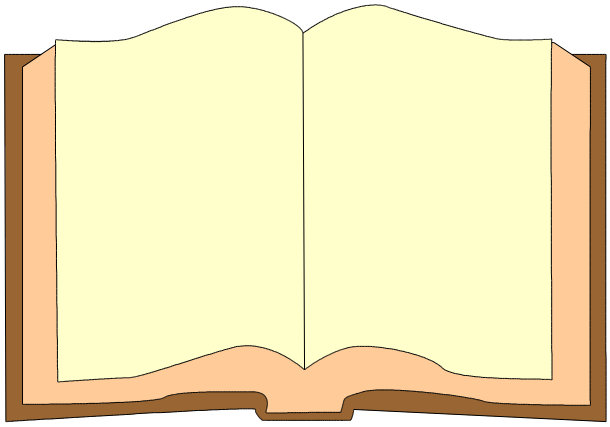
KMC基本情報受験学習講座シリーズ

基礎理論

データ構造



平成２８年１１月１日

加藤　正夫

データ構造目次

配列

リスト

スタックと待ち行列

二分木

バランス木とヒープ木

配列

◆定義と特徴

◇配列の定義と表現

配列は同じデータ型、同じ大きさのデータが物理的に連続して並んでいるデータ構造をもつ。

配列名を有し、個々のデータを表す配列の要素で構成され、定義された一定の大きさをもつ。配列の特定の要素は配列名の後ろに(　)とその中に添字を記入して表す。配列名をＴＢＬ、先頭の添字が１の場合、Ｉ番目の添字の要素はＴＢＬ(Ｉ)で表す。

◇配列の要素

配列の要素に含まれる各データを添字によって特定できる。添字は配列の先頭から何番目の要素であるかを表している。添字の先頭は０または１から始まる。大きさ７の配列ＴＢＬ(Ｉ)の各要素にＡ、Ｂ、Ｃ、…、Ｇの各文字を格納した例を示す。

添字	1	2	3	4	5	6	7
配列ＴＢＬ(Ｉ)	A	B	C	D	E	F	G

配列の大きさ７

配列の要素の大きさは一定であるから、配列の先頭アドレスが決まると、各要素のアドレスが決まり、各要素のアドレスが分かると先頭から何番目の要素であるかを決めることができる。

◆探索の方法

◇配列を利用した探索法

配列の探索法には、次のものがある。

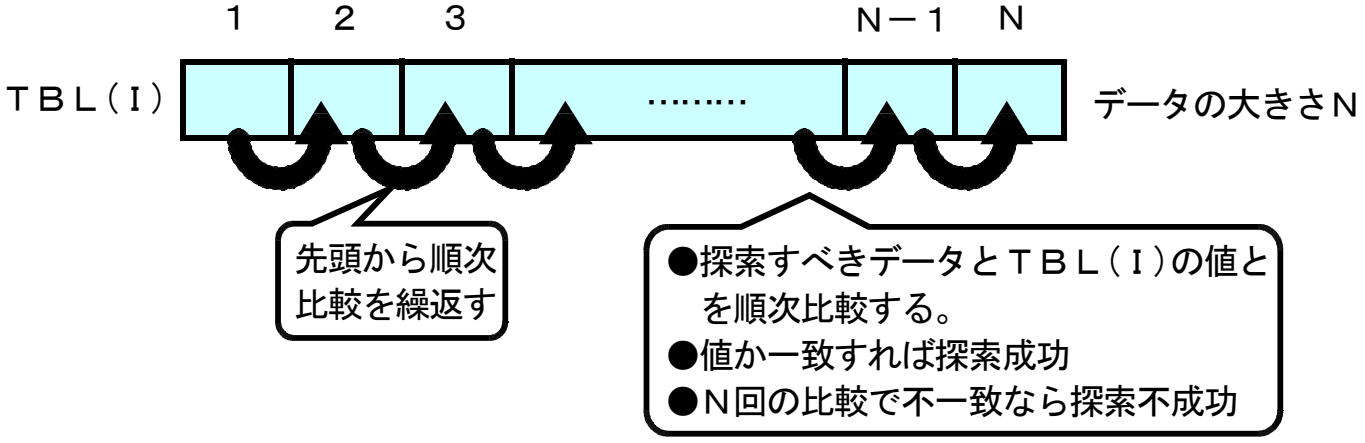
① 線形探索法

② 直接探索法

③ 二分探索法

◇線形探索法

線形探索は配列の先頭または後方から配列の要素と探索すべきデータを逐次比較して、要素の内容と探索データが等しくなる添字を求める方法である。



線形探索を行う場合、比較対象の要素が配列の定義域内に存在するかどうかは重要である。逐次比較する場合、配列の要素番号である添字のインクリメントを実行する。このインクリメントによって、コンピュータが認識できる要素を移動させることができる。

◇線形探索の手順

① 添字Ｉを１（またはＮ）で初期化する。

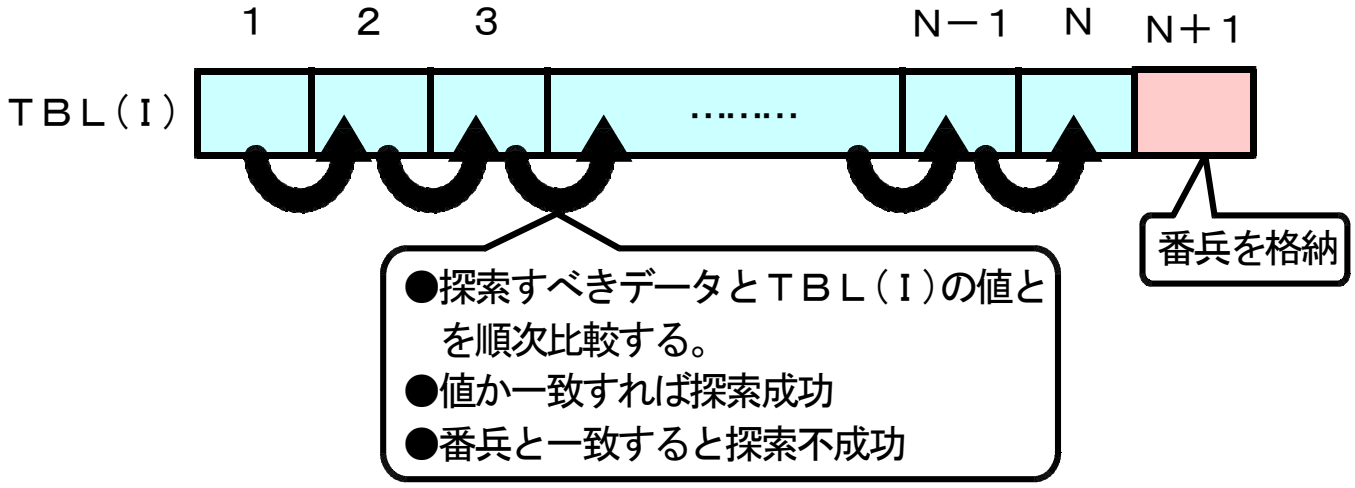
② 添字Ｉが配列の大きさＮを超える（または１より小さくなる）と、探索データが配列内に存在しないため探索を終了する。添字Ｉが配列の大きさＮ以下（または１以上）ならば③～⑤の処理を繰り返す。

③ 配列の要素ＴＢＬ（Ｉ）と探索データＫを比較する。

④ ＴＢＬ（Ｉ）＝Ｋならば探索成功。処理を終了する。

⑤ ＴＢＬ（Ｉ）≠Ｋならば探索不成功。Ｉ＋１→Ｉ（またはＩ－１→Ｉ）を実行し、②に戻る。

◇番兵を利用した線形探索



番兵を利用した探索は、大きさＮの配列のＮ＋１番目の要素に探索するデータを格納する方法である。番兵を利用すると、比較回数を減少させることができる。

◇直接探索法

探索すべき位置が分かっている場合には、その位置を示す添字の値を用いて直接探索する。

配列の添字が分かると、配列の先頭アドレス、配列の要素の大きさ、添字を使用して対象の要素のアドレスを求めることができる。

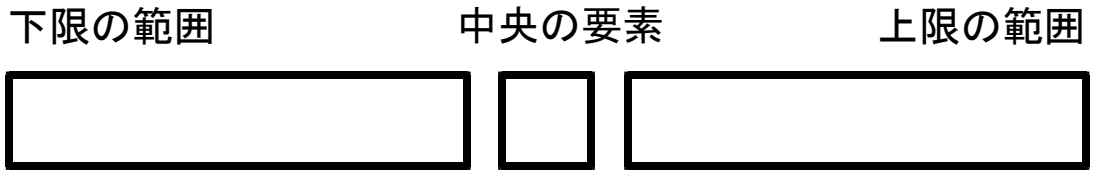
二つの配列を使用して、一方の配列で線形探索を用いて添字Ｉを求め、その添字Ｉを用いてもう一方の配列を直接探索し目的のデータを得ることができる。

◇二分探索法

データが昇順または降順に並んでいる場合には、二分探索法を用いて目的のデータを探索することができる。

昇順に並んでいる場合、二分探索は、中央の要素番号を求め、中央の要素の値と探索データを比較し、等しければ探索成功、探索データが小さい場合、中央の要素番号より小さい範囲を新しい範囲とし、探索データが大きい場合、中央の要素番号より大きい範囲を新しい範囲として探索を繰り返す方法である。

探索回数は線形探索と比較して、平均探索回数、最大探索回数共に小さくなる。探索対象のデータ数が多い場合、探索に必要な比較回数が大幅に減少し、探索時間を減少させることができる。



探索データが小さい場合の範囲　　探索データが大きい場合の範囲

◆データの更新と削除

◇データの更新

データの更新は、特定のデータを探索後、そのデータの内容を読み出し、内容を変更し、書き込む一連の操作である。

探索には、次の方法を使用する。

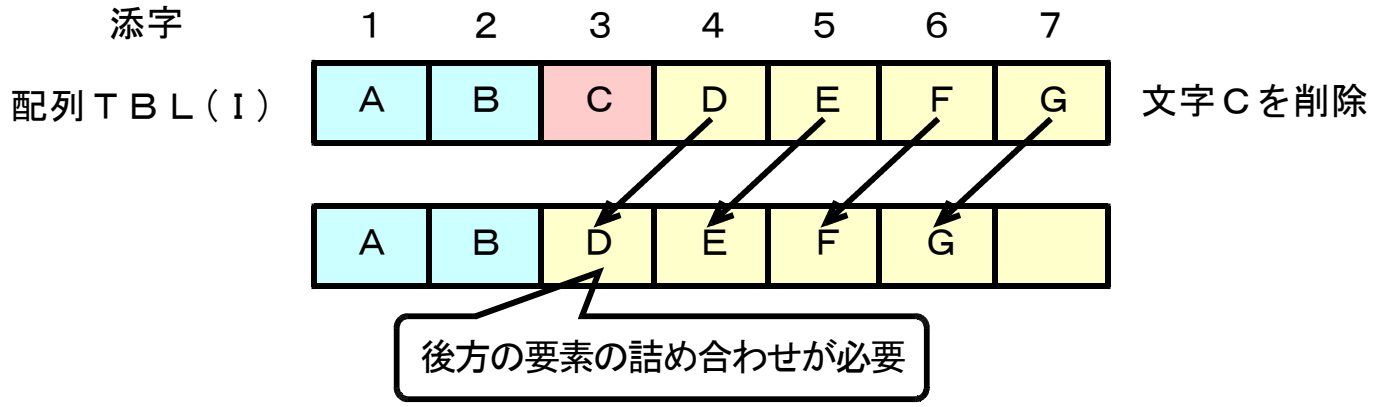
① 線形探索法
② 直接探索法
③ 二分探索法

◇データの削除

データの削除は、削除すべき配列の要素を探索法を利用して求め削除する。

削除の方法には物理的削除と論理的削除がある。

◇物理的削除の手順



物理的削除は、削除した要素の空白部分を無くするため、順次配列の要素を前に詰めていく。

大きさＮの配列ＴＢＬ(Ｉ)の添字Ｋから後方の各要素を１要素前に移動させる手順は次のようになる。

①　ＩにＫを格納する。

②　ＴＢＬ(Ｉ)→ＴＢＬ(Ｉ－１)を実行する。

③　Ｉ＋１→Ｉを実行する。

④　Ｉ＞Ｎになると終了する。Ｉ≦Ｎならば②に戻る。

Ｍ文字前に移動する操作は②の操作がＴＢＬ(Ｉ)→ＴＢＬ(Ｉ－Ｍ)となる。

◇論理的な削除

論理的な削除は、削除する要素の内容を削除されている意味の内容で書き換える。

処理対象データが多くなると、物理的な削除には、多くの処理時間が必要となるため、詰め合わせを実行しない論理的削除を用いる。

論理的削除を頻繁に利用していると、有効なメモリ空間が次第に減少し、必要なデータがメモリ内に配置できない現象が発生する。その対策として、一定間隔で、システムへの影響が最小限になるタイミングを考えて物理的な削除を実行する。

◆データの挿入

◇データ挿入とは

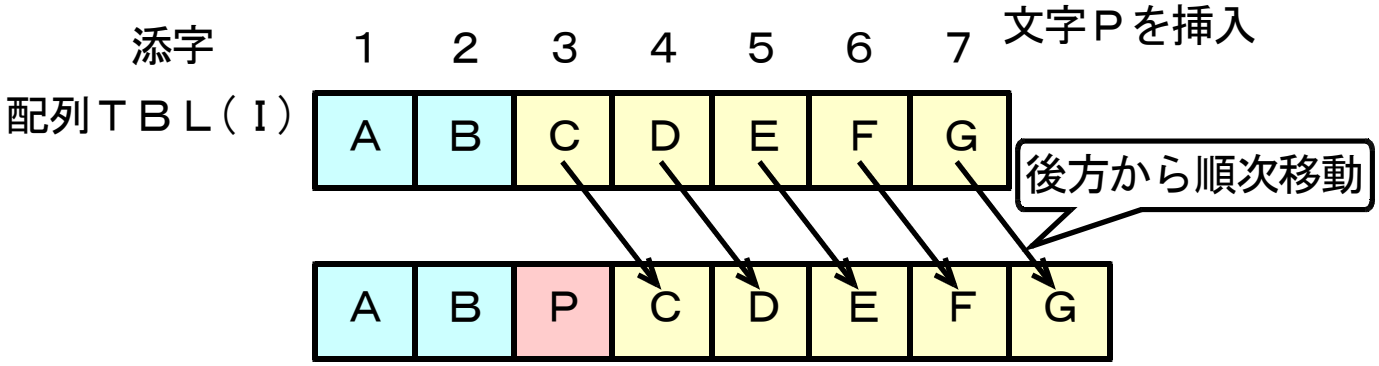
データの挿入は、挿入位置を探索法を利用して求め、挿入に必要な空白部を設ける。

挿入位置の探索には、線形探索、直接探索、二分探索を利用する。空白部を設ける場合、配列の要素の移動は配列の後方から実行する。

データの移動を後方から実行し、移動対象データを順次前方に移動させながら、データの挿入位置まで繰り返す。対象データの前方への移動は、添字のディクリメントで実行する。

データの移動を後方から実施することによって、既存のデータの消失を防ぐことができる。

図は配列ＴＢＬ(Ｉ)の添字３の要素の位置に文字Ｐを挿入する場合の例である。黄色の文字Ｃ～Ｇを順次後方から一つ後ろの要素に移動させている。



◇データ挿入手順

大きさＮの配列ＴＢＬ(Ｉ)の添字Ｋの位置に１文字を挿入する手順は次の通りになる。

①　ＩにＮを格納する。

移動の開始を後方から始める。

②　ＴＢＬ(Ｉ)→ＴＢＬ(Ｉ＋１)を実行する。

配列の要素を一つ後ろに移動させる。

③　Ｉ－１→Ｉを実行する。

移動対象の要素を一つ前に移動させる。

④　Ｉ≧Ｋならば②に戻る。

要素の移動を挿入位置まで繰り返す。

⑤　Ｉ＜Ｋになると、Ｉ＝Ｋの位置に新しい文字を格納する。

⑥　配列の大きさをＮ＋１→Ｎで修正する。

Ｍ個のデータを挿入する移動操作は②の操作がＴＢＬ(Ｉ)→ＴＢＬ(Ｉ＋Ｍ)となる。Ｍ個のデータを挿入後の配列の大きさはＮ＋Ｍ→Ｎとなる。

◆二次元配列

◇二次元配列

二次元配列の要素は、行をＩ、列をＪで表すと、ＴＢＬ(Ｉ、Ｊ)で表すことができる。

二次元配列ＴＢＬ(Ｉ、Ｊ)を利用して、行方向の合計、列方向の合計、全体の合計を求めることができる。

◇一次元配列への変換

Ｎ行Ｍ列の二次元配列ＴＢＬ(Ｉ、Ｊ)を一次元配列ＴＢＬ(Ｋ)に変換する場合、次の式を利用し、添字Ｋを求める。

Ｋ＝(Ｉ－１)×Ｍ＋Ｊ

二次元配列を一次元配列に変換することによって、一次元配列を使用して、二次元配列を操作することが可能になる。

◇三次元の配列の要素

三次元配列の要素は、ＴＢＬ(Ｉ、Ｊ、Ｋ)で表す。この場合のＩは面、Ｊは行、Ｋは列を表す。

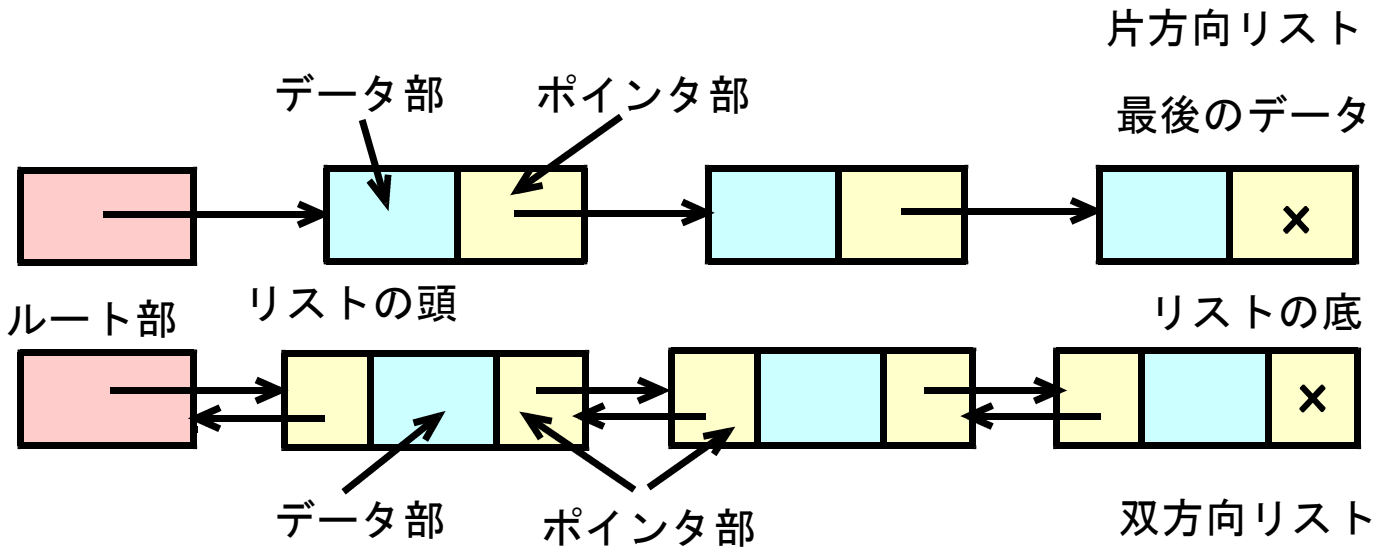
リスト

◆定義と特徴

◇リストの定義と構成

リストは、ルート部を先頭にして、各データがポインタ部で結合された一連のデータ群である。各データは物理的に連続に配置される必要がない。

データはデータ部とポインタ部で構成される。データはデータ部に格納され、ポインタ部には次のデータの所在が格納されている。各データはポインタを利用して、順次つながれた構造になっている。リストの最後のデータのポインタ部には、最後のデータであることを示す記号が格納される。



◇片方向リストと双方向リスト

片方向リストは一方向にしか辿ることができないリストであり、双方向リストは両方向に辿ることのできるリストである。

◆データの探索

◇リストの探索

リストの探索はルート部から開始し、各データのポインタ部を用いて、順次、各データのアドレスを求め、探索データと各要素の内容を比較する。

探索データと各要素の内容を比較して一致すれば探索成功、最後の要素まで比較して不一致ならば探索不成功になる。

ポインタを利用して探索するので線形探索のみ可能である。直接探索はできない。双方向リストの場合、前方後方いずれの方向にも探索可能である。

◇片方向リストの探索手順

① ルート部から最初のデータのアドレスを知る。

② 探索データとデータ部の内容を比較する。

③ (探索データ＝データ部の内容)ならば探索成功、処理を終了する。

④ (探索データ≠データ部の内容)ならば探索不成功、ポインタ部の情報から次のデータのアドレスを知る。②に戻る。

⑤ (探索データ≠最後のデータ部の内容)ならば、探索データが存在しないため、処理を終了する。

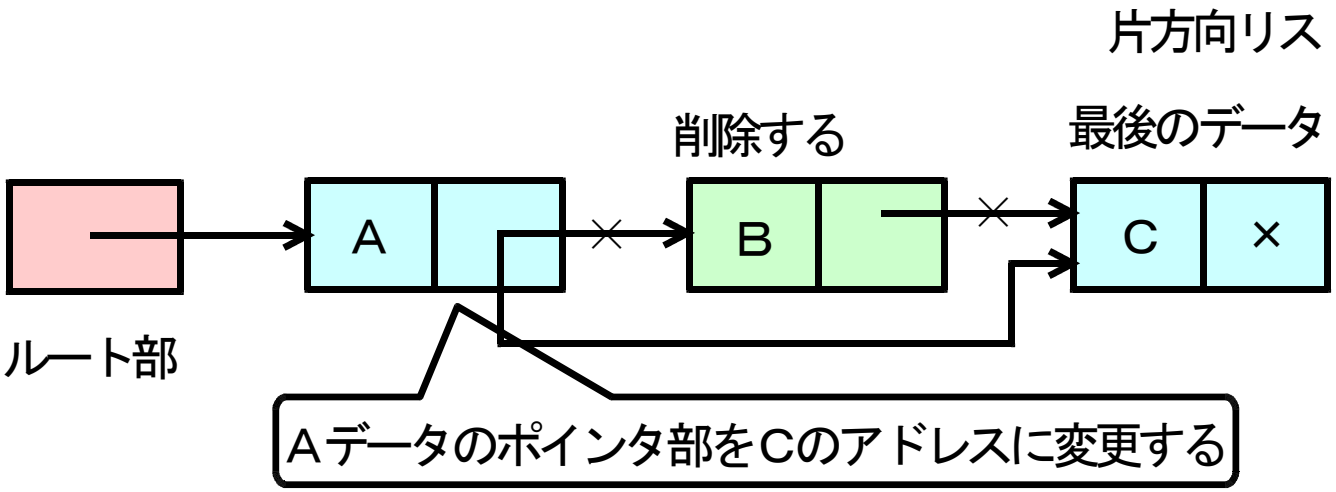
◆データの更新と削除

◇リストの更新

リストの更新は更新すべきデータを線形探索で求め、その内容を書き換えることである。

◇リストの削除

リストの削除は削除するデータの直前のデータのポインタ部を修正することによって行う。



図は、データＢを削除する例を示す。データＡのポインタ部をＢのアドレスから、Ｃのアドレスに変更すればよい。データＢのポインタ部の内容を、データＡのポインタ部に複写することによって可能となる。

リストの場合、配列のように削除による要素の詰め合わせは必要ない。削除するデータの確認のための探索、削除データの前のデータのポインタの操作に必要なアドレスの記憶が必要になる。

◇リストの削除の手順

① メモリ上にアドレスを格納する領域Ｑを確保し、ルート部の情報から最初のデータを読む。

削除データの一つ前のデータのアドレスを記憶するために必要な領域Ｑである。

② 最初のデータが削除するデータならば、最初のデータのポインタ部の情報をルート部に格納し、処理を終了する。

③ 探索したデータが削除するデータでないならば、探索したデータのアドレスをメモリ領域Ｑに格納し、探索したデータのポインタ部の情報から次の探索するデータのアドレスを知る。

④ 探索したデータが削除するデータの場合、削除するデータのポインタ部の情報を用いてメモリ領域Ｑのデータのポインタ部の内容を修正し、処理を終了する。

⑤ 最後のデータになるまで、③に戻り処理を繰り返す。

⑥ 最後のデータに達しても探索するデータでない場合、処理を終了する。

◆データの挿入

◇リストの挿入

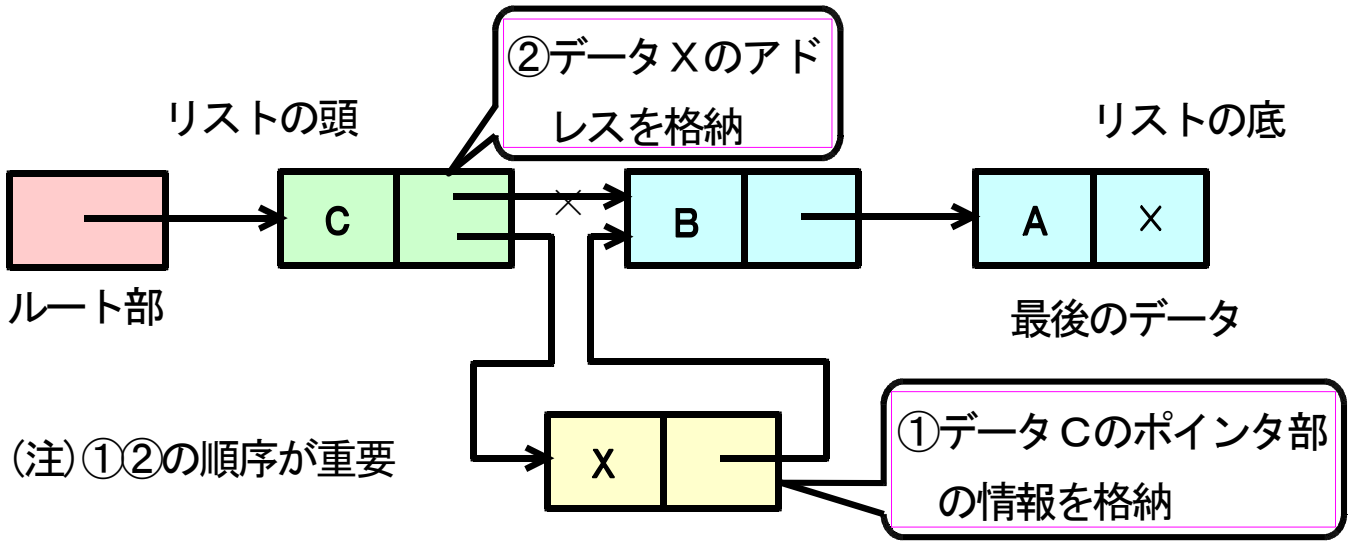
リストの挿入は、挿入するデータの直前のデータのポインタ部の書き換えと挿入するデータのポインタ部への書き込みによって行われる。

データの挿入には、次の異なる条件がある。

① 特定データの直後に挿入する。
② 特定データの直前に挿入する。
③ リストの先頭に挿入する。

◇特定データの直後に挿入

データCの直後にデータXを挿入する



挿入位置を線形探索で決定し、挿入位置のデータのポインタ部の書き換えと新しく挿入するデータのポインタ部への情報の書き込みによって行う。

挿入位置のデータのポインタ部の情報を挿入するデータのポイ

ンタ部に格納し、挿入位置のデータのポインタ部に新しく挿入するデータのアドレスを格納する。

図はデータＣの直後にデータＸを挿入する例である。データＣの所在を確認する線形探索後、データＣのポインタ部の内容をデータＸのポインタ部に格納し、データＣのポインタ部にデータＸのアドレスを格納する。

◇特定データの直後に挿入する手順

① メモリ上のデータ列の先頭位置にデータを挿入する場合、ルート部の情報を挿入するデータのポインタ部に格納し、ルート部の内容を挿入するデータのアドレスに書き換える。

② 最初のデータが挿入位置ならば、挿入位置のデータのポインタ部の情報を挿入するデータのポインタ部に格納し、挿入するデータのアドレスを挿入位置のデータのポインタ部に格納し、処理を終了する。

③ 探索したデータが挿入位置でないならば、探索したデータのポインタ部の情報から次の探索するデータのアドレスを知る。

④ 探索したデータが挿入位置のデータの場合、挿入位置のデータのポインタ部の情報を挿入するデータのポインタ部に格納し、挿入するデータのアドレスを挿入位置のデータのポインタ部に格納し、処理を終了する。

⑤ 最後のデータになるまで、③に戻り処理を繰り返す。

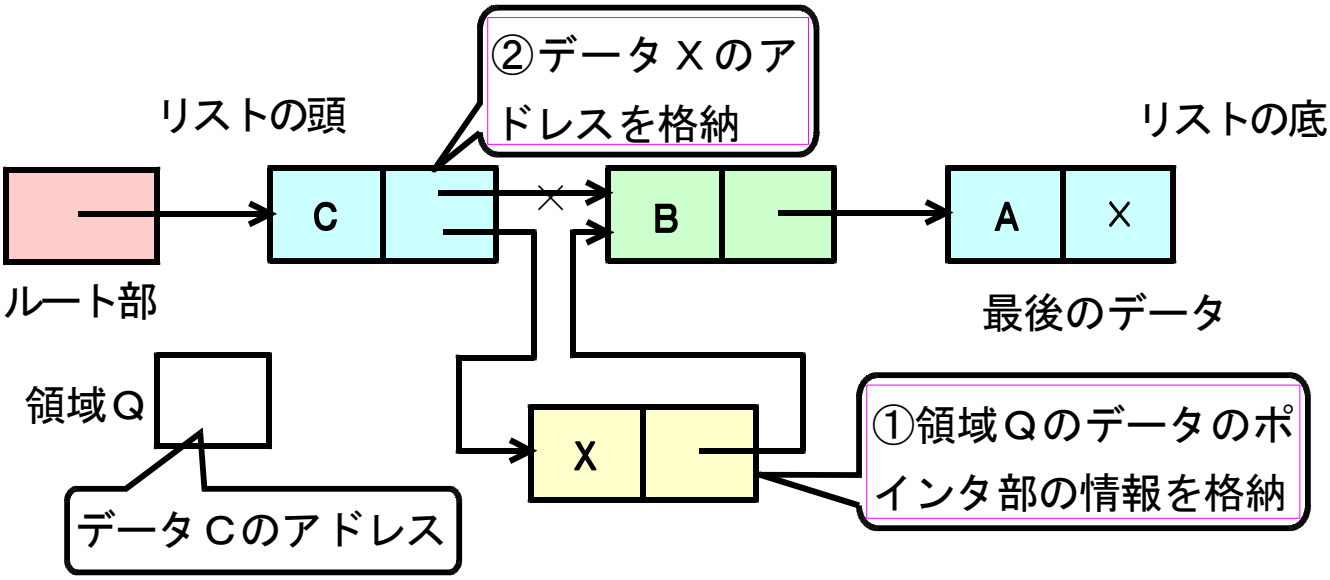
⑥ 最後のデータに達しても挿入位置がない場合、処理を終了する。

◇特定データの直前に挿入

挿入位置を線形探索で決定し、挿入位置の直前のデータのポインタ部の書き換えと新しく挿入するデータのポインタ部への情報の書き込みによって行う。

線形探索で挿入位置を探索する場合、探索データの直前のデータのアドレスを特定の記憶領域に格納する。探索直前のデータのアドレスを記憶していないと、挿入直前のデータのポインタ部を書き換えることができなくなる。

データＢの直前にデータＸを挿入する



挿入位置の直前のデータのポインタ部の情報を挿入するデータのポインタ部に格納し、挿入位置の直前のデータのポインタ部に新しく挿入するデータのアドレスを格納する。

図はデータＢの直前にデータＸを挿入する例である。線形探索を利用して、データＣ、データＢを探索し、データＢの所在を確認する。この場合、次のデータを探索する場合に、直前のデータのアドレスを記憶領域Ｑに格納する。次に、データＣのポインタ部の情報をデータＸのポインタ部に格納し、データＣのポインタ

部にデータＸのアドレスを格納する。

◇特定データの直前に挿入する手順

①　アドレスを格納する領域Ｑを確保し、ルート部から最初のデータを読む。

　領域Ｑは挿入直前のデータのアドレスを確保するために、線形探索の最初から実行する。

　線形探索によって、コンピュータが認識できるデータは探索対象のデータであり、直前のデータは認識の対象外になるため特定の記憶領域を利用して直前のデータのアドレスを認識できるようにしておく。

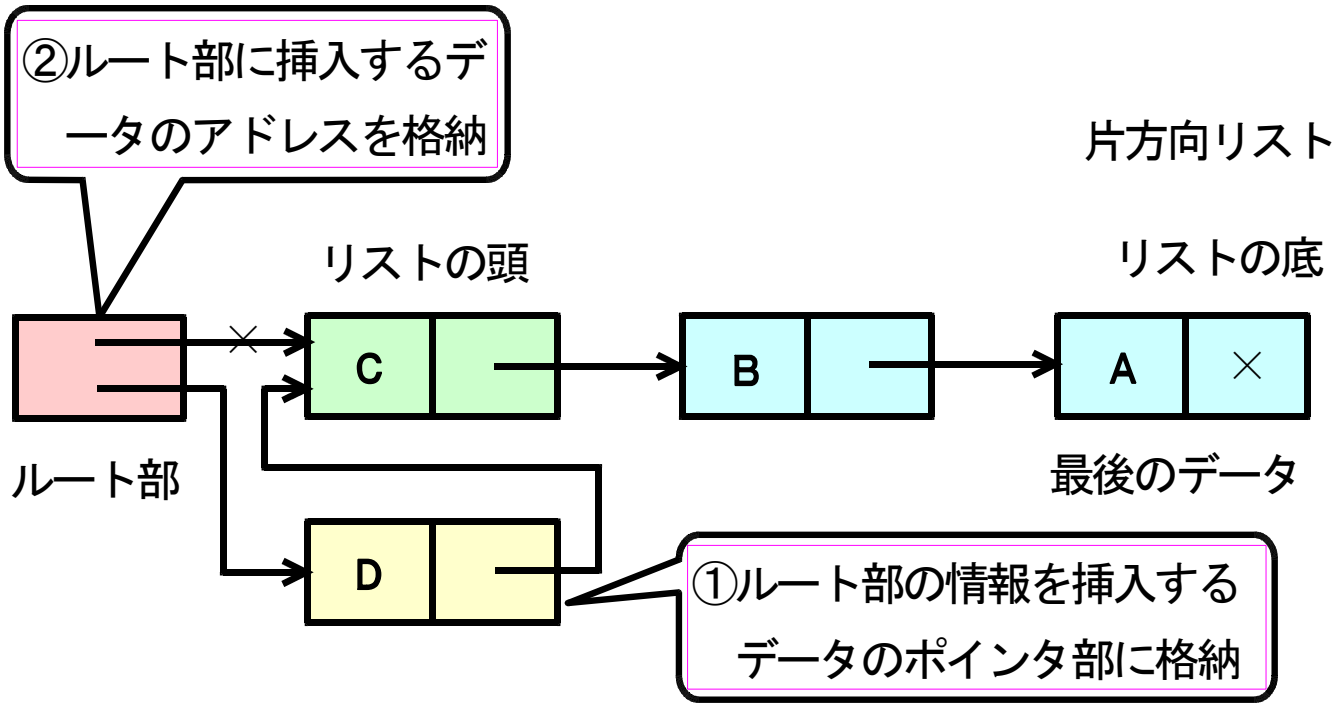
②　最初のデータが挿入位置の場合、ルート部の情報を挿入するデータのポインタ部に格納し、挿入するデータのアドレスをルート部に格納し、処理を終了する。

③　探索したデータが挿入位置でない場合、処理④に移り、挿入位置の場合、処理⑤に移る。

④　探索したデータが最後のデータでなければ、探索したデータのアドレスを領域Ｑに格納し、探索したデータのポインタ部の情報から次に探索するデータのアドレスを知り、処理③に戻り、最後のデータならば、処理を終了する。

⑤　領域Ｑのデータのポインタ部の情報を挿入するデータのポインタ部に格納し、挿入するデータのアドレスを領域Ｑのデータのポインタ部に格納し、処理を終了する。

◇リストの先頭部に挿入



図はリストの先頭位置にデータを挿入する場合の状態を示したものである。ルート部の操作によって挿入が可能となる。

ルート部の情報をデータＤのポインタ部に格納し、ルート部にデータＣのアドレスを格納する。

◇初期化操作

初期化は、ルート部にリストの終端マークを設定すればよい。ルート部に次の要素のアドレスがなければ、次のデータを探索することができない。

スタックと待ち行列

◆スタックの定義と特徴

◇スタックの定義

スタックは、データの挿入と削除が一方の端でのみ行われる。挿入や削除が行われる一方の端を頂上、もう一方の端を底という。挿入をプッシュ(ＰＵＳＨ)、削除をポップ(ＰＯＰ)という。

スタックは、配列またはリストの構造を利用して実現することができる。

◇スタックポインタ

スタックは、スタックポインタという一つの変数によってデータを管理する。スタックポインタはスタック内の一番上のデータのアドレスを示す。データが挿入されるとスタックポインタは１増加し、データが削除されると１減少する。

◇スタックの特徴

① 要素の挿入や削除がある先頭の領域でのみ行われる。

② 後入れ先出しの特徴をもった構造である。

③ 要素の挿入をＰＵＳＨ、取り出しをＰＯＰという。

④ プログラムからサブルーチンや関数などを呼び出すときにスタックを使用し、割り込み処理やサブルーチンの戻り番地の管理に利用される。

◇スタックに関係する操作と状態

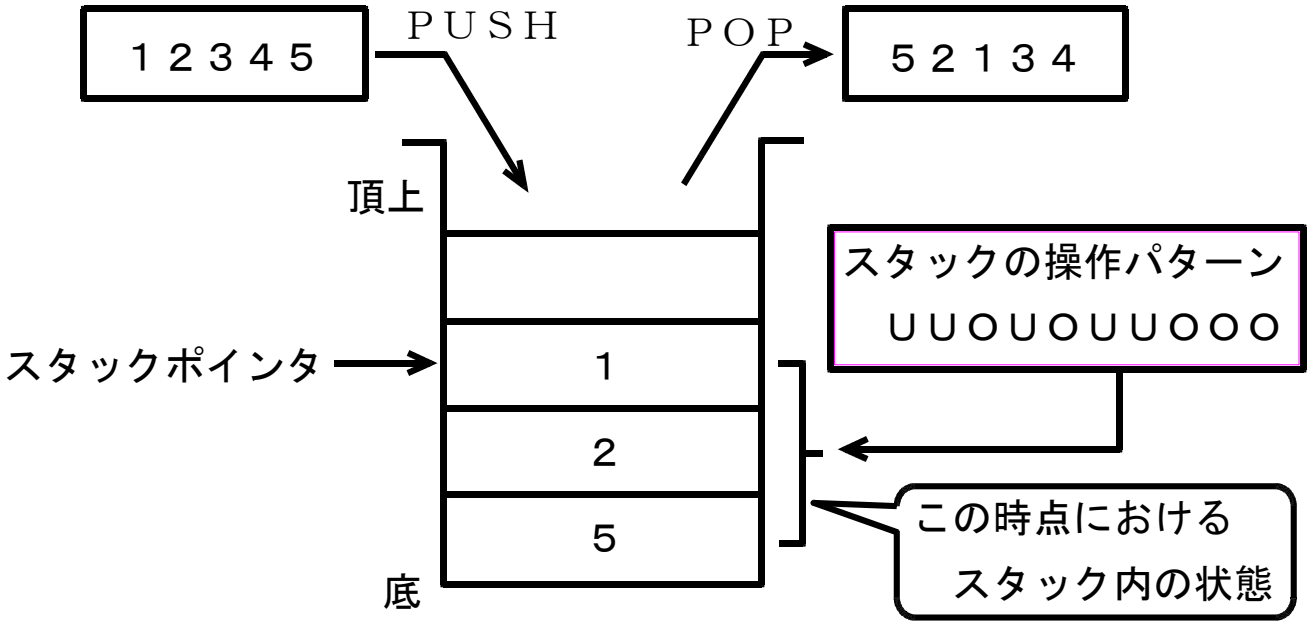
① プッシュ前のデータの並び

② ポップ後のデータの並び

③ プッシュ・ポップの操作パターン

④ プッシュ・ポップが行われた直後のスタック内のデータの状態

⑤ スタックポインタの値

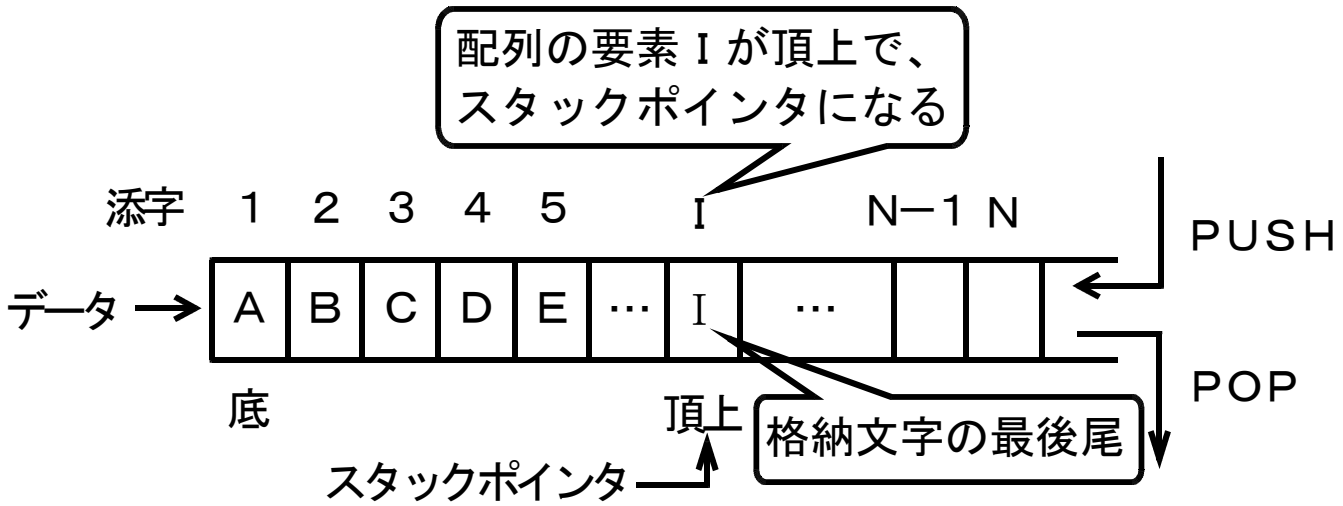


◇配列を利用したスタック

大きさＮの配列ＴＢＬ(Ｉ)を用いてスタックを実現するとき、ＴＢＬ(１)がスタックの底を表し、ＴＢＬ(Ｎ)がスタックの頂上になる。

スタックポインタはスタックの頂上のアドレスを示し、配列の添字が用いられる。配列の最後尾の添字がスタックポインタになる。

スタックにデータを格納するとき、スタックのオーバフローをチェックし、スタックからデータを取り出すとき、スタックが空でないことをチェックする。



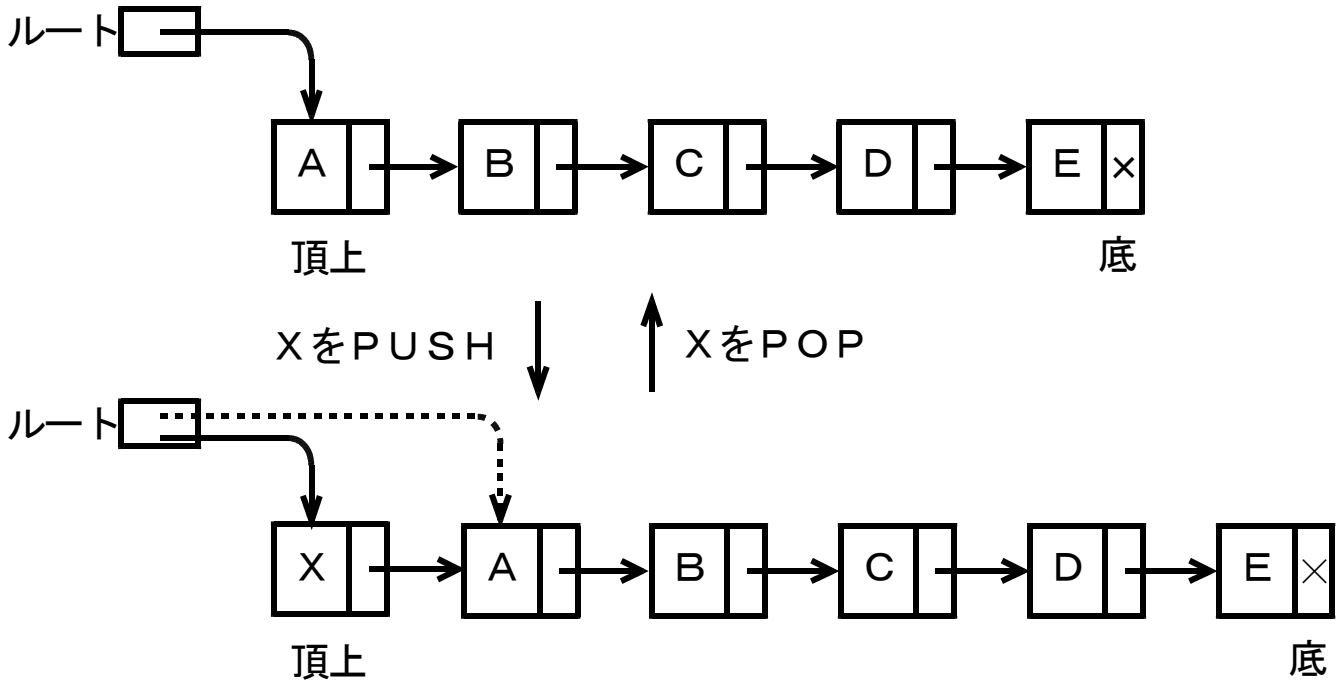
◇リストを利用したスタック

リストの最初のデータを頂上とし、リストの最後のデータを底とする。頂上がスタックポインタの位置になる。

スタックへのデータの格納と取り出しはリストの先頭から行う。データの格納はリストの先頭への挿入であり、データの取り出しはリストの先頭のデータを削除することである。

データを取り出すとき、リストが空であるかどうかはルートの内容が終端記号であるかどうかでわかる。

図はＥＤＣＢＡの５個のデータを順次格納されたスタックに、データＸをプッシュした場合を示している。ルート部の情報をデータＡのポインタ部に格納し、ルート部にデータＸのアドレスを格納すればよい。



◇スタックを利用した四則演算の例

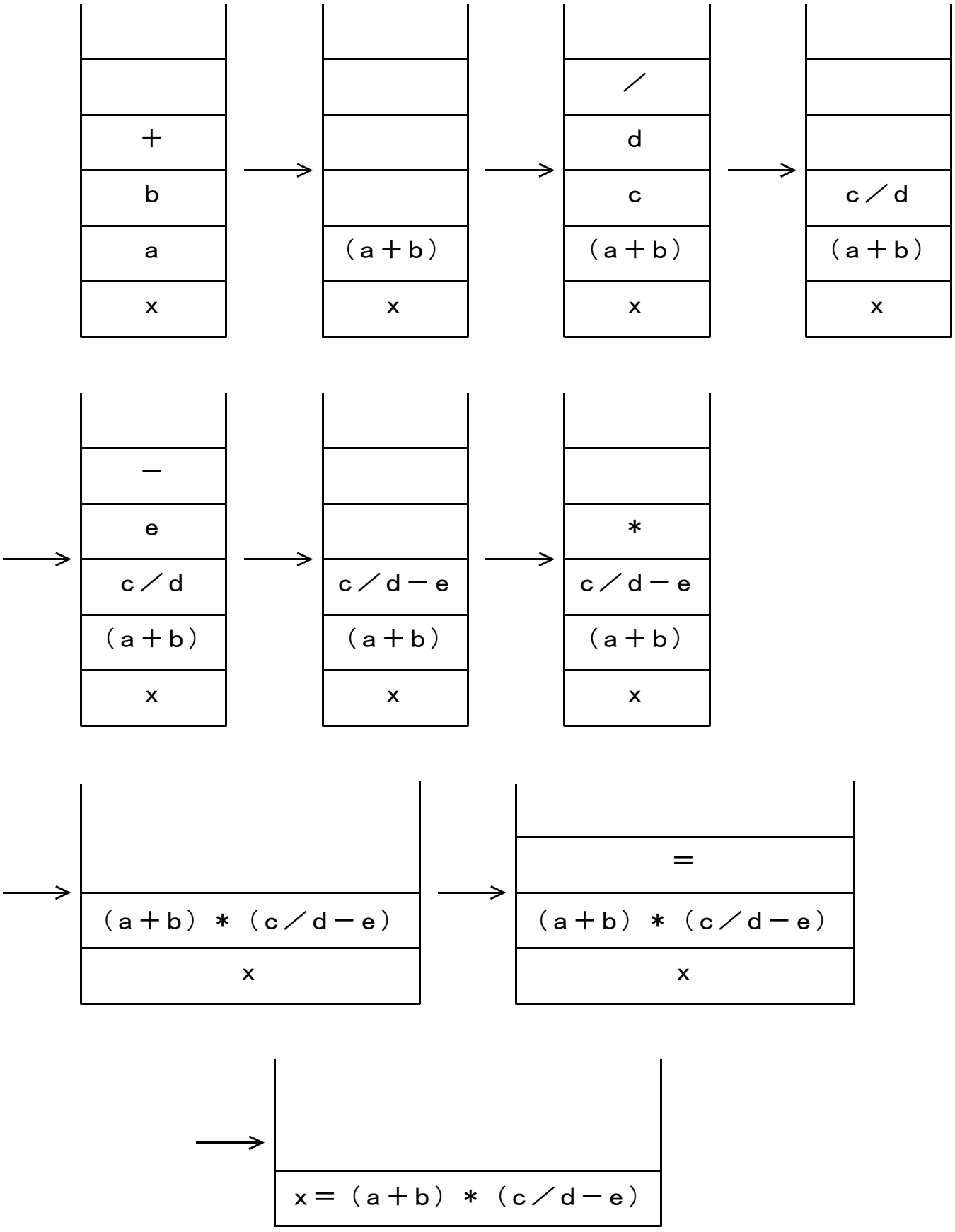
①　四則演算式

ｘ＝（ａ＋ｂ）＊（ｃ／ｄ－ｅ）

②　逆ポーランド記法

ｘ ａ ｂ＋ｃ ｄ／ｅ－＊＝

③　スタックを利用した演算の推移


＋
b
a
x
（a＋b）
x
／
d
c
（a＋b）
x
c／d
（a＋b）
x
－
e
c／d
（a＋b）
x
c／d－e
（a＋b）
x
＊
c／d－e
（a＋b）
x
（a＋b）＊（c／d－e）
x
＝
（a＋b）＊（c／d－e）
x
x＝（a＋b）＊（c／d－e）

◇スタックを利用した四則演算のアルゴリズム

四則演算式から二分木を作成し、作成した二分木を使用して、後項順の巡回を用いると、逆ポーランド記法を作成することができる。この逆ポーランド記法をスタックを用いて次の処理を実行すると四則演算の結果を求めることができる。

逆ポーランド記法の作成要領は二分木の巡回法を参照のこと。

スタックを利用した逆ポーランド記法のアルゴリズムは次のようになる。

① スタックポインタを初期化する。

② 先頭から＝が現れるまで、③、④の処理を繰り返す。

③ 変数または数値ならばスタックにＰＵＳＨする。

④ 演算子が現れると、次の処理を行う。

ⓐ スタックから２つの要素を連続してＰＯＰする。

ⓑ 取り出した２つの要素を(後で取り出した要素)演算子(先に取り出した要素)の順に処理する。

ⓒ 演算結果をスタックにＰＵＳＨする。

⑤ ＝が現れると、スタックの内容をＰＯＰし、処理を終了する。

◆待ち行列の定義と特徴

◇待ち行列の定義

待ち行列は、一方の端で挿入が行われ、他方の端で削除が行われるものである。新しいデータは最後のデータの後ろに挿入され、最初の古いデータが削除の対象になる。

待ち行列は２つの変数によって管理される。frontは先頭のデータのアドレスであり、rearは末尾の次のアドレスである。

利用例として、多重プログラミングの実行待ちプログラムの待ち行列がある。

◇待ち行列の特徴

① 要素の挿入が一方の端で行われ、取り出しが他方の端で行われるデータ構造である。

② 待ち行列は先入先出、ＦＩＦＯの特徴をもつ。

③ front､rearの二つの変数で管理される。

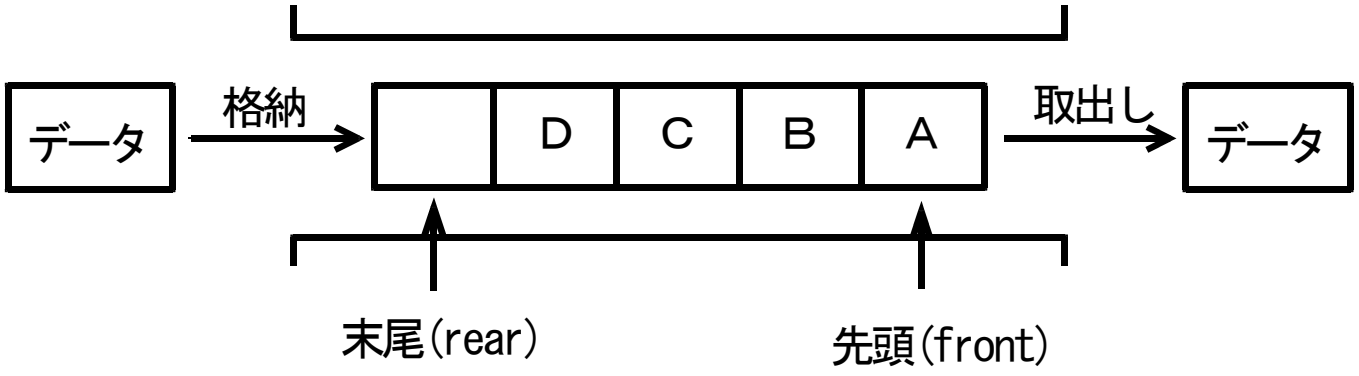
④ サービスを受けるために待っている行列。

◇配列を利用した待ち行列

配列を利用して、待ち行列の先頭を示す変数frontと末尾を示す変数rearを定義し、この２つの変数に挟まれる部分を待ち行列と見なす。変数rearは配列の最後の要素の次の要素の添字を示す。

変数rearと変数frontの差がデータの個数を示す。変数rearと変数frontが一致したとき配列が空になる。

◇待ち行列の格納・取出しの操作



行列を利用した待ち行列からのデータの取り出し、格納の操作は次のようになる。

① 配列の添字の小さい要素から順にデータを格納する。

② データを１つ格納すると変数rearを１増加させる。

③ データを１つ取出すと変数frontを１増加させる。

図において、待ち行列から先頭のデータＡを取り出すと、先頭のポインタfrontはデータＢに移動する。

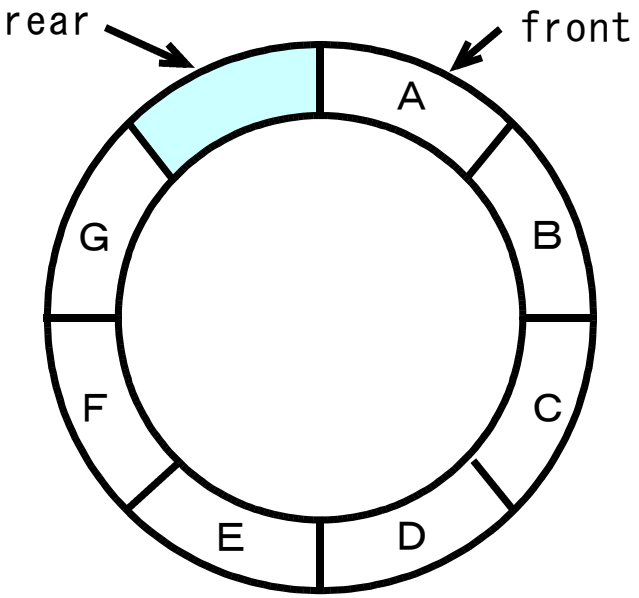
待ち行列にデータＥが格納されるとポインタrearはデータＥの次の要素に移動する。

◇配列を利用した待ち行列の問題点

配列を利用した待ち行列を考えると、無限大の長さの待ち行列が必要になる。メモリ空間内に無限の配列を考えることは不可能であり、リングバッファを利用することになる。

◇リングバッファを利用した待ち行列

リングバッファは配列を環状にし、配列の最後の次は配列の先頭を指すように操作する。リングバッファを使用することによって、有限のメモリ領域で待ち行列を実現することができる。



リングバッファが満杯になったとき、先頭＝末尾となり、待ち行列の空の状態rear＝frontに等しくなる。満杯と空の状態が同じ条件になり、論理的に矛盾する。

待ち行列が空である変数を設けたり、最大個数－１の個数しか格納できない規則を設けたりして、この矛盾を解決する。

データ構造問題１

問１問題

計算機の中で、時刻などを記憶させる配列と２個のポインタを用いて表すＦＩＦＯ型のデータ構造はどれか。

ア　リスト
イ　ストリング
ウ　スタック
エ　キュー

◇問１解説

キューに関する問題である。

キュー(待ち行列)は配列またはリストを利用して表すことができる。キューを管理するためにfront､rearの二つのポインタが利用される。

アのリストはデータ部と単一または複数のポインタで管理されるが、ポインタはデータの前後関係を表し、各要素は連続に配置される必要がない。
イのストリングは文字を１列に並べたものである。
ウのスタックは最後に格納したデータを最初に取り出すＬＩＦＯの特徴を持つ。
エのキューは先入れ先出し（ＦＩＦＯ）の特徴のあるデータ構造で、先頭を示すfrontと末尾を示すrearの二つのポインタで管理する。求める答えはエとなる。

◇問１解答　エ

問２問題

待ち行列に対する操作を次のとおり定義する。
ＥＮＱｎ：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，ＥＮＱＩ，ＥＮＱ２，ＥＮＱ３，ＤＥＱ，ＥＮＱ４，ＥＮＱ５，ＤＥＱ，ＥＮＱ６，ＤＥＱ，ＤＥＱの操作を行った。次のＤＥＱの操作で取り出される値はどれか。

ア　１
イ　２
ウ　５
エ　６

◇問２解説

待ち行列の操作に関する問題である。

問題に与えられた手順を逐次実行する。次のようになる。

①　３回の挿入の結果　　３２１先頭
②　４回目で１つ取り出す　　３２先頭
③　２回挿入した結果　　５４３２先頭
④　７回目で１つ取り出す　　５４３先頭
⑤　１回の挿入の結果　　６５４３先頭
⑥　９回目、10回目の取出し　　６５先頭
⑦　次の取出しが求める答えとなる。取り出される数字は５である。求める答えはウとなる。

◇問２解答　ウ

問３問題

キューに四つの値８、３、６、１がこの順に格納されている。このキューから最初に取り出される値はどれか。

ア　１
イ　３
ウ　６
エ　８

◇問３解説

キューに関する問題である。

キューの特徴はＦＩＦＯであるから、格納した順序に取り出される。

格納順は、８→３→６→１であるから、最初に取り出される値は８であり、求める答えはエとなる。

◇問３解答　エ

問４問題

スタックに関する記述として，適切なものはどれか。

ア　最後に格納したデータを最初に取り出すことができる。
イ　最初に格納したデータを最初に取り出すことができる。
ウ　探索キーからアドレスに変換することによって，データを取り出すことができる。
エ　優先順位の高いデータを先に取り出すことができる。

◇問４解説

スタックに関する問題である。

スタックはデータの挿入と削除が一方の端でのみ行われる。挿入をプッシュ(ＰＵＳＨ)、削除をポップ(ＰＯＰ）という。後入先出、ＬＩＦＯの特徴をもつ。

プログラムからサブルーチンや関数などを呼び出すときにスタックを使用する。

アがスタック、イが待ち行列、ウはハッシュ、エは優先順序別待ち行列であり、求める答えはアとなる。

◇問４解答　ア

問５問題

空のスタックに対して次の操作を行った場合，スタックに残っているデータはどれか。

ここで，“push x”はスタックへデータｘを格納し，“pop”はスタックからデータを取り出す操作を表す。

push１→push２→pop→push３→push４
→pop→push５→pop

ア　１と３
イ　２と４
ウ　２と５
エ　４と５

◇問５解説

スタックの操作に関する問題である。

後入れ先出しの特徴を利用して、push、popの操作を実行する。

① push１、push２でスタック内は１、２
② popでスタック内に１
③ push３、push４でスタック内は１、３、４
④ popでスタック内は１、３
⑤ push５でスタック内は１、３、５
⑥ popでスタック内は１、３となる。求める答えはアとなる。

◇問５解答　ア

問６問題

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

◇問６解説

スタックに関する問題である。

ア～エの解答に一致する操作が可能かどうかを検討する。

アの場合、ＵＯＵＵＵＯの操作で、ＡＤは可能であるが、次にＢをＰＯＰすることができない。
イの場合、ＵＵＯＵＵＯの操作で、ＢＤは可能であるが、次にＡをＰＯＰすることができない。
ウの場合、ＵＵＵＯＯＵＯＯの操作で、ＣＢＤＡが可能である。求める答えはウとなる。
エの場合、ＵＵＵＵＯＯの操作で、ＤＣは可能であるが、次にＡをＰＯＰすることができない。

◇問６解答　ウ

問７問題

次の二つのスタック操作を定義する。

ＰＵＳＨ：スタックにデータ（整数ｎ）をプッシュする。
ＰＯＰ　：スタックからデータをポップする。

空のスタックに対して、次の順序で操作を行った結果はどれか。

ＰＵＳＨ１→ＰＵＳＨ５→ＰＯＰ→ＰＵＳＨ７→ＰＵＳＨ６→
ＰＵＳＨ４→ＰＯＰ→ＰＯＰ→ＰＵＳＨ３

ア

3
4
5

イ

3
4
6

ウ

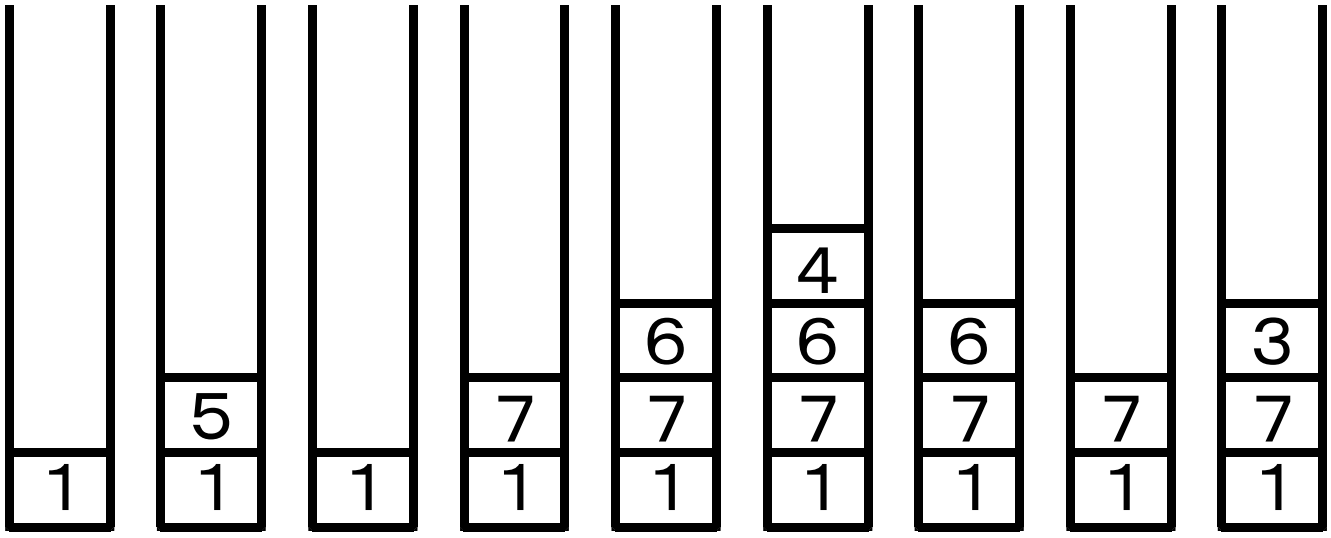
3
7
1

エ

6
4
3

◇問７解説

スタックの操作に関する問題である。



示された操作パターンであるＰＵＳＨ、ＰＯＰの操作を逐次実行する。

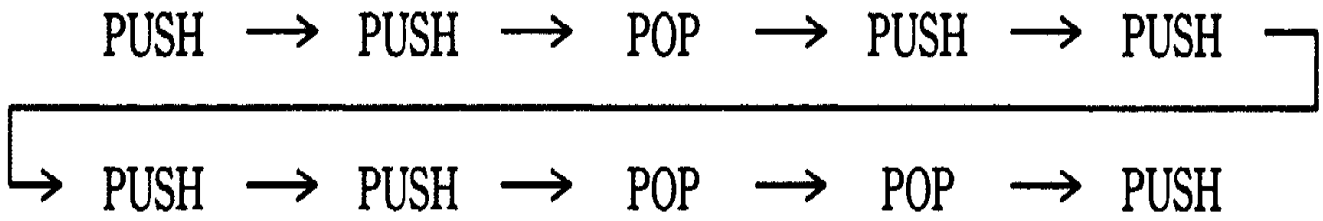
ＰＵＳＨ、ＰＯＰの操作を実行すると図のようになる。

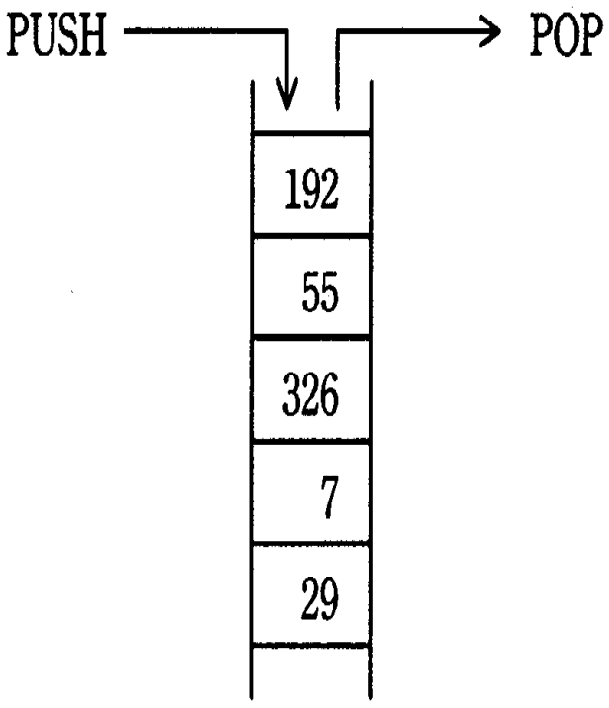
スタックの最後の結果は、スタックの頂上から順次３、７、１となる。求める答えはウとなる。

◇問７解答　ウ

問８問題

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

◇問８解説

スタックに関する問題である。

最後にPUSHであるから、最後に入れたものから何番前のものが最初のものかを調べればよい。

① PU→PU→POの３回の操作で、スタックの中に操作後のものが１個残っている。

② PU→PU→PU→PU→PO→POの６回の操作で、スタックの中に残るのは２個となり、①のものと合わせると３個となる。

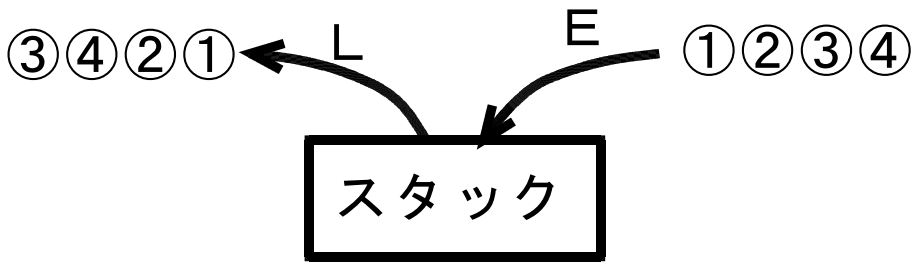
③ 最後の操作はPUであるから、操作を開始してから後に４個がスタックに残ったことになる。

４番目のものが最初にPUSHしたものであるから７となる。求める答えはアとなる。

◇問８解答　ア

問９問題

図のように４個の要素①、②、③、④が順番に入口に並んでいる。入口からスタックに要素を入れる操作をＥ、スタックから出口に要素を出す操作をＬとする。出口に出てくる要素の順序が③、④、②、①になる操作列はどれか。



ア　ＥＥＥＥＬＬＬＬ
イ　ＥＥＥＬＥＬＬＬ
ウ　ＥＥＥＬＬＥＬＬ
エ　ＥＥＥＬＬＬＥＬ

◇問９解説

スタックの操作に関する問題である。

アの場合、ＥＥＥＥＬＬＬＬであり、１２３４の順にスタックに入り、４３２１の順に取り出される。

イの場合、ＥＥＥＬＥＬＬＬであり、最初は１２３の順にスタックに入り、３が取り出されて、４が入り、次に４２１の順に取り出される。取り出した順序は３４２１となる。求める答えはイとなる。

ウの場合、ＥＥＥＬＬＥＬＬであり、最初は１２３の順にスタックに入り、３２が取り出されて、４が入り、次に４１の順に取

り出される。取り出した順序は３２４１となる。

エの場合、ＥＥＥＬＬＬＥＬであり、最初は１２３の順にスタックに入り、３２１が取り出されて、４が入り、４が取り出される。取り出した順序は３２１４となる。

◇問９解答　イ

問10問題

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

◇問10解説

スタックに関する問題である。

function f()は、ポインタｐをインクリメントし、要素番号ｐの配列Ａの要素にｘを格納する。

function g()は、要素番号ｐの配列Ａの要素の内容をｘに代入

して、ポインタｐをディクリメントし、ｘを戻す。

関数ｆはPUSHの操作であり、関数ｇはPOPの操作であるから、スタックのデータ構造となる。求める答はイとなる。

◇問10解答　イ

問11問題

関数や手続を呼び出す際に，戻り番地や処理途中のデータを一時的に保存するのに適したデータ構造はどれか。

ア　２分探索木
イ　キュー
ウ　スタック
エ　双方向連結リスト

◇問11解説

データ構造に関する問題である。

アの２分探索木は節がある規則によって並べられ、節が探索可能な二分木である。
イのキューは最初に格納されたデータが最初に読み出されるリストである。
ウのスタックは最後に格納されたデータが最初に取り出されるリストであり、関数や手続きを呼び出す際に使用する。求める答えはウとなる。
エの双方向連結リストは先行データと後続データの位置情報をもった連結リストである。

◇問11解答　ウ

問12問題

整数や実数などを格納するスタックがあり、次のルールで運用される。

（１） 入力値が整数のとき、そのままスタックにプッシュする。

（２） 入力値が実数のとき、小数部分を切り捨ててスタックにプッシュする。

（３） 入力値が算術演算子(＋、－、＊、／)のとき、スタックからポップを２回実行し、

式(後からポップされた値)
算術演算子（先にポップされた値）

を実行後、結果をスタックにプッシュする。

例えば演算子が“－”のとき、

（後からポップされた値）－（先にポップされた値）

の演算結果(差)をスタックにプッシュする。

（４） 入力値がＰのとき、スタック上に最新値を表示する。

（５） 入力値がＱのとき、スタックをクリアする。

このスタックに次の５件のデータが、Ｑを先頭にこの順序で入力されたとき、スタック上の最新値はどれか。

Ｑ、 －２.５、 ３.５、 －、 Ｐ

ア　－６
イ　－５
ウ　１
エ　５

◇問12解説

スタックを使用した四則演算に関する問題である。

次の順序で実行される。

①　スタックをクリアする
②　－２をスタックにプッシュする。
③　３をスタックにプッシュする。
④　３、－２をポップし、－２－３＝－５を計算する。
⑤　－５を表示する。

スタック上の最新値は－５となり、求める答えはイとなる。

◇問12解答　イ

問13問題

スタックとキューの二つのデータ構造がある。次の手続き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ａをスタックに挿入することをpush(ａ)、スタックからデータを取り出すことをpop() 、データａをキューに挿入することをenq(ａ) 、キューからデータを取り出すことをdeq() 、とそれぞれ表す。

push(ａ)、push(ｂ)、enq(pop())、enq(ｃ)、push(ｄ)、
push(deq())、 ｘ＝pop()

ア　ａ
イ　ｂ
ウ　ｃ
エ　ｄ

◇問13解説

スタックとキューに関する問題である。

問題の操作を順次実行すると次のようになる。

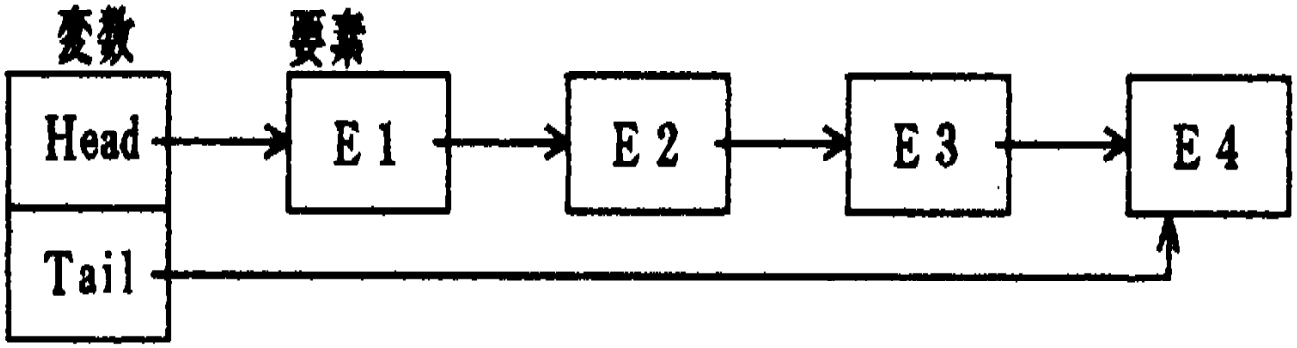
スタック	ａ	→	ｂ ａ	→	ａ	→	ｄ ａ	→	ｂ ｄ ａ	→	ｄ ａ
キュー				→	ｂ	→	ｃ ｂ	→	ｃ		→ ｃ
ｘ											＝ｂ

ｘに格納されるのはｂとなり、求める答えはイとなる。

◇問13解答　イ

問14問題

右のような構造をもった線形リストについて、正しい記述はどれか。



ア　要素の削除に要する処理量は、先頭と最後尾とでほぼ同じである。
イ　要素の追加と取出し(読出し後削除)を最後尾で行うスタックとしての利用に適している。
ウ　要素の追加に要する処理量は、先頭と最後尾とでほぼ同じである。
エ　要素の追加は先頭に、取出し(読出し後削除)は最後尾からとするＦＩＦＯ(First-in First-out)のキューとしての利用に適している。

◇問14解説

リストに関する問題である。

図の線形リストの特徴は次の通りである。

①　ルート部には先頭のアドレスと最後尾のアドレスがある。
②　先頭から２番目のアドレスは先頭の次の位置で読み出せる。
③　最後尾の前の位置は先頭のアドレスから順次求める必要がある。

アの場合、Ｅ１の削除はHeadの内容をＥ２のアドレスに修正するのみであるが、Ｅ４の削除はＥ３のアドレスを先頭から順次求めて、Tailの修正をする必要があるため両者の処理量は同じではない。

イの場合、Ｅ４の追加はTailを操作することで可能であるが、Ｅ４の削除は、Ｅ４を削除後、Ｅ３に設定するために先頭のＥ１から順次探索する必要があり、スタックのような１変数の処理ではできない。

ウの要素の追加は、先頭への追加は新しい要素にＥ１のアドレスを記入し、Headに新しい要素のアドレスを格納すればよい。最後尾への追加は追加する要素にＥ４のポインタの内容を複写し、Ｅ４のポインタにTailのアドレスを格納し、Tailに新しい要素のアドレスを格納する。従って、先頭から線形探索する必要がなく、両者の処理量はほぼ同じになる。求める答えはウとなる。

エの待ち行列では、リストの構造の先頭から最後尾に繋がっており先頭から取り出されて最後尾に追加される構造になる。図の場合は、先頭に追加されて最後尾から取り出されるため最後尾の次の要素の設定が待ち行列のような仕組みにならない。従って、待ち行列のfront、rearの様な条件にならない。

◇問14解答　ウ

問15問題

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，→は代入を表す。

列

A	1	2
1	“東京”	2
行 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 → A(1, 2)	A(A(1, 2) , 2) → A(5, 2)
イ	5 → A(1, 2)	A(A(2, 2) , 2) → A(5, 2)
ウ	A(A(1, 2) , 2) → A(5, 2)	5 → A(1, 2)
エ	A(A(2, 2) , 2) → A(5, 2)	5 → A(1, 2)

◇問15解説

リストに関する問題である。

変更するポインタは、Ａ（１，２）とＡ（５，２）の２カ所である。

変更する順序が問題になる。まず、Ａ（２，２）の内容をＡ（５，２）に複写し、その後、Ａ（１，２）に５を格納する操作となる。従って、Ａ（Ａ（１，２），２）→Ａ（５，２）、５→Ａ（１，２）となる。求める答えはウとなる。

順序が逆になると、Ａ（１，２）を５に変更し、Ａ（５，２）の内容をＡ（５，２）に格納することになりＡ（５，２）の内容が正しくならない。

◇問15解答　ウ

問16問題

先頭へのポインタ

１０

アドレス	データ	ポインタ
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	

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。

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。

◇問16解説

リスト構造のポインタ操作に関する問題である。

図の内容は、東京→新横浜→熱海→浜松→名古屋の順を示している。

熱海と浜松の間に静岡を挿入すると、東京→新横浜→熱海→静岡→浜松→名古屋となり、熱海のポインタを静岡のアドレス１５０に変更し、静岡のポインタを浜松のアドレス７０に変更すればよい。求める答えはイとなる。

◇問16解答　イ

問17問題

次のような双方向のポインタをもつリスト構造のデータがある。社員Ｇを社員Ａと社員Ｋの間に追加する場合，追加後の表のポインタａ～ｆのうち，追加前と比べて値が変わるのは何か所か。

追加前

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

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　１
イ　２
ウ　３
エ　４

◇問17解説　イ

リストの挿入に関する問題である。

社員Ａと社員Ｋの間に社員Ｇを挿入した場合のポインタは表のようになる。

ａ～ｆの間で変化したのはａの３００→４００とｆの１００→

４００の２カ所である。
　求める答えはイとなる。

アドレス	社員名	次ポインタ	前ポインタ
１００	社員Ａ	４００	０
２００	社員Ｔ	０	３００
３００	社員Ｋ	２００	４００
４００	社員Ｇ	３００	１００

◇問17解答　イ

問18問題

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。

◇問18解説

連結リストの特徴に関する問題である。

アのリストの更新はポインタを順番にたどり目的のデータの更新を行うため配列よりも処理時間は長くなる。
イの削除の場合、削除要素の後方の要素を前に移動させるのは配列の特徴である。
ウのランダムに参照できるのは配列の特徴である。
エの挿入の場合、数個のポインタを書き換えるだけという特徴はリストの特徴である。求める答えはエとなる。

◇問18解答　エ

問19問題

次の規則に従って配列の要素Ａ[０]，Ａ[１]，…，Ａ[９]に正の整数ｋを格納する。

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，ｘmodｙはｘをｙで割った剰余を返す。また，配列の要素はすべて０に初期化されている。

〔規則〕

（１）　Ａ[ｋmod１０]＝０ならば，
　　ｋ→Ａ[ｋmod１０]とする。

（２）（１）で格納できないとき，
　Ａ[(ｋ＋１)mod１０]＝０ならば，
　　ｋ→Ａ[(ｋ＋１) mod１０] とする。

（３）（２）で格納できないとき，
　Ａ[(ｋ＋４)mod１０]＝０ならば，
　　ｋ→Ａ[(ｋ＋４)mod１０]とする。

ア　Ａ[３]
イ　Ａ[５]
ウ　Ａ[６]
エ　Ａ[９]

◇問19解説

配列を利用したハッシュ法に関する問題である。

１６、４３、７３、２４、８５の格納位置を示すと次の表のようになる。

A（0）	A（1）	A（2）	A（3）	A（4）
			４３	７３

A（5）	A（6）	A（7）	A（8）	A（9）
２４	１６			８５

８５が格納されるのはA（9）となる。求める答えはエとなる。

◇問19解答　エ

問20問題

三次元の配列を表すとき“Ｘ（Ｉ、Ｊ、Ｋ）”と表記するが、このとき、Ｊが表しているものはどれか。

ア　行
イ　面
ウ　要素
エ　列

◇問20解説

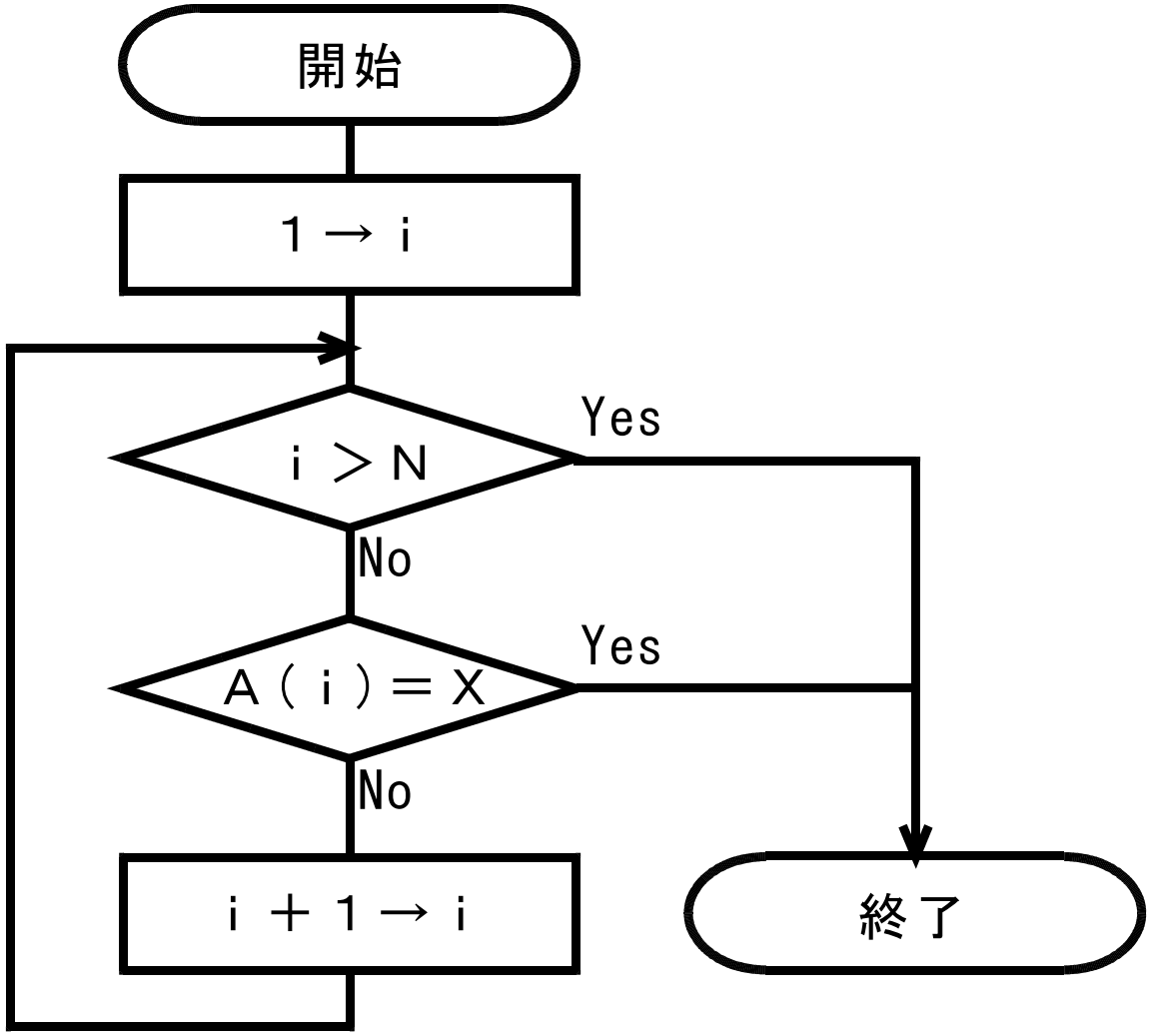
３次元配列の問題である。

３次元の配列は、面、行、列で表現される。

Ｉは面、Ｊは行、Ｋは列を表す。従って、Ｊは行を表す。求める答えはアとなる。

◇問20解答　ア

問21問題



Ｎ個の要素からなる配列Ａの各要素に整数が格納されている（Ｎ＞１）。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。

ア　Ｘと同じ値が配列中にない場合、ｉ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、ｉ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番目にある場合、ｉ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番目にある場合、ｉにはＮが設定されている。

◇問21解説

配列を利用した線形探索に関する問題である。

配列を利用した探索には、線形探索、２分探索、直接探索がある。

線形探索は、先頭または最後尾の要素から順次比較しながら探索する。

２分探索は、中央の値と探索すべき値を比較して、その大小関係で探索範囲を１／２に狭めて等しい値が探索されるまで繰り返す操作で探索する。

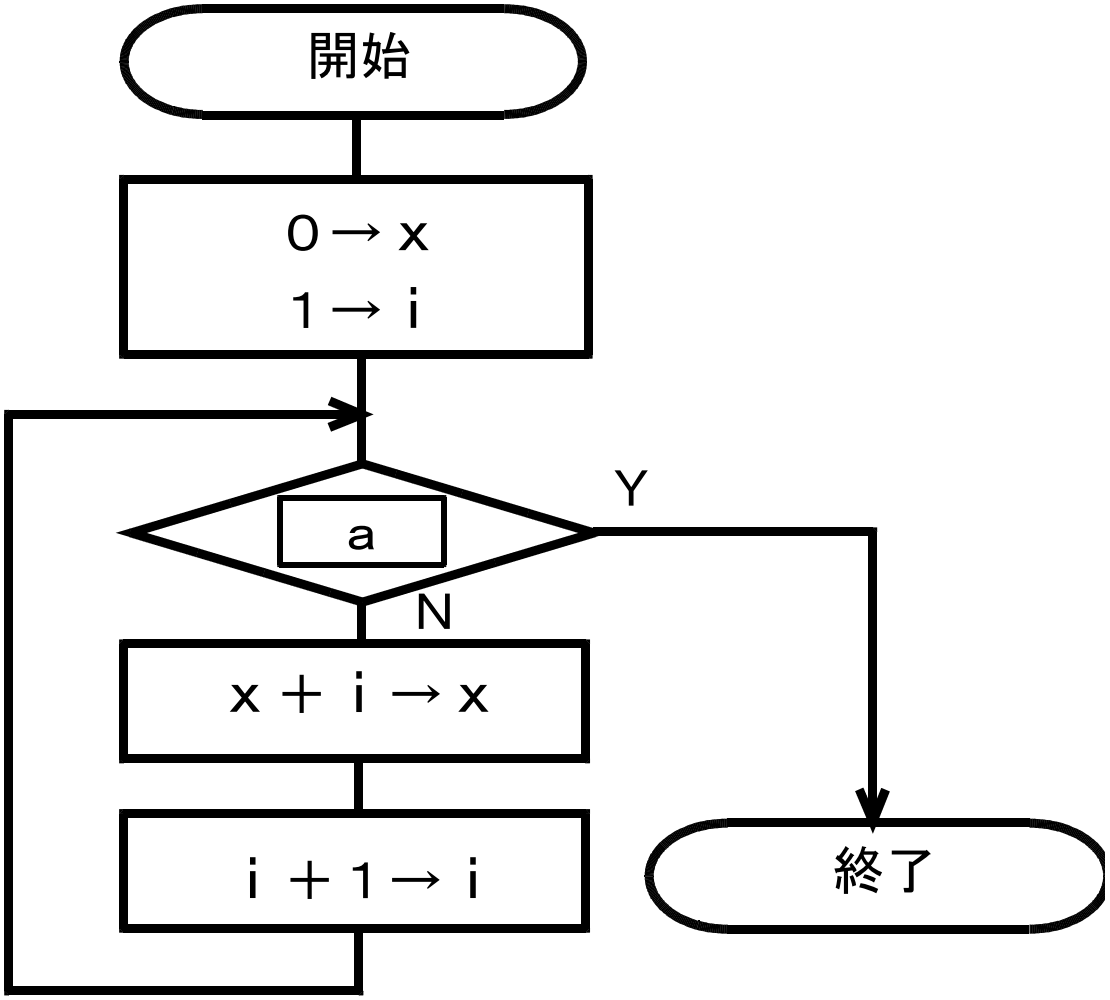
直接探索は、添字で特定し探索する。

アは同じ値が配列にない場合であるから、配列をＮ回調査してｉ＝Ｎ＋１になると繰り返しから脱出する。従って、ｉ＝１ではない。
イはｉにはＮ＋１が設定される。Ｎではない。
ウはＸが見つかると終了するため、ｉ＝１である。正しい。求める答えはウである。
エはｉ＝１であって、ｉ＝Ｎではない。

◇問21解答　ウ

問22問題

流れ図は，１からＮ（Ｎ≧１）までの整数の総和（１＋２＋ … ＋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

◇問22解説

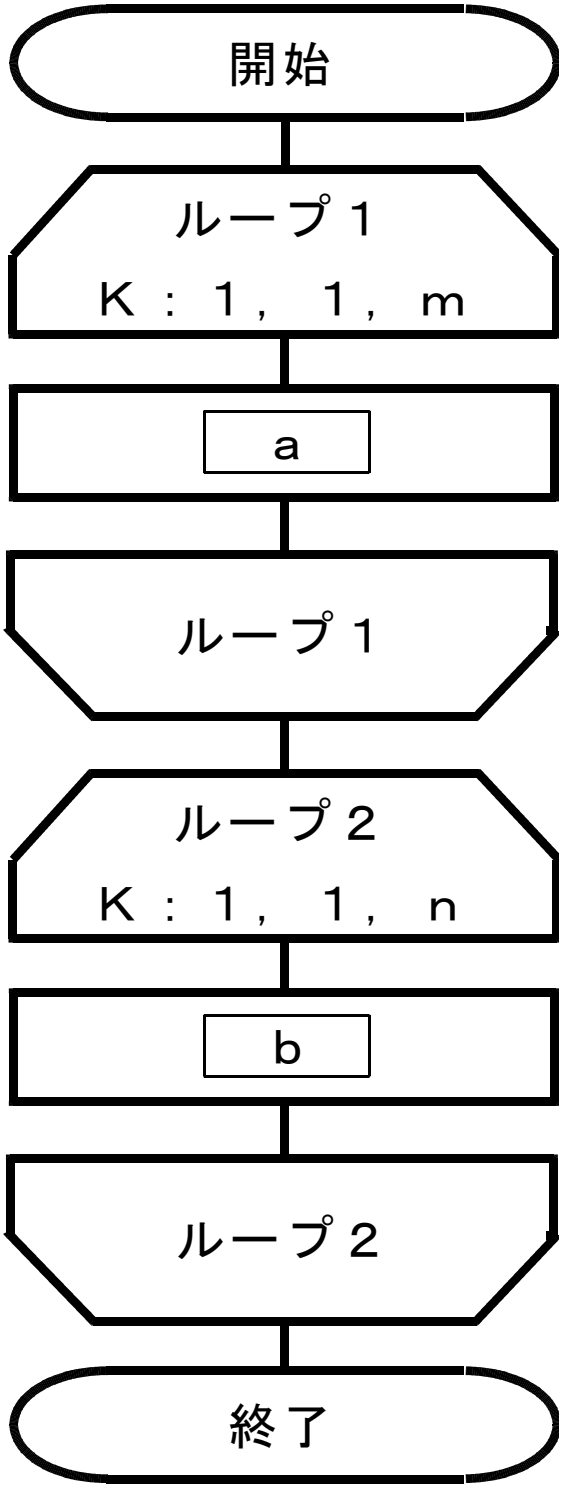
流れ図に関する問題である。

流れ図は１～Ｎまでの整数の総和を求める問題である。

ａは終了条件の判定であるからｉ＞Ｎとなる。求める答えはウとなる。

◇問22解答　ウ

問23問題



長さｍ、ｎの文字列を格納した配列Ｘ、Ｙがある。図はこれらの文字列をこの順に連結した文字列を配列Ｚに格納する処理の流れ図である。ａ、ｂに入れる処理として、正しいものはどれか。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。

	a	b
ア	X（k）→Z（k）	Y（k）→Z（m＋k）
イ	X（k）→Z（k）	Y（k）→Z（n＋k）
ウ	Y（k）→Z（k）	X（k）→Z（m＋k）
エ	Y（k）→Z（k）	X（k）→Z（n＋k）

◇問23解説

２つの配列を結合する流れ図に関する問題である。

結合する手順は次の通りである。

① 最初に大きさmの配列X（k）を配列Zに格納する。
② 配列Xを格納後の配列Zの大きさはmである。
③ その次に大きさnの配列Y（k）を配列Zに格納する。
④ 配列YはZ（m＋１）からZ（m＋n）の要素に格納される。
⑤ 配列X、Yを格納後の配列Zの大きさはm＋nとなる。

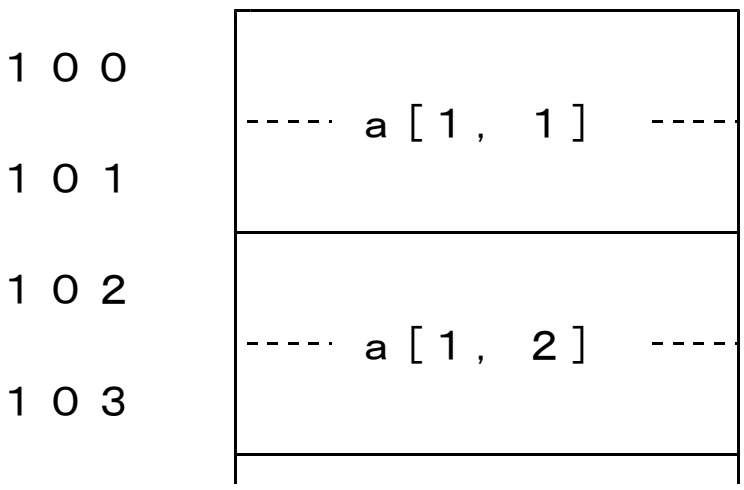
ループ１は、配列Xから配列Zへの文字列の格納処理である。従って、配列Xの長さはmであるから、X（k）→Z（k）となる。

ループ２は、配列Yから配列Zのm＋１番目の要素からn個の要素を配列Zに格納し、配列の要素番号m＋nまでの要素に格納する。即ち、Y（k）→Z（m＋k）となる。

a、bの組み合わせはアとなる。求める答えはアとなる。

◇問23解答　ア

問24問題



１０行１０列の２次元配列ａを、次のようにメモリ上の連続した領域へ行方向に格納するとき、ａ［５，６］が格納される場所の番地はどれか。

ここで、番地は１０進数表示とする。

ア　１４５
イ　１８５
ウ　１９０
エ　２０８

◇問24解説

２次元配列を１次元配列に換算する問題である。

ａ［ｉ、ｊ］を格納するアドレスは次式から計算する。

１００＋２×{（ｊ－１）＋１０×（ｉ－１）}

上記の式を用いてａ[５、６]を計算すると、次のようになる。

１００＋２×(５＋４０)＝１００＋９０＝１９０

求める答えはウとなる。

◇問24解答　ウ

問25問題

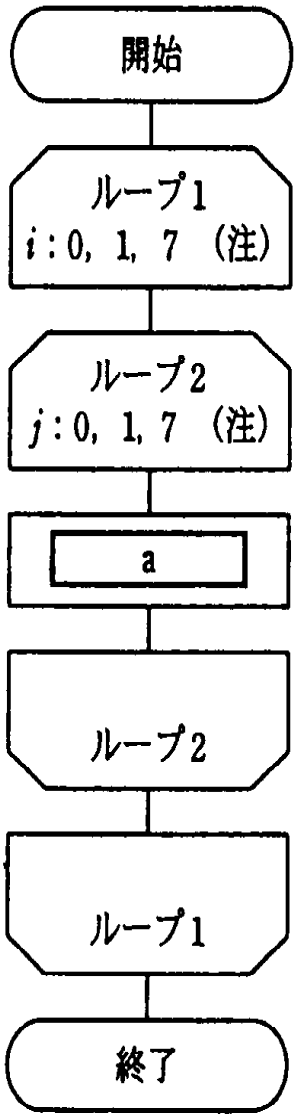


図1　流れ図

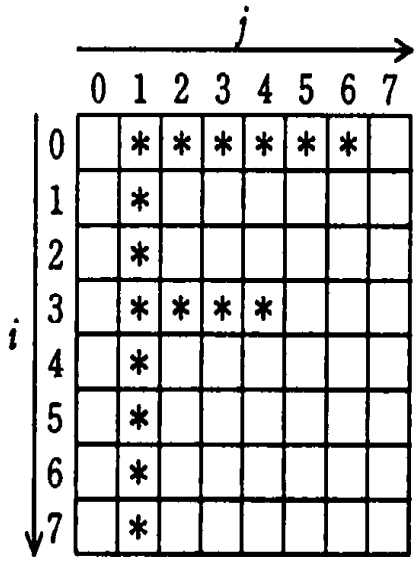


図2　配列Aの状態

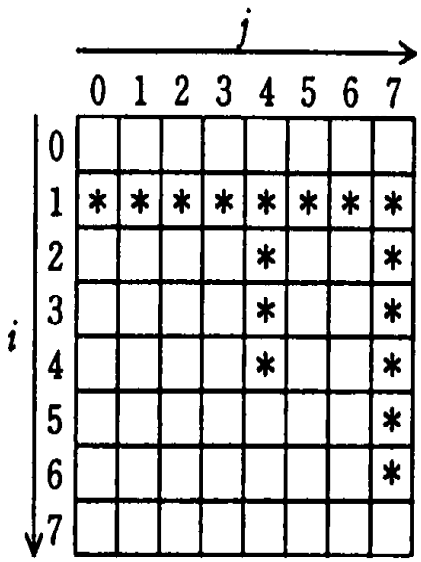


図3　実行後の配列Bの状態

（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。

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に入れるべき操作はどれか。

ここで，配列Ａ，Ｂの要素をそれぞれＡ（ｉ，ｊ），Ｂ（ｉ，ｊ）とする。

ア　Ａ（ｉ，ｊ）→Ｂ（ｉ，７－ｊ）
イ　Ａ（ｉ，ｊ）→Ｂ（ｊ，７－ｉ）

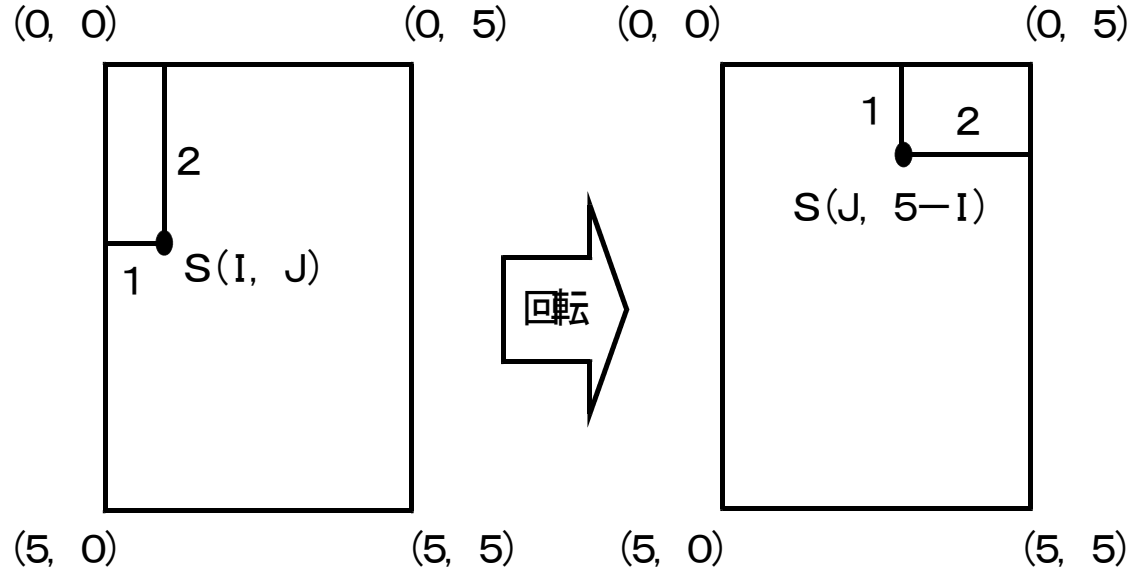
ウ　Ａ（ｉ，ｊ）→Ｂ（７－ｊ，ｉ）
エ　Ａ（ｉ，ｊ）→Ｂ（７－ｉ，７－ｊ）

◇問25解説

２次元グラフの回転に関する問題である。

２次元のグラフを９０度右に回転させる問題である。図は右に９０度回転させた場合の座標の変化を示したものである。

点Ｓの座標（Ｉ，Ｊ）は、９０度右に回転すると新しい座標は（Ｊ，５－Ｉ）に変化する。長さ１、２の変化に着目する。左の図の２はＩに相当し、１はＪに相当する。従って、右に９０度回転後は、１のＪはＩの座標に、２の５－ＩはＪの座標になる。



同様に考えると、図２のｊの値が図３のｉの値になり、図２のｉの値は７－ｊとなる回転となる。従って、式はＡ（ｉ，ｊ）→Ｂ（ｊ，７－ｉ）となり、求める答えはイとなる。

◇問25解答　イ

問26問題

節点１，２，…，ｎをもつ木を表現するために，大きさｎの整数型配列Ａ[１]，Ａ[２]，…，Ａ[ｎ]を用意して，節点ｉの親の番号をＡ[ｉ]に格納する。節点ｋが根の場合はＡ[ｋ]＝０とする。

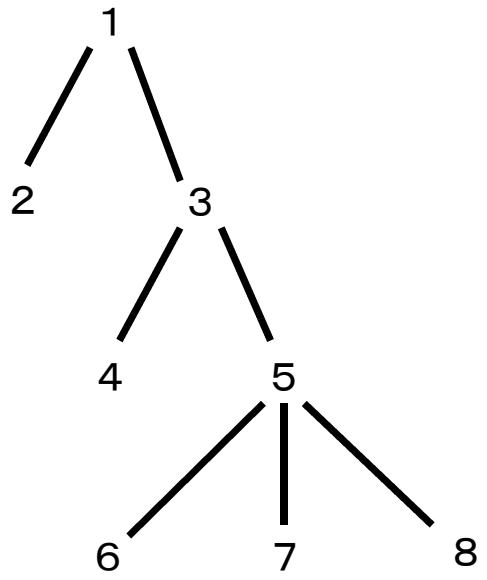
表に示す配列が表す木の葉の数は，幾つか。

i	1	2	3	4	5	6	7	8
A[i]	0	1	1	3	3	5	5	5

ア　１
イ　３
ウ　５
エ　７

◇問26解説

木構造に関する問題である。



与えられた配列表を利用して木構造を作成すると図のようになる。

配列表の配列の値は親の番号である。

ノード１の子は２、３である。
ノード３の子は４、５である。
ノード５の子は６、７、８である。

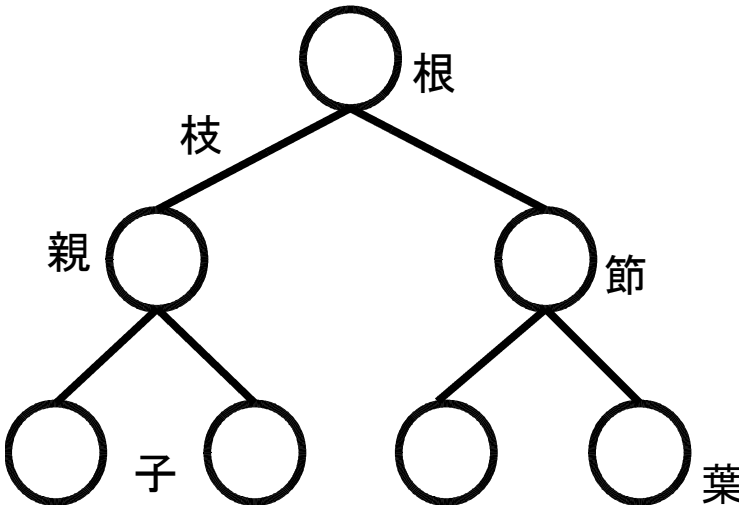
葉は２、４、６、７、８の５個である。求める答えはウとなる。

◇問26解答　ウ

二分木

◆二分木の定義と表現

◇木構造(ツリー構造)の構成要素



要素名	説　　明
ノード(節)	キー値、データが対応
ブランチ(枝)	ノードとノードを結ぶライン
ルート(根)	最上位のノード、親を持たないノード
子	ノードの下部に分岐するノード、 親のノードの下部のノード
親	分岐元のノード、子の上部のノード
リーフ(葉)	最下位のノード、子を持たないノード

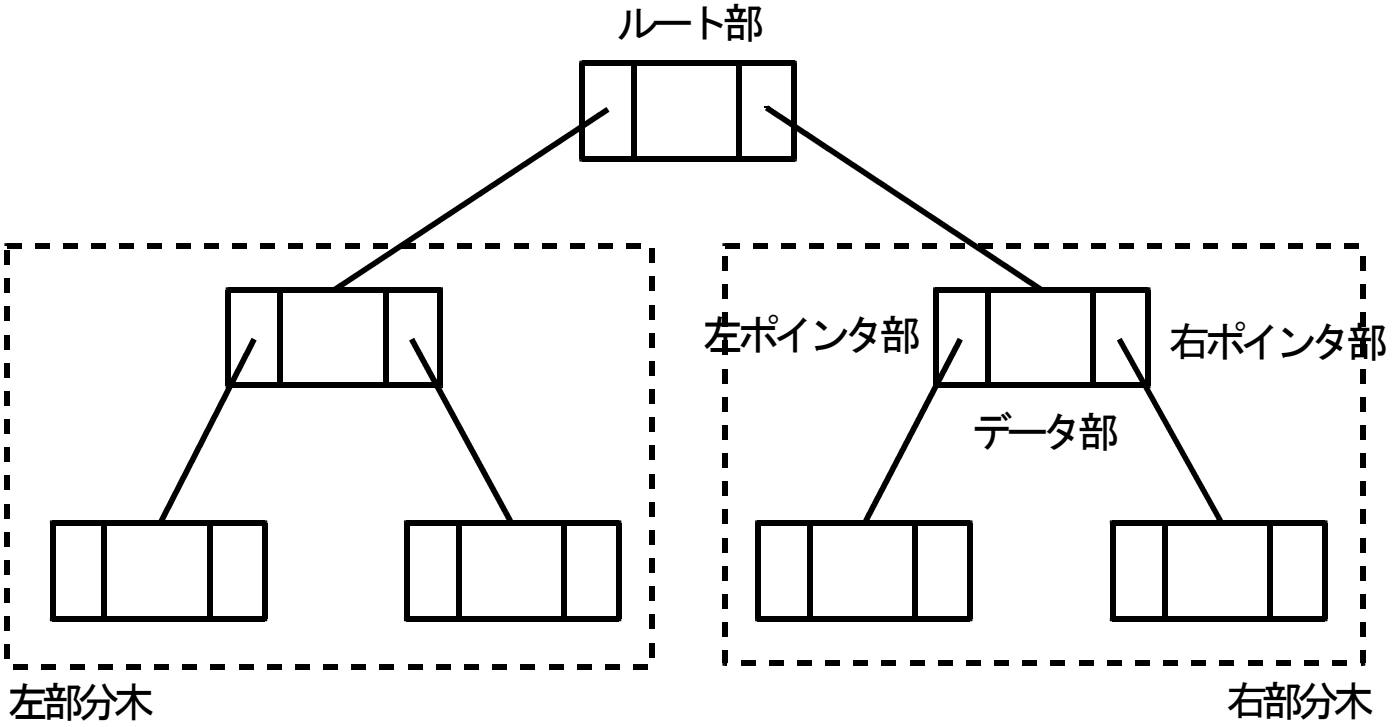
◇二分木の定義

空の木は二分木である。
次のいずれかを満足する節だけからなる木は二分木である。

①　子をもたない木
②　左の子だけもつ木

③　右の子だけもつ木
④　左右二つの子をもつ木

◇部分木



左の子を根とする部分木を左部分木、右の子を根とする部分木を右部分木という。

二分木はデータ部とポインタ部で構成する。

二分木は最高二つの子をもつので、ポインタ部も二つ必要である。左の子のアドレスを示すポインタ部を左ポインタ、右の子のアドレスを示すポインタ部を右ポインタという。

◆二分探索木の定義と探索

◇二分探索木の条件

任意の根とその部分木について、左部分木に含まれる各要素は根の要素よりも小さい。

任意の根とその部分木について、右部分木に含まれる各要素は根の要素よりも大きい。

二分木において、任意に選んだすべてのノードについて、左部分木のすべてのノードの値に対して、任意のノードの値が大きく、右部分木のすべてのノードの値に対して、任意のノードの値が小さい場合、その二分木は二分探索木である。

◇二分探索木を使用した探索

根から枝分かれした二分探索木のデータ構造を利用して、データの探索を行うことができる。

二分探索木の条件に従って、各節にデータを置き、根から葉に向かって辿る経路がそのまま探索のアルゴリズムになる。

節の値と探索データの値を比較して、探索データの値が小さい場合は左の部分木を、探索データが大きい場合は右の部分木を探索する。

◇二分探索木の探索手順

① 探索データをＫとする。

② 根の値とＫを比較する。

③ 根の値＝Ｋならば、探索成功で処理を終了する。

④　根の値＞Ｋならば、探索データは左部分木にあるので左の節を次の根と考えて②に戻る。

⑤　根の値＜Ｋならば、探索データは右部分木にあるので右の節を次の根と考えて②に戻る。

⑥　比較対象となる根が存在しなくなると、該当する探索データが存在しないことになり、探索不成功で処理を終了する。

◇二分探索木の探索時間

探索時間は枝分かれしているレベルの高さに依存する。探索木のレベルが高いほど探索に時間がかかる。二分探索木ではレベルが低いほど効率がよいことになる。

◆二分探索木の挿入

◇挿入位置の探索

挿入位置を見つけるために、挿入するデータを探索データとして二分探索木による探索を実行する。根からはじめて二分探索木を葉に向かって辿り探索する。

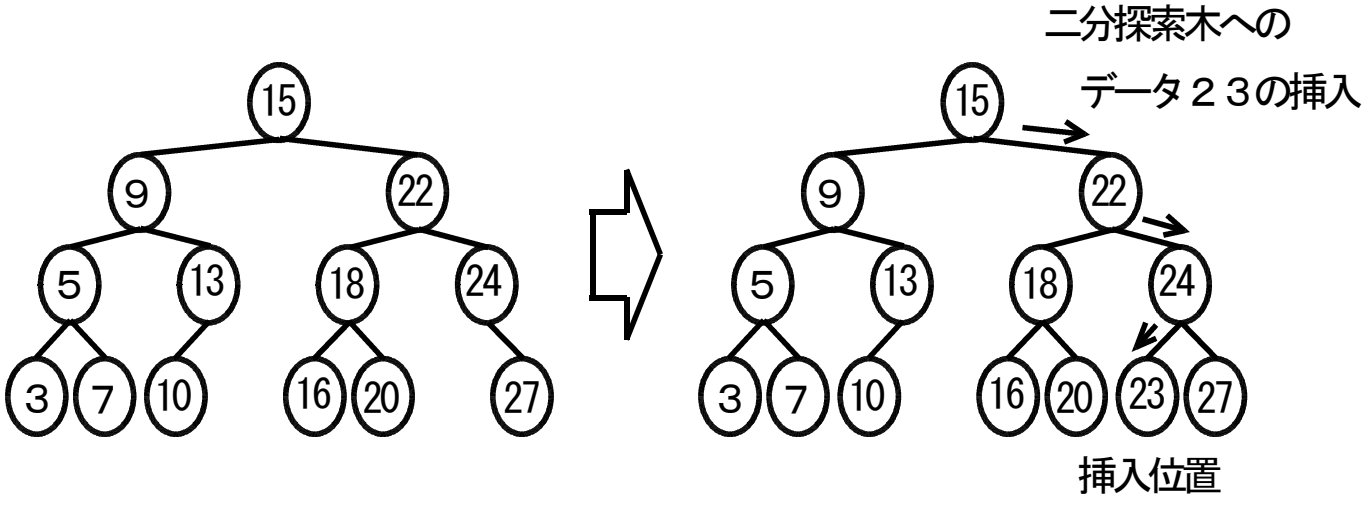
◇二分探索木のデータの挿入手順

① 探索は二分探索木の根から始める。

② 探索データの値と節の値を比較する。

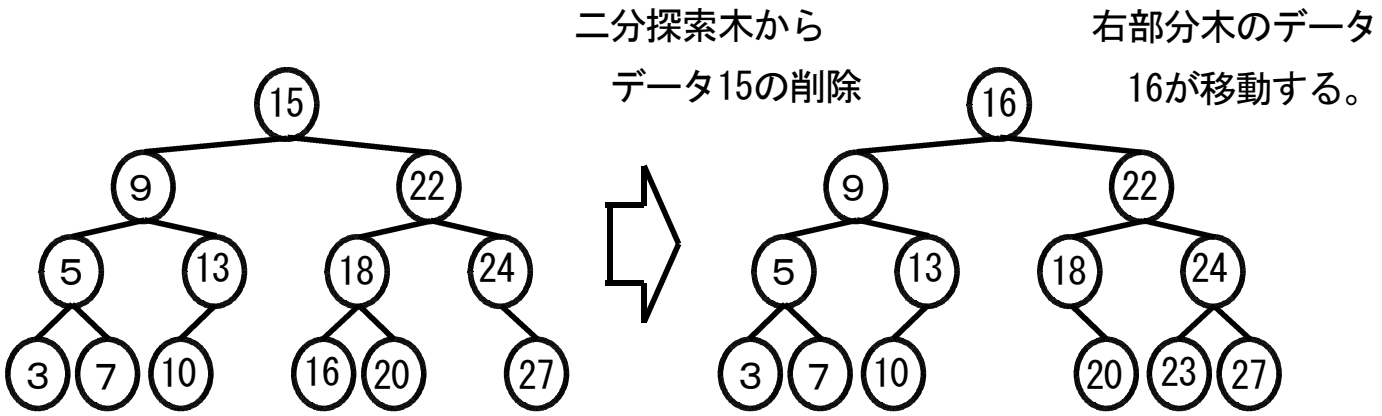
探索データと節の値が一致すると、挿入データが挿入済みであるので処理を終了する。探索データが小さいと、左の節に移動する。探索データが大きいと、右の節に移動する。

③ 探索すべき節または葉が存在しなくなると、その位置に新しいデータを挿入する。探索すべき節が存在すると、②にもどる。



◆二分探索木の削除

◇データの削除



左部分木のデータ13が移動することも可能である。

削除するデータを探索し、削除したデータを節とする右部分木または左部分木を考えて、左部分木の最も大きい値の節かまたは右部分木の最も小さい値の節を削除した節に移動させる。以下同様の処理を繰り返す。

◇二分探索木のデータの削除手順

① 二分探索木を探索し、削除すべきデータを見つけ削除する。

② 削除した節が子を持たないときは何もしない。

③ 削除した節が子を一つ持つとき、その子を親の代わりにする。

④ 削除した節が子を二つ持つ場合は、右部分木に含まれる最小のデータまたは左部分木に含まれる最大のデータを親の代わりとして移動する。

⑤ 移動した子が更に子を持つときは、移動した子が削除されたものとして、同様の処理を繰り返す。

◆二分木の巡回法

◇深さ優先順

深さ優先順の巡回法は、根から始めて左の子からかつ葉の方向に向かって、左の子から右の子の順に巡回を行う方法である。

木の深さの方向に対して巡回を繰り返すことになる。

◇巡回の順番を決める方法

① 先行順

節点に来たときに先に順位を割り当てる方法である。

親に番号を与え、左の子、右の子の順番になる。

② 中間順

節点に来たときにその左の子を根とする部分木を巡回し終えてから、戻ってきたときに順位を割り当てる方法である。

節点の左部分木に順位を割り当てた後に、その節点に順位を割り当てる。

左の子に番号を与え、親、右の子の順番になる。

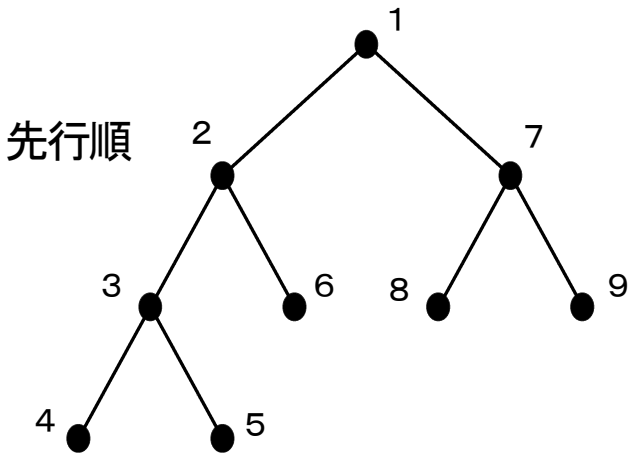
③ 後行順

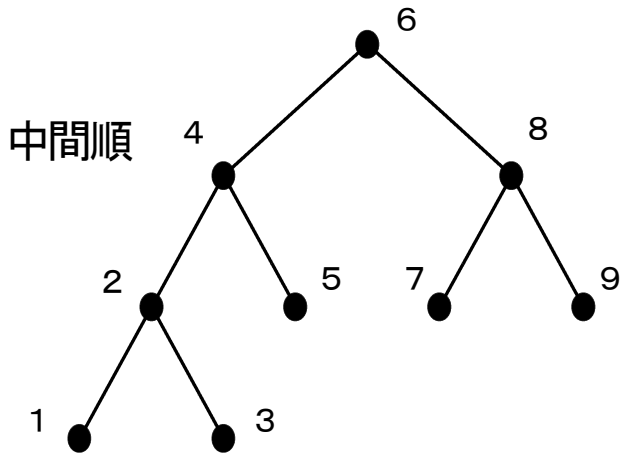
節点に来たときに自分を根とする部分木を巡回し終えてから、その節点に順位を割り当てる方法である。

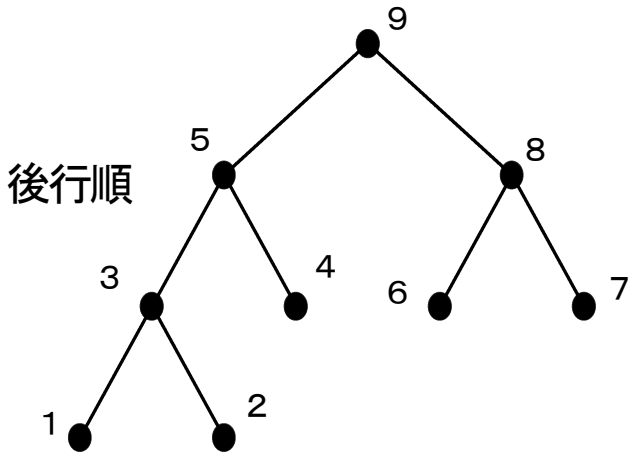
節点の左部分木、右部分木に順位を割り当てた後に、その節点に順位を割り当てる。

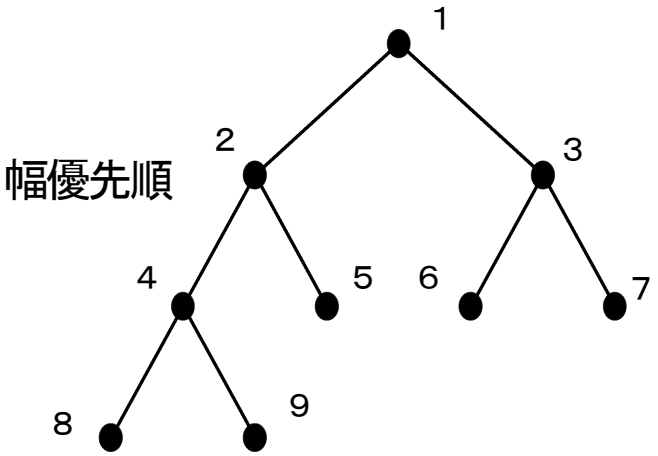
左の子に番号を与え、右の子、親の順番になる。

図は先行順、中間順、後項順の場合の、各ノードに与えた番号を示す。









◇幅優先順

幅優先順の巡回法は、根から始まって、深さの浅い方からかつ左から巡回する方法である。木の幅の方向に対して巡回を繰り返すことになる。

◆二分木と四則演算

◇二分木で表現する算術式

二分木の葉の部分に変数を配置し、各節に四則演算の優先順位に従って、演算子を配置すると二分木を使用して算術式を表すことができる。

◇四則演算式の表記法

四則演算式の表記法には次の３通りの方法がある。

① 前置記法

演算子をオペランドの前に書く方法。

＋ａ＊－ｂｃｄ

② 中置記法

演算子をオペランドの間に書く方法。

（ａ＋（（ｂ－ｃ）＊ｄ））

③ 後置記法

演算子をオペランドの後に書く方法。

ａｂｃ－ｄ＊＋

二分木を使用した算術式と二分木の巡回法を組み合わせると、３の表記法を求めることができる。

◆逆ポーランド記法への活用

◇四則演算と逆ポーランド記法

逆ポーランド記法は四則演算の後置記法である。

四則演算式を逆ポーランド記法に変換する場合、二分木巡回の後行順を使用する。

変数を葉に、演算子を節に配置し、四則演算の優先順位が成立するように節間に枝を設ける。

◇Ｘ＝（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）の二分木とならい

①　＝を根に配置する。

②　Ｘを根の左側の葉に配置する。

③　Ａ、Ｂ、Ｃ、Ｄ、Ｅを右部分木の葉に配置する。

④　四則演算子の優先レベルの高いものから演算子で結合する。

⑤　完成した二分木に後行順のならいを適用すると、

ＸＡＢＣ／－ＤＥ＋＊＝

の逆ポーランド記法を求めることができる。

◇二分木巡回法の後行順から逆ポーランド記法

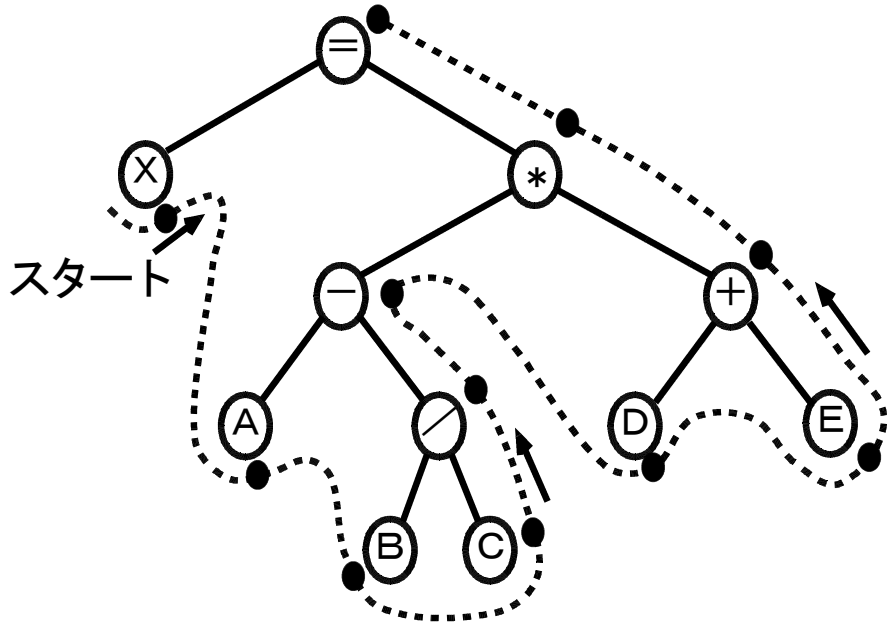
作成した図の二分木に、二分木巡回法の後行順を用い、ノードの番号順に左から順次記号を並べると逆ポーランド記法を求めることができる。

図のスタートの位置から、点線に従って、各ノードの値または演算子を順次並べると、次のようになる。

Ｘ＝（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）の逆ポーランド記法は

ＸＡＢＣ／－ＤＥ＋＊＝

となる。



○印は節、葉
実線は二分木の枝
点線はならい
矢印はならいの方向

◇スタックを利用した演算

逆ポーランド記法を左から順に解析し、数値であればスタックに格納し、演算子であれば、スタックから２つの数値を取り出して演算を行い、演算結果をスタックに格納する。

２つの数値の演算は、

（後から取り出した数値）　演算子　（先に取り出した数値）

の方法で行う。

◇演算のアルゴリズム

① スタックポインタを初期化する。

② 先頭から＝が表れるまで、③、④の処理を繰り返す。

③ 数値ならば、スタックにPUSHする。

④ 演算記号ならば、次の処理を行う。

ⓐ スタックから数値を２つPOP する。
ⓑ 取り出した２数を(後から取り出した数値)演算子(先に取り出した数値)の方法で演算する。
ⓒ 演算結果をスタックにpushする。

⑤ ＝の時演算結果をPOP して処理を終了する。

◇スタックを使用した演算具体例

元の四則演算　Ｘ＝（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）

逆ポーランド記法　ＸＡＢＣ／－ＤＥ＋＊＝

◇演算手順

① Ｘ、Ａ、Ｂ、Ｃの順にスタックにＰＵＳＨする。

② 次は演算子の／であるから、Ｃ、ＢをＰＯＰし、Ｂ／Ｃを計算し、その結果をＰＵＳＨする。

③ 演算子－であるから、Ｂ／Ｃ、ＡをＰＯＰし、Ａ－Ｂ／Ｃを計算し、その結果をＰＵＳＨする。

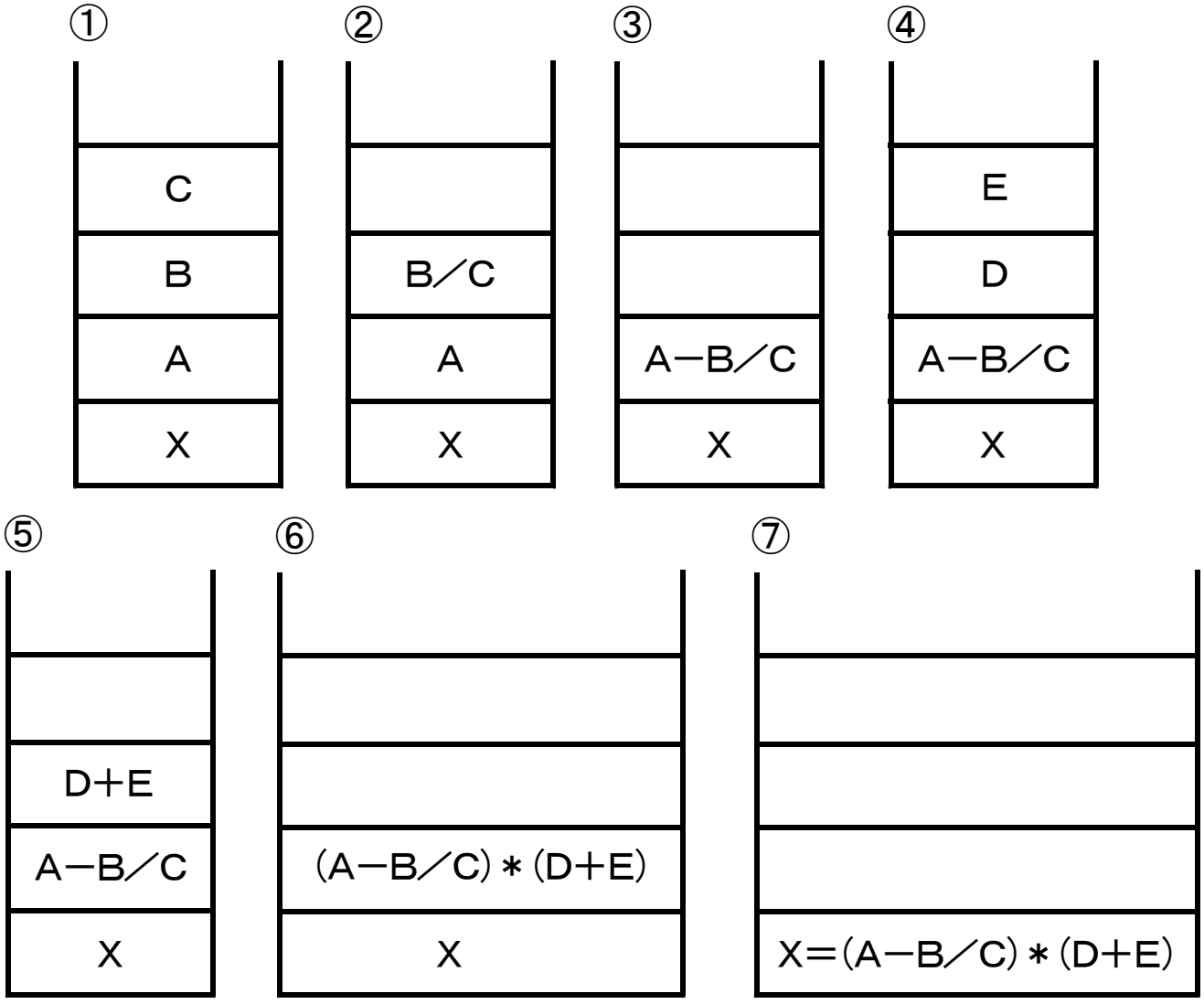
④ Ｄ、Ｅの順にスタックにＰＵＳＨする。

⑤　演算子＋であるから、Ｅ、ＤをＰＯＰし、Ｄ＋Ｅを計算し、その結果をＰＵＳＨする。

⑥　演算子＊であるから、Ｄ＋Ｅ、Ａ－Ｂ／ＣをＰＯＰし、（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）計算し、その結果をＰＵＳＨする。

⑦　演算子＝であるから、（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）、ＸをＰＯＰし、次の答えを求める。

Ｘ＝（Ａ－Ｂ／Ｃ）＊（Ｄ＋Ｅ）



バランス木とヒープ木

◆バラス木とＡＶＬ木

◇バランス木

二分木や多分木はデータの挿入や削除を繰り返していると、木の形が変形し、アンバランスな木の形になる。

このようなアンバランスな状態を避ける仕組みを持った木がバランス木である。

バランス木はデータの挿入や削除によって、一定の条件以上になると、分裂や統合によって木全体の構造が再調整する機能をもっている。

バランス木は探索木の一種である。

◇ＡＶＬ木

ＡＶＬ木は、二分木をもとにしたバランス木である。

各節点において、その部分木の左と右の深さの差が高々一つしか異ならないような二分木である。

各部分木は左と右の深さの情報をもっている。データの挿入や削除が行われる際に、木の構造を調整する。

◆完全二分木

◇完全二分木とは

根から葉までの経路の長さが等しい二分探索木を完全二分木という。

完全二分木が成立するのは、節の数が2^K-1個の時である。Kは階層数を示す。

通常は、根から最も遠い葉までの経路の長さと、根から最も近い葉までの経路の長さの差が１以下である二分木を完全二分木と定義する。

◇完全二分木の階層と節点の個数

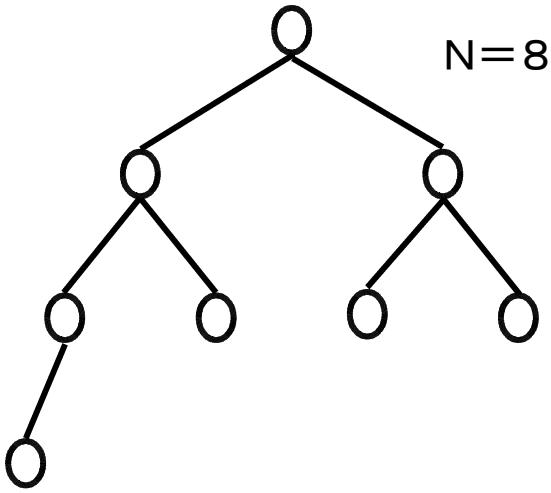
階層数	節点の個数		
	最小	最大	数式
１	１	１	$1=2^1-1$
２	２	３	$3=2^2-1$
３	４	７	$7=2^3-1$
４	８	１５	$15=2^4-1$
⋮	⋮	⋮	
⋮	⋮	⋮	
Ｋ－１	2^{K-2}	$2^{K-1}-1$	$2^{K-1}-1$
Ｋ	2^{K-1}	2^K-1	2^K-1

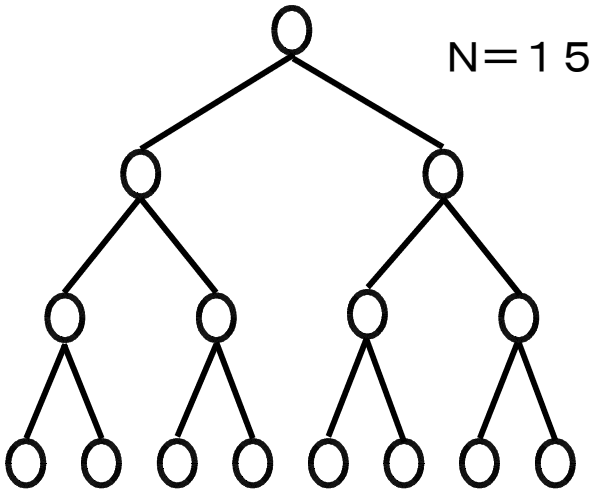
階層数をＫとし、節点の個数をＮとすると、次の式が成立する。

$$N=2^K-1$$

◇階層数Ｋ＝４の場合の完全二分木

階層数４の場合の完全二分木を図で示す。





階層数が４の場合、ノードの数の最小は８であり、最大は１５となる。

◇階層数Ｋの場合のノードの数Ｎ

階層数Ｋの場合のノードの最小数は次の式で求めることができる。

$$N = 2^{K-1}$$

階層数Ｋの場合のノードの最大数は次の式で求めることができる。

$$N = 2^{K} - 1$$

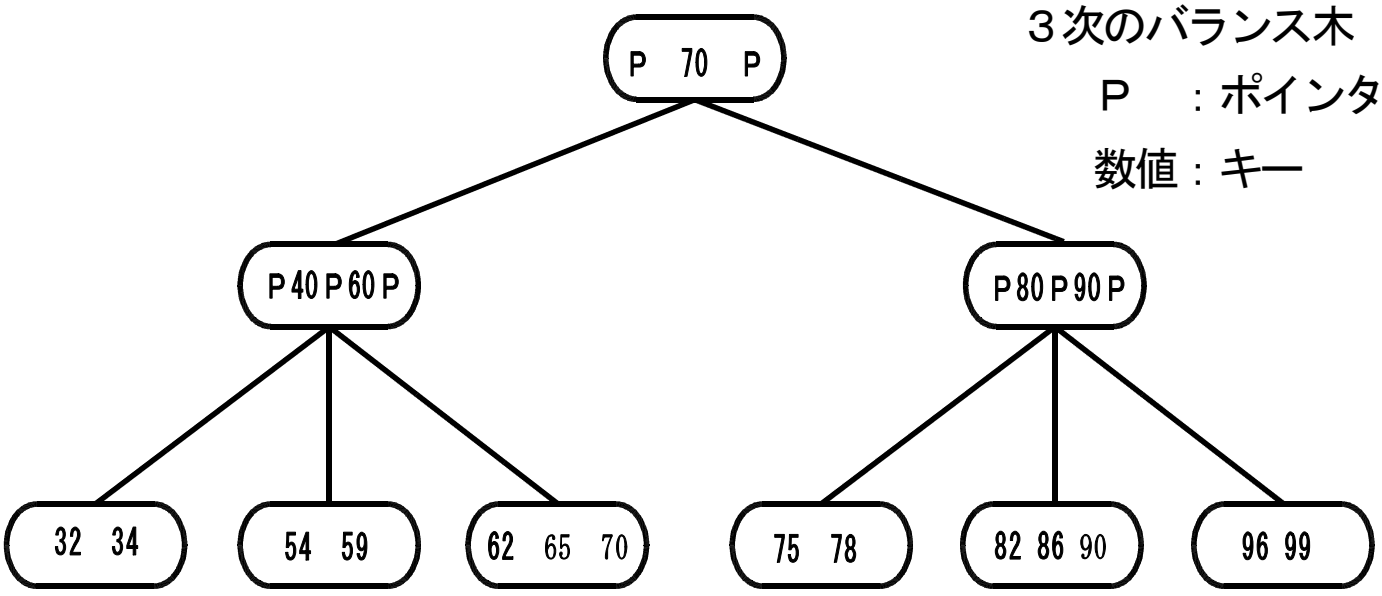
◆多分木をもとにしたバランス木

◇m分木

ｍ分木は一つの節が最大ｍ個の子をもつことができる木構造である。

レベル２の３次のバランス木の例を示す。レベル２は階層の深さ、３次はポインタの数を示す。

階層の深さは、階層数－１となる。ポインタ数は一つの節が持つ子の最大数になる。



図のルート部は、値７０よりも小さい節の値は左部分木に、大きい値は右部分木に含まれていることを示している。

左下の節の内容は、値が４０、６０の２個存在し、その子が３個あり、４０未満は左の子に属し、４０～６０の間の子は中の子に属し、４０～７０の間の子は右の子に属することを示している。

葉の部分は、その葉に属する値を示している。

◇ｍ次のバランス木の条件

① 根は２～ｍ個の子を持つ。

② 根と葉以外の節は([ｍ／２]～ｍ)個の子を持つ。[ｍ／２]はｍ／２以上の最小の整数。

③ 根からすべての葉までの経路は長さは等しい。

④ ｋ個の子を持つ親は、(ｋ－１)個のキーを持っている。

⑤ すべての葉は同じレベルにあって、キーは持たない。

◇バランス木の構造

２分木は節が値を持っているが、バランス木は値を持つのは葉だけであり、葉以外の節は値を持たずキーだけを持っている。キーとはポインタで示す節の持つ値の範囲を示す。

目的の値を求める場合、根から探索を開始し、キー値を比較し、各節のキー値との大小関係によって、対応するポインタを利用して次のキー値を求める操作を繰り返し、葉に到達する。

キー値５４を求める場合、根の７０と比較し、５４＜７０であるから、左の節に移動し、節のキー値と比較する。４０＜５４＜６０であるから、節の真ん中の子にたどり着く、その葉に５４が存在することになる。

目的の値が葉に到着する途中の節にある場合の値は、その左の子の最も右のポインタに属するグループにその値は存在する。

根にあるキー７０の値は、６２、６５、７０の葉に、キー値９０は、８２、８８、９０の葉に存在する。

◆５次の多分木をもとにしたバランス木

◇５次の多分木の構造

５次の多分木は各節にキーが４個、ポインタが５個あるバランス木である。

すべてのレコードは葉の部分にあって、各ノードに配置されたキーを辿ることによって目的のレコードに到着することができる。

値を探索する手順は、キー値を利用して、バランス木の構造で示した手順で求めることができる。

５次のバランス木に含まれるデータの最大数を求めることができる。データの最大数は階層の深さと関係する。

多次のバランス木に含まれる最大のデータ数は、次数と階層数によって決まる。

◇深さが２の場合のキーの数

ルート部のキーの数は４個である。次のレベルはノードが５個になる。更に、その下のレベルのノードの数は、５×５＝２５となり、ルート部からのノードの総数は

１＋５＋２５＝３１個

となる。

各ノード部にはキーが４個格納されるから、格納できるキーの総数は、次のようになる。

（１＋５＋２５）×４＝３１×４＝１２４個

◇深さが３の場合のキーの数

ノードの数が２５×５＝１２５個追加されて、ノードの総数は、３１＋１２５＝１５６となり、格納できるキーの総数は、次のようになる。

（１＋５＋２５＋１２５）×４＝６２４個

レベルが３の場合、葉は１２５個で、それぞれの葉の部分に５個のレコードが含まれ、最右端の葉のみが４個のレコードとなる。従って、全レコード数は、次のようになる。

１２４×５＋４＝６２０＋４＝６２４

レコードの総数とキーの総数が一致する。

◇深さがＬの場合のキーの数

ノードの数Ｎは次の式で求めることができる。

$N = 1 + 5^{1} + 5^{2} + \cdots + 5^{L}$

データ数Ｓは次の式になる。

$S = N \times 5$

ｍ次、深さＬの多分木のノード数Ｎ、データ数Ｓは、次のようになる。

$N = 1 + m + m^{2} + \cdots + m^{l}$

$S = N \times m$

◆ヒープ木の定義と操作

◇ヒープ木の定義

ヒープ木は完全二分木の各節に一連番号を付け、その番号をもとに、完全二分木のデータ構造を配列で表現したものである。

二分木に番号をつけ、どの親子の値をとってみても必ず

「親の値＜子の値」(または「親の値＞子の値」)

の関係になっている配列をヒープ木という。

親の節の値が子の節の値より小さくなるようにすれば、根の値は二分木の中で最小になる。逆にすれば最大になる。

◇一連番号付けの規則

① 根には番号１をつける。

② 番号ｋの節の左の子には２ｋ、右の子には２ｋ＋１という番号をつける。

③ 親の節の値は子の節の値よりも小さい(または大きい)値となるようにする。

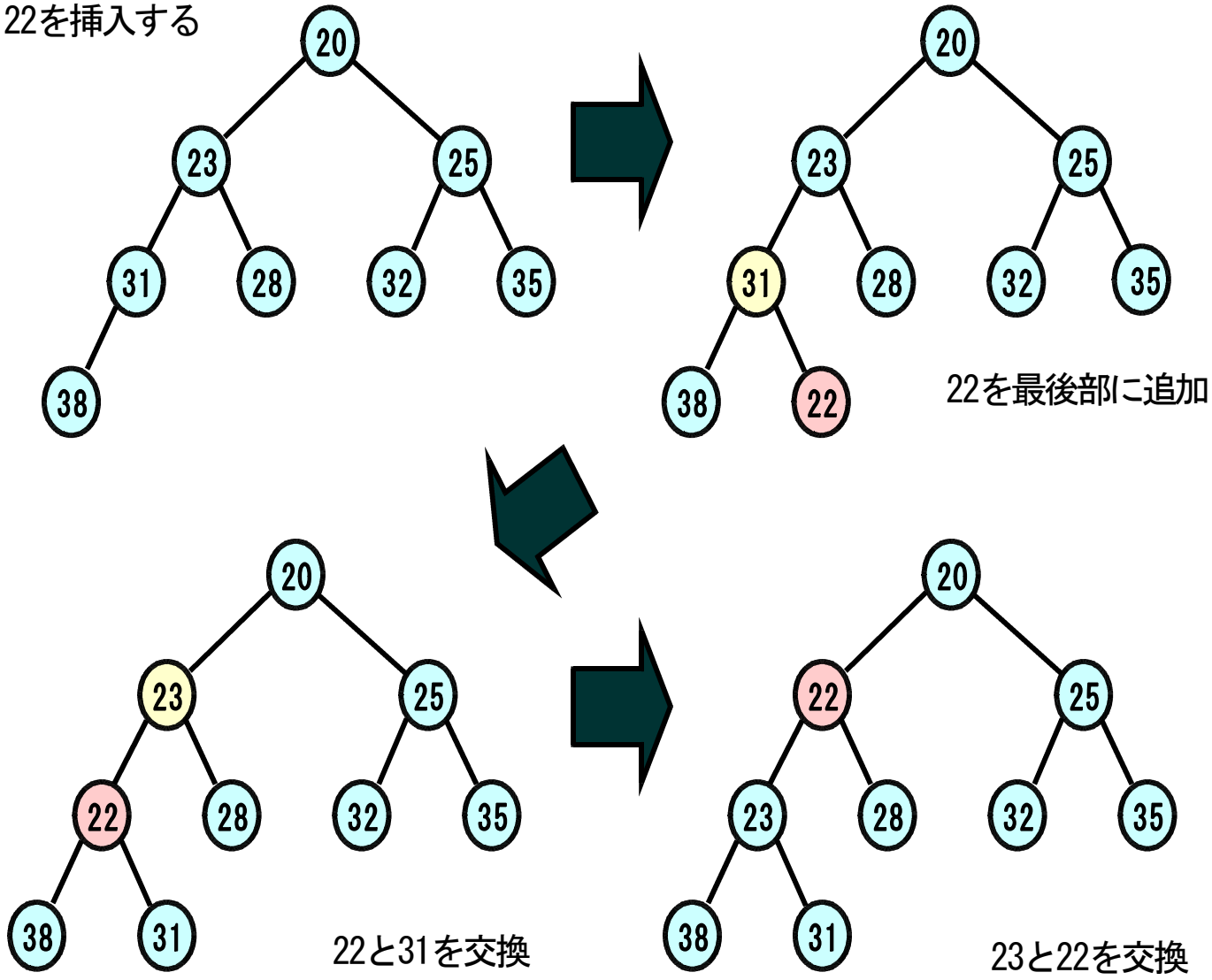
◇ヒープの探索と挿入

ヒープの探索は線形探索と同じである。

節につけられた一連の番号を順次たどっていきながら、探索データと各要素を比較し値が一致すれば探索成功、最後まで探索して一致するデータが存在しなければ探索不成功になる。

ヒープ木に値２２のデータを挿入する場合の具体例を次に示す。挿入の手順は矢印の方向に進められる。

挿入するデータ２２を配列の最後の要素に追加し、二分木の各ノードの値がヒープの条件を満たすように操作する。



挿入する値２２を値３１の右の子に入れる。３１と２２を交換する。２３と２２を交換する。この状態の二分木でヒープ木の条件を満たす。

ヒープ木は同じレベルの節の間、同じ親の子の間での大小関係は存在しない。

◇親の値≦子の値の場合の挿入手順

① 挿入する要素を最後部へ無条件に追加する。

最後部は配列の最後の要素に相当する。

② 追加した要素の値がその親の値よりも小さいと親と子を交換する。

③ 新しく親となった値とその親の値を比較して、子の方が小さければその親と子を交換する。

④ この比較・交換を根に向かって行い、「親の値≦子の値」の関係になるまで続ける。

◇ヒープの削除

ヒープの削除は最小の値(または最大の値)の要素を取り出すことである。

最小の値の要素(最大の値の要素)はヒープの性質上、根にある。従って、ヒープの削除は根の削除に相当する。

ヒープ木を使用して、データを昇順または降順に整列することができる。ヒープ木で作成される根の値を順次取り出して、データを並べることによって可能となる。

◇親の値≦子の値の場合の削除手順

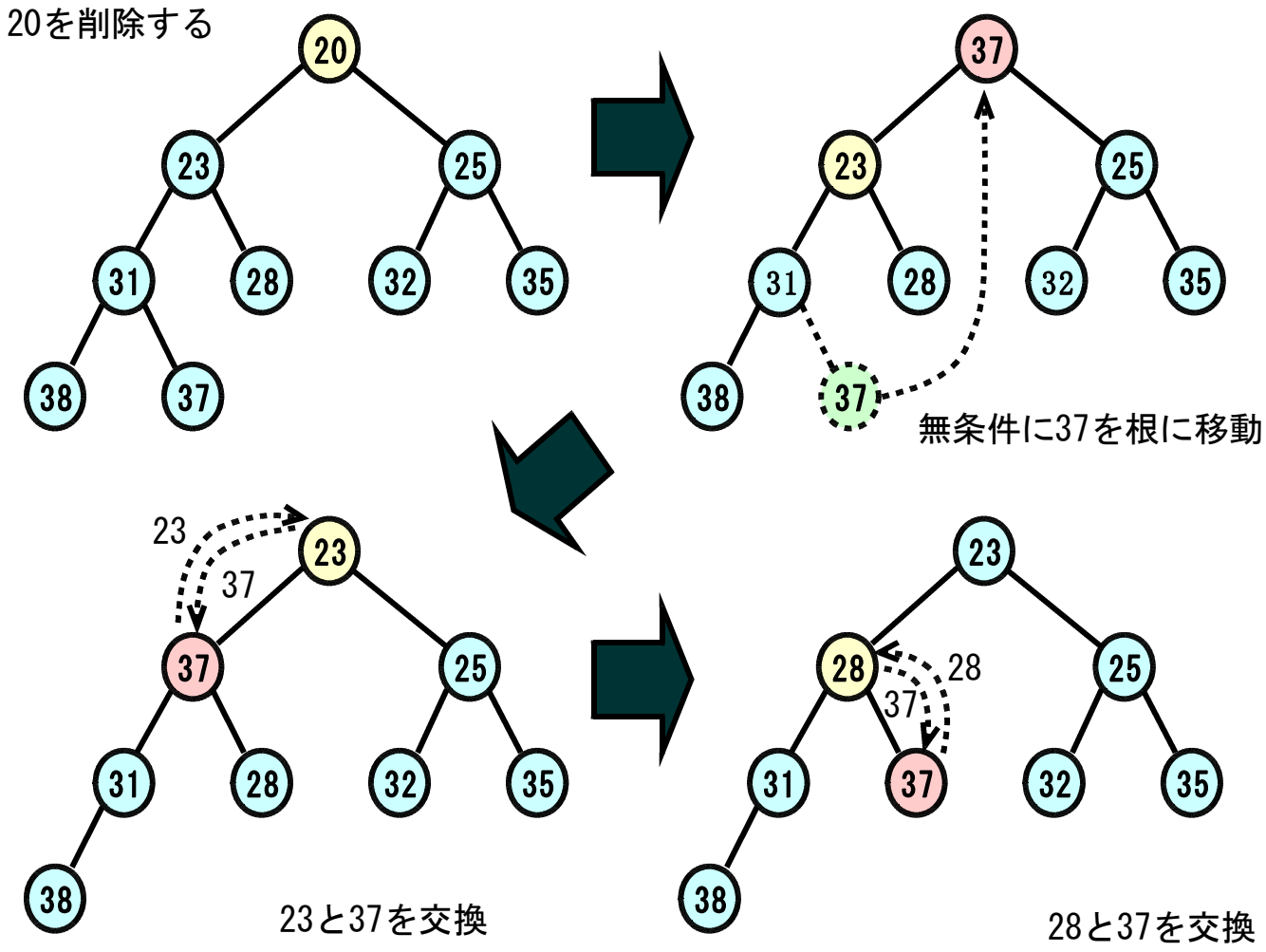
① 根の部分の値を削除する。

② 根の部分にヒープの最後の要素を移動する。

③ 根から順次「親の値≦子の値」が保たれるように要素の値を比

較・交換する。

④　ヒープ木のすべての親子関係が「親の値≦子の値」になると、その木が新しいヒープ木となる。



ヒープ木から２０の値を削除する。削除後、最後尾の値３７を根に移動する。ヒープ木の条件を満たすように節の値を交換する。

３７と２３を交換する。３７と２８を交換する。この状態で二分木はヒープ木の条件を満たす。

親の値≦子の値の場合、親と交換の対象になるのは、子の小さい方の値となる。

データ構造問題２

問１問題

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り，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　データ部と一つのボ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。

◇問１解説

データ構造に関する問題である。

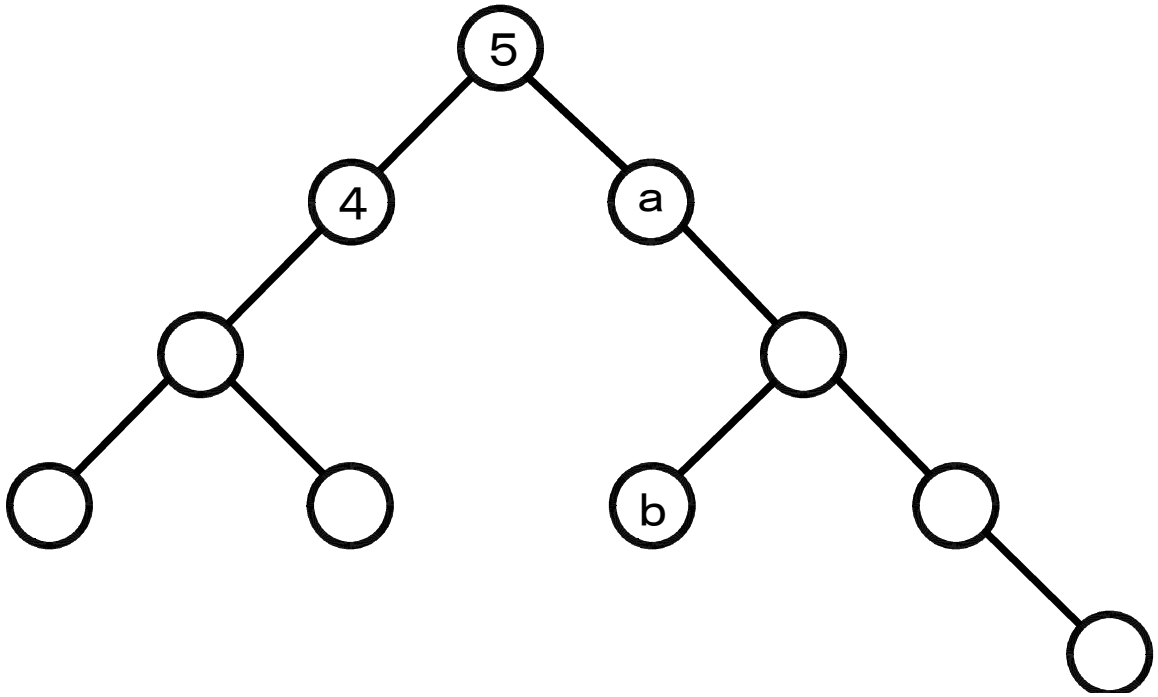
木構造は節点と枝が木の形をした構成になっている。頂点の節点を根と呼び、末端の節点を葉と呼ぶ。上位側の節点を親、下位側の節点を子、同列の節点を兄弟と呼ぶ。根から特定の節点まで行く枝の数を深さという。階層の上位から下位に節点を辿ることによって、データを取り出すことができる。

アの上位から下位に節点を辿ることによって、データを取り出す内容は木構造に関するものである。求める答えはアとなる。
イ、ウは順編成のファイルのデータ構造である。
エは片方向リストのデータ構造である。

◇問１解答　ア

問２問題

１０個の節（ノード）からなる次の２分木の各節に，１から１０までの値を一意に対応するように割り振ったとき，節ａ，ｂの値の組合せはどれになるか。ここで，各節に割り振る値は，左の子及びその子孫に割り振る値より大きく，右の子及びその子孫に割り振る値より小さくする。



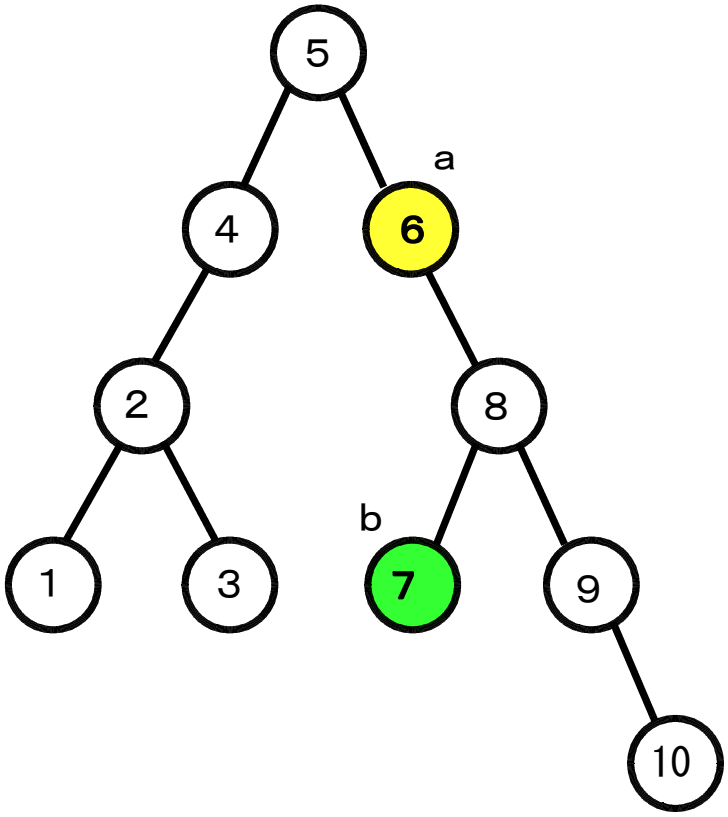
ア　ａ＝６、ｂ＝７
イ　ａ＝６、ｂ＝８
ウ　ａ＝７、ｂ＝８
エ　ａ＝７、ｂ＝９

◇問２解説

探索２分木に関する問題である。

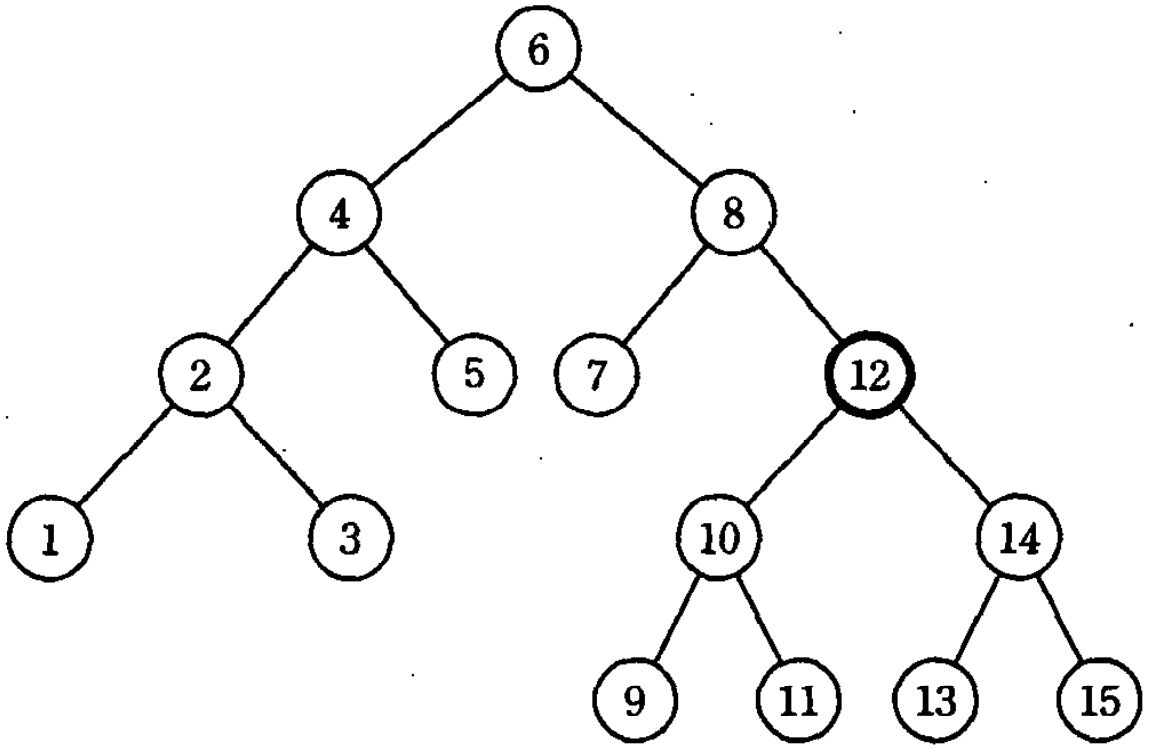
与えられた問題の２分木を完成すると図のようになる。

ａは黄色の箇所、ｂは緑色の箇所であるから、ａ＝６、ｂ＝７となる。求める答えはアとなる。



◇問２解答　ア

問３問題



図の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、削除された節点の位置にどの要素を移動すればよいか。

ア　９
イ　１０
ウ　１３
エ　１４

◇問３解説

２分探索木に関する問題である。

２分探索木の条件

① 任意の根とその部分木について、左部分木に含まれる各要素は根の要素よりも小さい。

② 任意の根とその部分木について、右部分木に含まれる各要素は根の要素よりも大きい。

１２のノードを削除する場合であるから、左部分木の最大値は１１、右部分木の最小値は１３となり、いずれかが１２のノードの部分に移動する。

解答群の中には１３があるから、求める答えはウとなる。

◇問３解答　ウ

問４問題

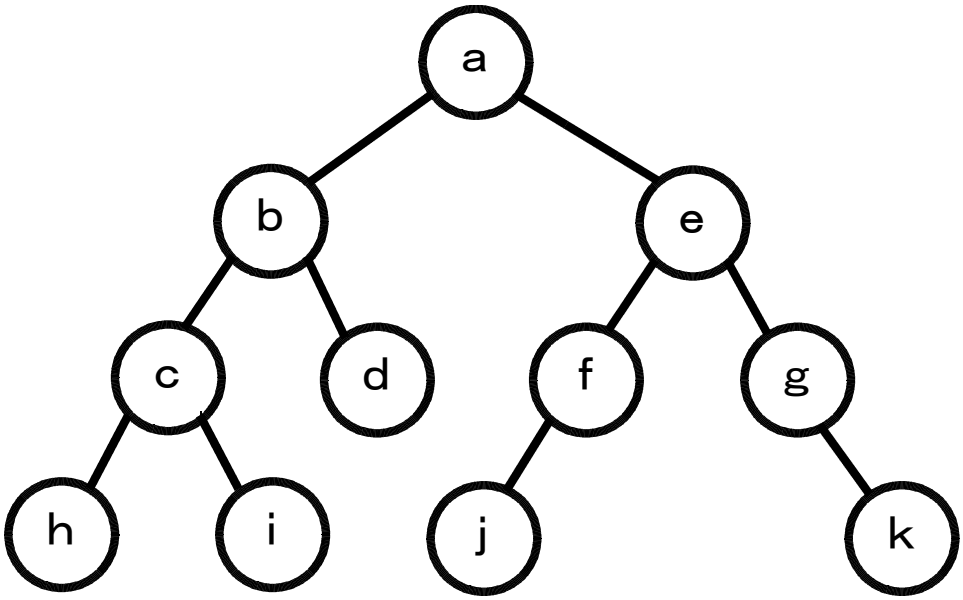
それぞれの節から分岐する枝が２本以下である木を２分木という。２分木は一つの節とその左部分木と右部分木からなり、走査の方法にはその順序によって次の三つがある。

（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分木に対して後順に走査を行い、節の値を出力した結果はどれか。



ア　a b c h i d e f j g k
イ　a b e c h i d f j g k
ウ　h c i b d a j f e g k
エ　h i c d b j f k g e a

◇問４解説

二分木の後行順操作に関する問題である。

後行順は、節点に来たときに自分を根とする部分木を巡回し終えてから、順位を割り当てる方法である。

各節に順位を割り当て、その順位に従って出力するとよい。

文字の出力順序は、ｈｉｃｄｂｊｆｋｇｅａとなり、求める答えはエとなる。

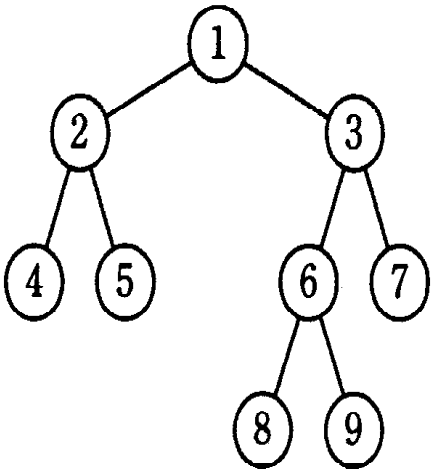
◇問４解答　エ

問５問題

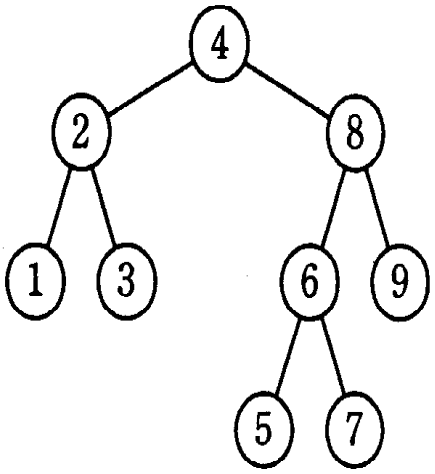
２分探索木として適切なものはどれか。

ここで，１～９の数字は，各ノード(節)の値を表す。

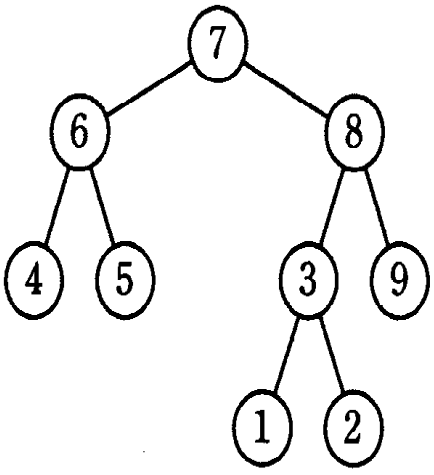
ア



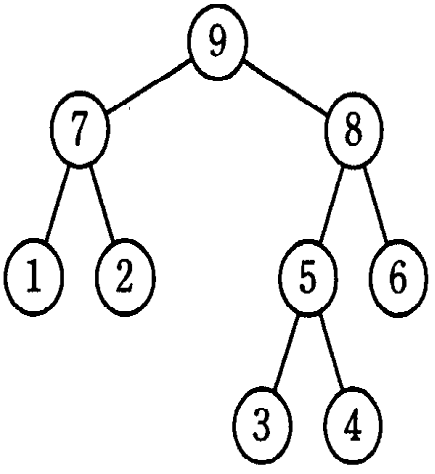
イ



ウ



エ



◇問５解説

２分探索木に関する問題である。

２分探索木の条件

① 任意の根とその部分木について、左部分木に含まれる各要素は根の要素よりも小さい。

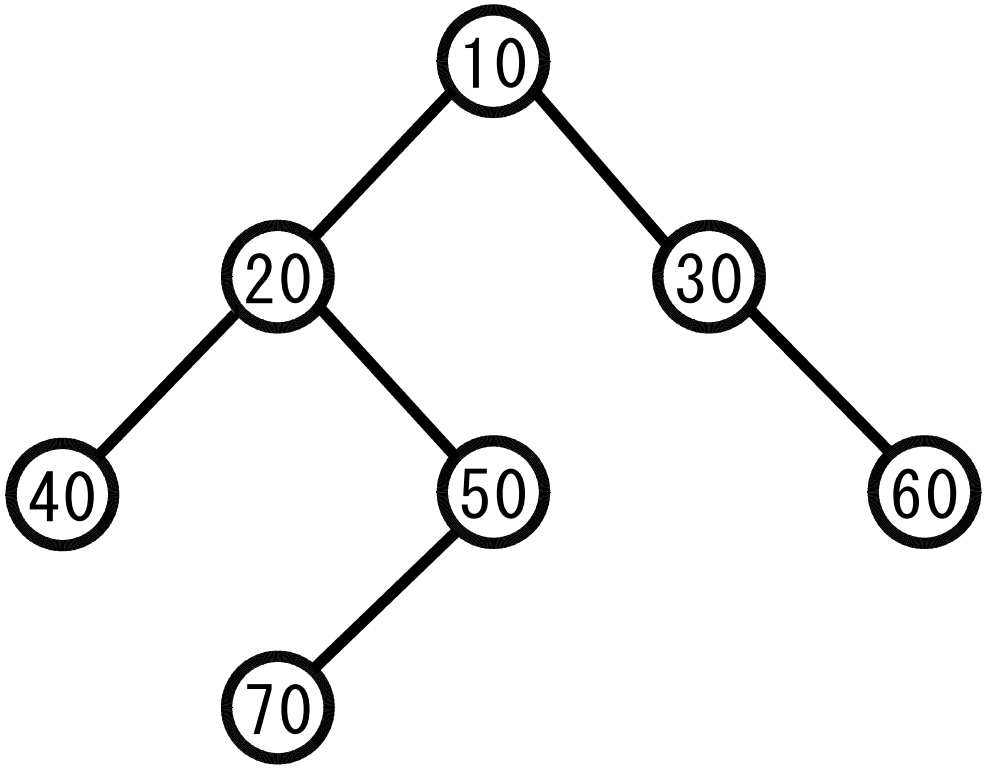
② 任意の根とその部分木について、右部分木に含まれる各要素は根の要素よりも大きい。

上記の条件を満たす２分木はイである。求める答えはイとなる。

アは、１と２、２と４、３と６、６と８が問題である。
ウは、６と５、３と２が問題である。
エは、７と２、８と６、５と４が問題である。

◇問５解答　イ

問６問題



図の二分木において、ある走査を行ったところ、４０、７０、５０、２０、６０、３０、１０の順に節の値が出力された。この走査法はどれか。

ア　幅優先走査
イ　中間順走査
ウ　後行順走査
エ　先行順走査

◇問６解説

二分木の巡回法に関する問題である。

深さ優先順は根から始めて左の子からかつ葉の方向に向かって、左の子から右の子の順に巡回を行う方法である。

巡回の順番を決める方法

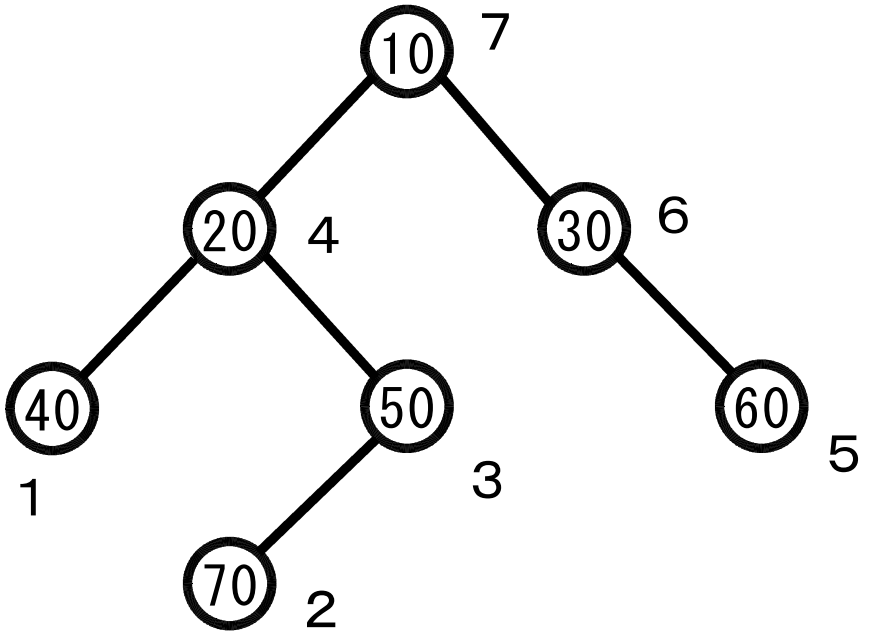
① 先行順は、節点に来たときに先に順位を割り当てる方法である。

② 中間順は、節点に来たときにその左の子を根とする部分木を巡回し終えてから、戻ってきたときに順位を割り当てる方法である。

③ 後行順は、節点に来たときに自分を根とする部分木を巡回し終えてから、順位を割り当てる方法である。

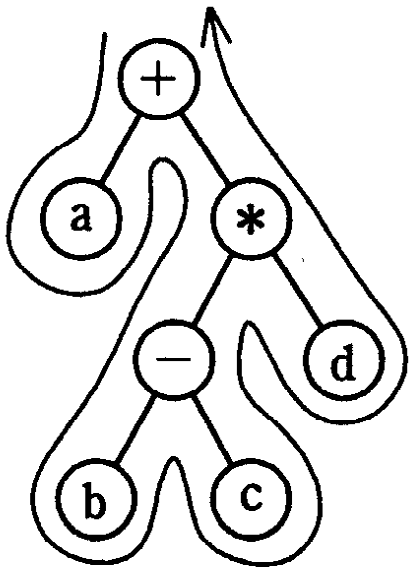
走査の順序を記入すると図の二分木になる。

左右の部分木に番号が付加されると、その元の番号を付加する規則で各ノードの番号が付けられている。
この方式は後行順走査である。求める答えはウとなる。



◇問６解答　ウ

問７問題



前置記法　＋a＊－bcd

中置記法　(a＋((b－c)＊d))

後置記法　abc－d＊＋

四則演算の式の書き方には，演算子をオペランドの前に書く方法（前置記法），オペランドの間に書く方法（中置記法），オペランドの後に書く方法（後置記法）の３通りがある。

図は，２分木で表現された式のたどり方と，各記法によって表される式を例示したものである。

各記法で式を書く手順の説明として，適切なものはどれか。

ア　前置記法：節から上に戻るときにそこの記号を書く。
イ　中置記法：節に下りたときにそこの記号を書く。
ウ　後置記法：節から上に戻るときにそこの記号を書く。
エ　後置記法：葉ならばそこの記号を書いて戻る。演算子ならば下りるときに左括弧を書き，左の枝から右の枝に移るときに記号を書き，上に戻るときに右括弧を書く。

◇問７解説

二分木のならいに関する問題である。

アの説明は後置記法、イは前置記法、ウは後置記法、エは中置記法であり、求める答えはウとなる。

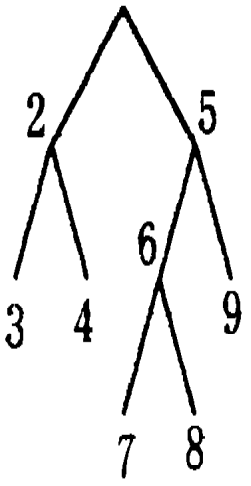
◇問７解答　ウ

問８問題

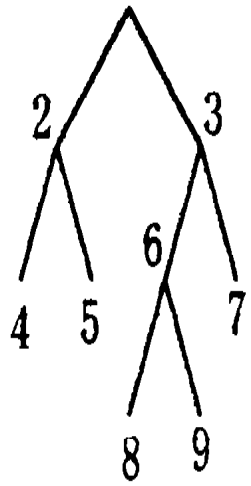
２分木に対する系統的な巡回法のうち、幅優先順を表しているグラフはどれか。

グラフ中の数字は巡回順序を示すものである。

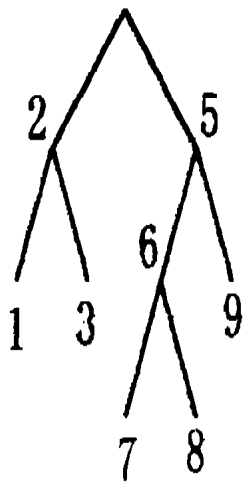
ア



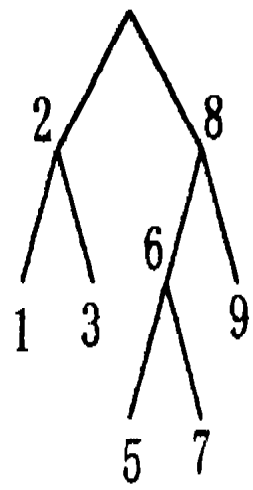
イ



ウ



エ



◇問８解説

幅優先順のならいに関する問題である。

幅優先順は根から始まって、深さの浅い方からかつ左から巡回する方法である。木の幅の方向に対して巡回を繰り返すことになる。

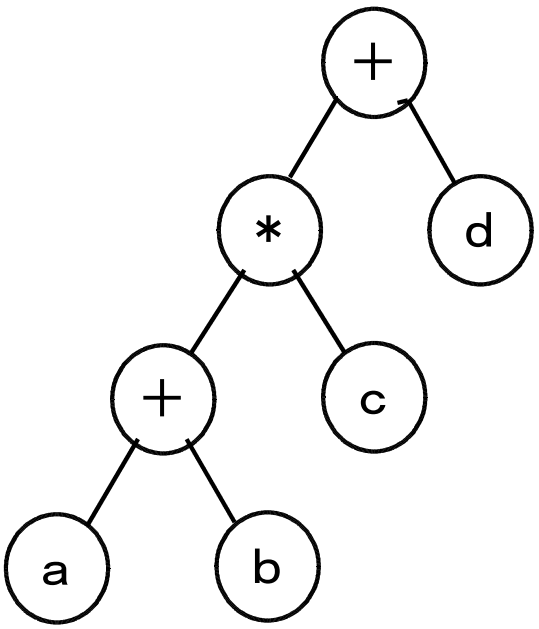
ルート部が１で次のレベルを左から右に向かって２、３となり、その次のレベルを左から４、５、６、７、その次のレベルが８，９となっている２分木が幅優先順である。求める答えはイとなる。

◇問８解答　イ

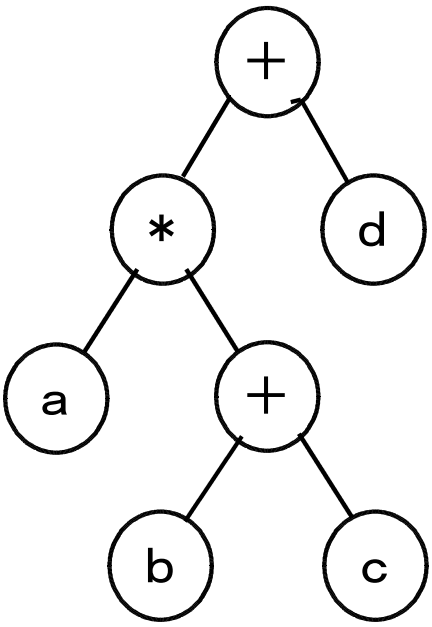
問９問題

ａ＊（ｂ＋ｃ）＋ｄを木構造で表現したものはどれか。

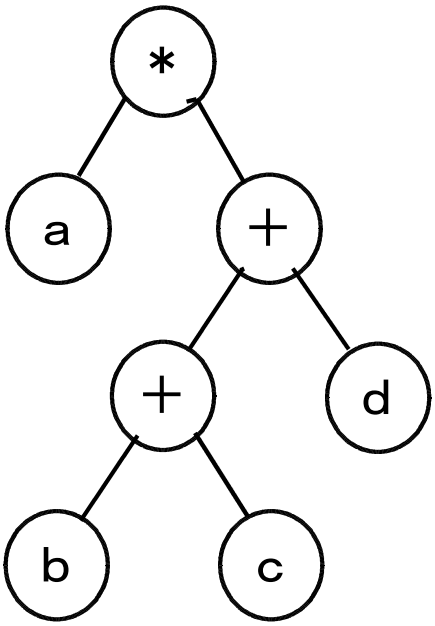
ア



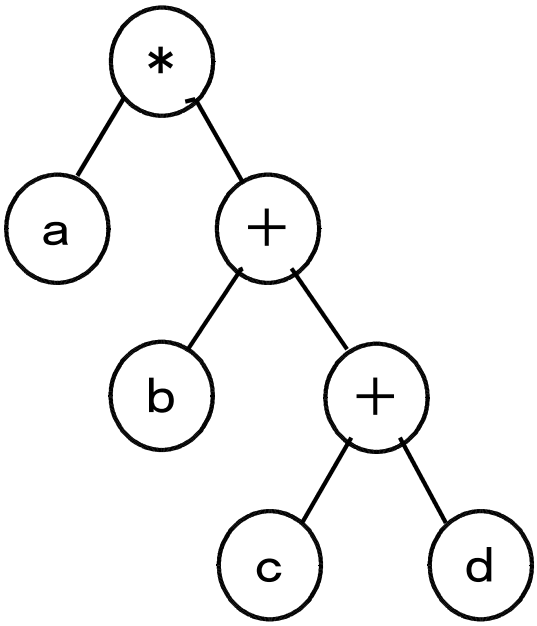
イ



ウ



エ



◇問９解説

二分木を利用した算術式に関する問題である。

二分木の算術式の配置の規則

①　各変数は葉の部分に配置する。

②　演算子は葉以外の節の部分に配置する。

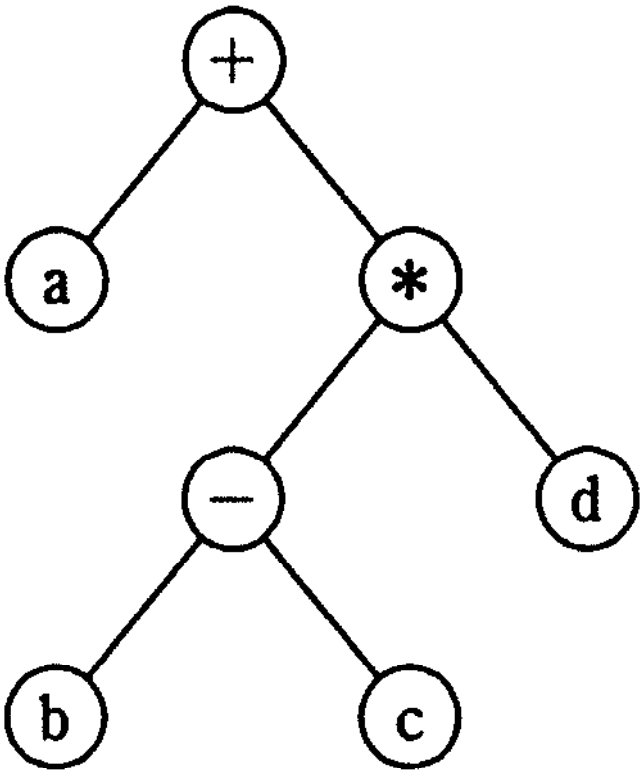
③　演算子の優先順位は下位レベルのものが高い。

演算レベルが最も高いのは、（ｂ＋ｃ）
その次が、ａ＊（ｂ＋ｃ）のａ＊になる。
最後が、＋ｄとなる。

この順序で二分木の下位のレベルから割り当てられているのは、イである。求める答えはイとなる。

◇問９解答　イ

問10問題



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＊＋

◇問10解説

２分木に関する問題である。

プログラムProc()を根から順次適用すると、次の出力が得られる。

ａｂｃ－ｄ＊＋

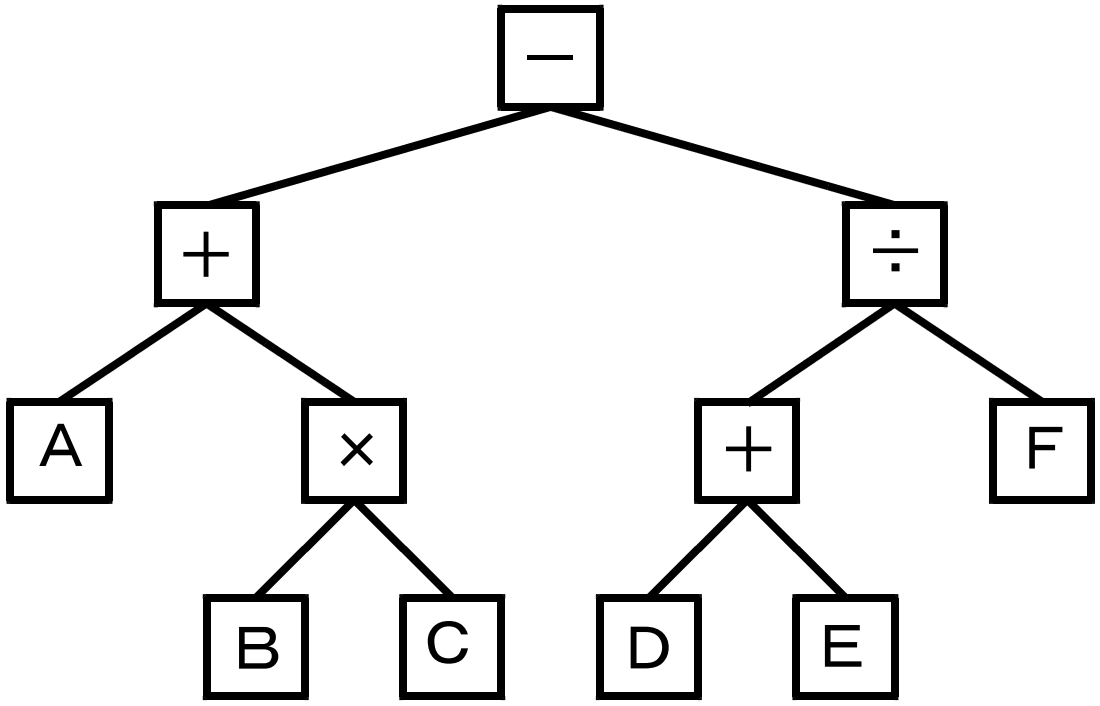
＋の左の子ａ、＋右の子＊の左の子－の左の子ｂ、－の右の子ｃ、－、＊の右の子ｄ、＊、＋の順になる。

求める答はエとなる。

◇問10解答　エ

問11問題

図の２分木で表される算術式はどれか。



ア　Ａ＋Ｂ×Ｃ＋（Ｄ＋Ｅ）÷Ｆ
イ　Ａ＋Ｂ×Ｃ－（Ｄ＋Ｅ）÷Ｆ
ウ　Ａ＋Ｂ×Ｃ－Ｄ＋Ｅ÷Ｆ
エ　Ａ×Ｂ＋Ｃ＋（Ｄ－Ｅ）÷Ｆ

◇問11解説

二分木を利用した算術式に関する問題である。

二分木の算術式の配置の条件

①　各変数は葉の部分に配置されている。

②　演算子は葉以外の節の部分に配置されている。

③　演算子の優先順位は下位レベルのものが高い。

二分木で表される演算式は変数が葉に、演算子が節に配置され、レベルの深いものを優先して演算する。

Ｂ、Ｃのかけ算とＤ、Ｅの加算が最も早く計算される。順次計算すると次の式になる。

Ａ＋Ｂ×Ｃ－（Ｄ＋Ｅ）÷Ｆ

となり、求める答えはイとなる。

◇問11解答　イ

問12問題

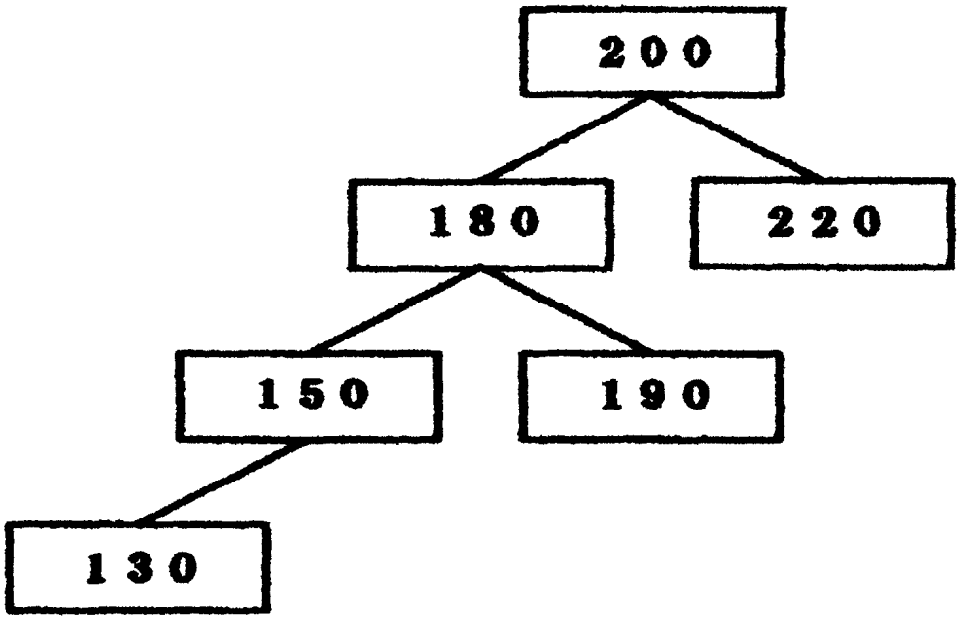


図１　二分木

添字	値	ポイント１	ポイント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

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。

ア　２
イ　３
ウ　４
エ　５

◇問12解説

二分木の配列表現に関する問題である。

次のような内容の配列表現になっている。

①　各節は値部とポイント１、ポイント２で構成されている。

②　各節には添字が付けられている。

③　添字の番号の大小関係は、親の節＜右の子＜左の子の順になっている。

④　各節のポイント１は左の子の添字、ポイント２は右の子の添字を表している。

⑤　子のないポイントは０となる。

求める答えの値部１８０のポイント２は右部分の子を表しているから、値部の１９０が該当する。値部１９０の添字は４であるから、求める答えはウとなる。

◇問12解答　ウ

問13問題

最下位のレベル以外の節点には必ず左右に子が存在する２分探索木から，あるデータを探索する。節点の総数が１５のとき，比較する節点の数は最大で幾つか。

ここで，探索するデータが存在するとは限らないものとする。

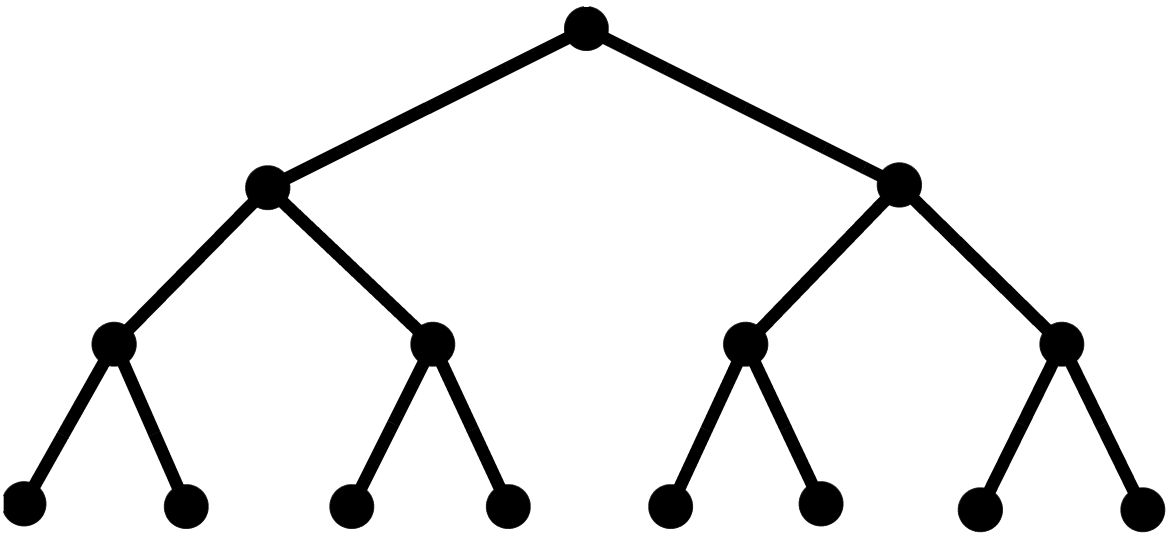
ア　３
イ　４
ウ　７
エ　１５

◇問13解説

二分探索木に関する問題である。

節点には必ず左右に子があり、節点の総数が１５である二分木は図のようになる。

探索は根の部分から実施するため葉に至る比較回数は最大で４となる。求める答えはイとなる。



二分木の節点の総数Ｎは、階層数をＫとすると、次の式で与えられる。

$$N = 2^K - 1$$

Ｎ＝１５の場合、2^K＝１６となり、Ｋ＝４となる。

◇問13解答　イ

問14問題

どの節から見ても、左右の部分木の高さの差が高々１しかない２分木(平衡２分木)において、節が七つの時、この２分木の高さの最低は２になるが、最高はいくつになるか。

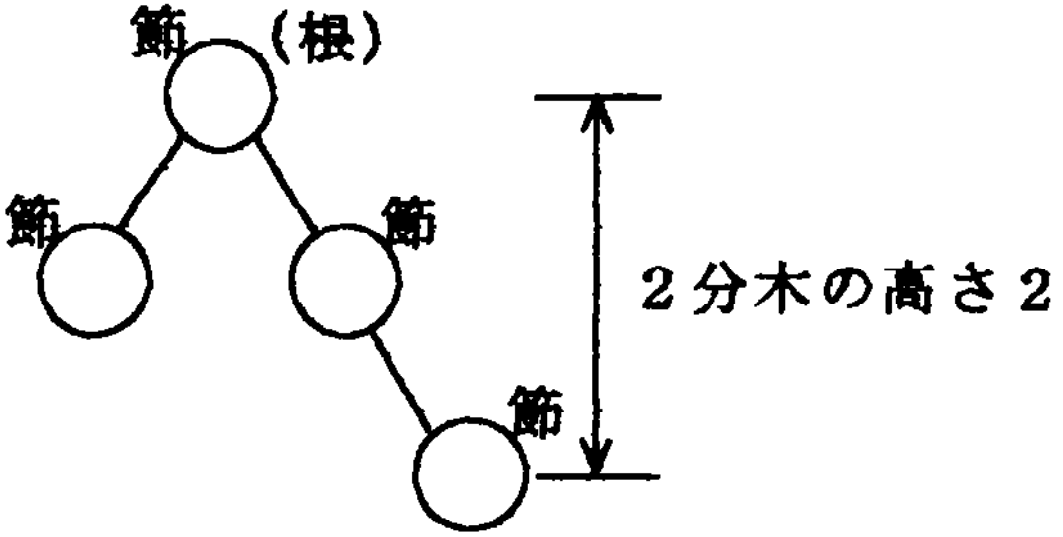


図　平衡２分木の例

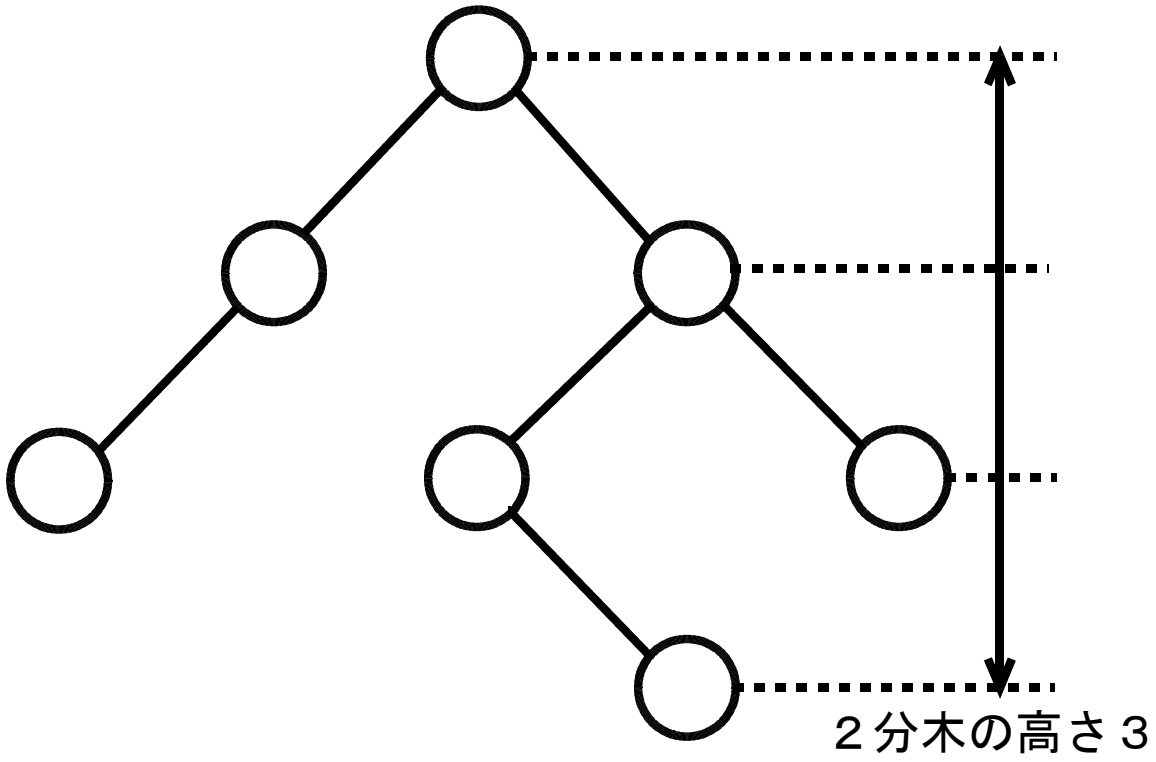
ア　２
イ　３
ウ　４
エ　５

◇問14解説

平衡２分木に関する問題である。

どの節から見ても左右の部分木の差が高々１しか差がないという平衡２分木の定義から考え、節の数が計７個の場合の平衡２分木の例は図のようになる。

図から、２分木の高さの差は３となり、求める答えはイである。



◇問14解答　イ

問15問題

Ｂ木に関する記述のうち、正しいものはどれか。

ア　Ｂ木内の各要素は、追加された順に格納されている。
イ　根からそれぞれの葉までのレベル(深さ)は、要素のキー値に偏りがあると、一定にならない。
ウ　要素の削除に伴って、部分木の要素数が一定値を下回るときは、隣の葉や節を含めて再構成される。
エ　要素は、必ず新しい葉に対して追加される。

◇問15解説

バランス木に関する問題である。

２分木や多分木は要素の挿入・削除を繰り返していくと、木の形が変形しアンバランスな木の形になる。このようなアンバランスな状態を避けるのがバランス木である。

バランス木の特徴

①　要素の挿入や削除を行う毎に木全体の構造を再構成する機能を持っている。

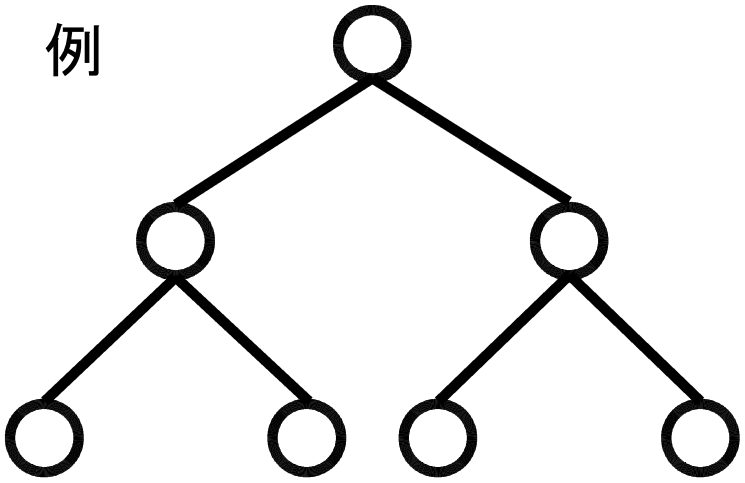
②　バランス木は探索木の一種である。

バランス木は探索の回数を平均化するため、データの追加、削除によりバランスが崩れると、分裂・統合により木のバランスを保つように木全体の構造を再調整する機能を持っている。即ち、要素の削除に伴って、部分木の要素数が一定値を下回ると、隣の葉や節を含めて再構成する。求める答えはウとなる。

◇問15解答　ウ

問16問題

例



ｋ＝３

すべての葉をもつ完全２分木がある。この完全２分木で成り立つ関係式はどれか。

ここで、ｎは節点の個数、ｋ（ｋ≧１）は根から葉までの階層数を表す。例の階層数(ｋ)は３である。

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

◇問16解説

バランス木の節点に関する問題である。

階層と節点の個数の関係

①　階層数によって、節点の数は変化する

②　各階層の節点の数は、値を除いて、最小値と最大値がある。

階層数	節点の個数		
	最小	最大	数式
１	１	１	１＝２^{1}－１
２	２	３	３＝２^{2}－１
３	４	７	７＝２^{3}－１
４	８	１５	１５＝２^{4}－１
…	…	…	
Ｋ－１	２$^{K-2}$	２$^{K-1}$－１	２$^{K-1}$－１
Ｋ	２$^{K-1}$	２^{K}－１	２^{K}－１

階層数をＫとし、節点の個数をＮとすると、次の式が成立する。

$$N = 2^{K} - 1$$

求める答えはウとなる。

◇問16解答　ウ

問17問題

５次のＢ木（各節点から出る枝の数≦５、または各節点に格納されているキーの数≦４）で、レベル２（深さ２でも同じ）までに格納できるキーの数はどれか。

ア　９６
イ　１００
ウ　１２４
エ　１２５

◇問17解説

各節に最大４個のキーを格納し、深さのレベルが２の５次のＢ木の最大キーの数を求める問題である。

５次のＢ木の規則

①　葉を除く各節点が５個までのポインタを持つ。

②　各節点は４個までのキーを持つ。

③　すべての葉は同一レベルに現れる。根から葉までの深さが同じである。

５次のＢ木の特徴

①　メモリの使用効率を上げるため、各節点のポインタの数は、３個以上５個以下である。

②　根を除く要素のキーの数は４個。従って、２個以上４個以下である。

③　根のポインタ数は２個以上、要素のキーの数は１個以上

④　データを追加・削除するたびに、節点の要素数は５個以上、または２個未満になると自動的に木の構造を変更し、葉のレベルが同一になるように調整する。

根の部分に４個、１レベルの部分は節から枝が５個出るから、５×４＝２０、２レベルの部分は更に各節から５個の枝が出るから、５×６×４＝１００、従って、キーの全部の数は

１００＋２０＋４＝１２４個

となる。求める答えはウとなる。

◇問17解答　ウ

問18問題

ＡＶＬ木に関する記述のうち、正しいものはどれか。

ア　任意の節点において左右の部分木の高さが等しい。
イ　任意の節点において左右の部分木の高さの差が１以下である。
ウ　根からすべての葉までの高さが等しい。
エ　根からすべての葉までの高さの差が１以下である。

◇問18解説

ＡＶＬ木に関する問題である。

ＡＶＬ木は左右の部分木の深さの差が１以下の２分木である。完全２分木は根から最も遠い葉までの経路の長さと、根から最も近い葉までの経路の長さの差が１以下である。

アは左右の部分木の高さの差が等しいが正しくない。
イの任意の節点において左右の部分木の高さの差が１以下であるが正しい記述になる。求める答えはイとなる。
ウ、エは完全２分木に関する内容である。

◇問18解答　イ

問19問題

後置記法(逆ポーランド記法)では、例えば、式Ｘ＝(Ａ－Ｂ)×ＣをＸＡＢ－Ｃ×＝と表現する。

次の式を後置記法で表現したものはどれか。

Ｘ＝(Ａ＋Ｂ)×(Ｃ－Ｄ÷Ｅ)

ア　ＸＡＢ＋ＣＤＥ÷－×＝
イ　ＸＡＢ＋Ｃ－ＤＥ÷×＝
ウ　ＸＡＢ＋ＥＤＣ÷－×＝
エ　ＸＢＡ＋ＣＤ－Ｅ÷×＝

◇問19解説

逆ポーランド記法に関する問題である。

四則演算を逆ポーランド記法に変換するには二分木の後行順ならいを利用する。

二分木の葉の部分に変数をＸＡＢＣＤＥの順に並べる。四則演算の演算子の優先度レベルの順位の高いものから順次、次のように演算子で結合する。

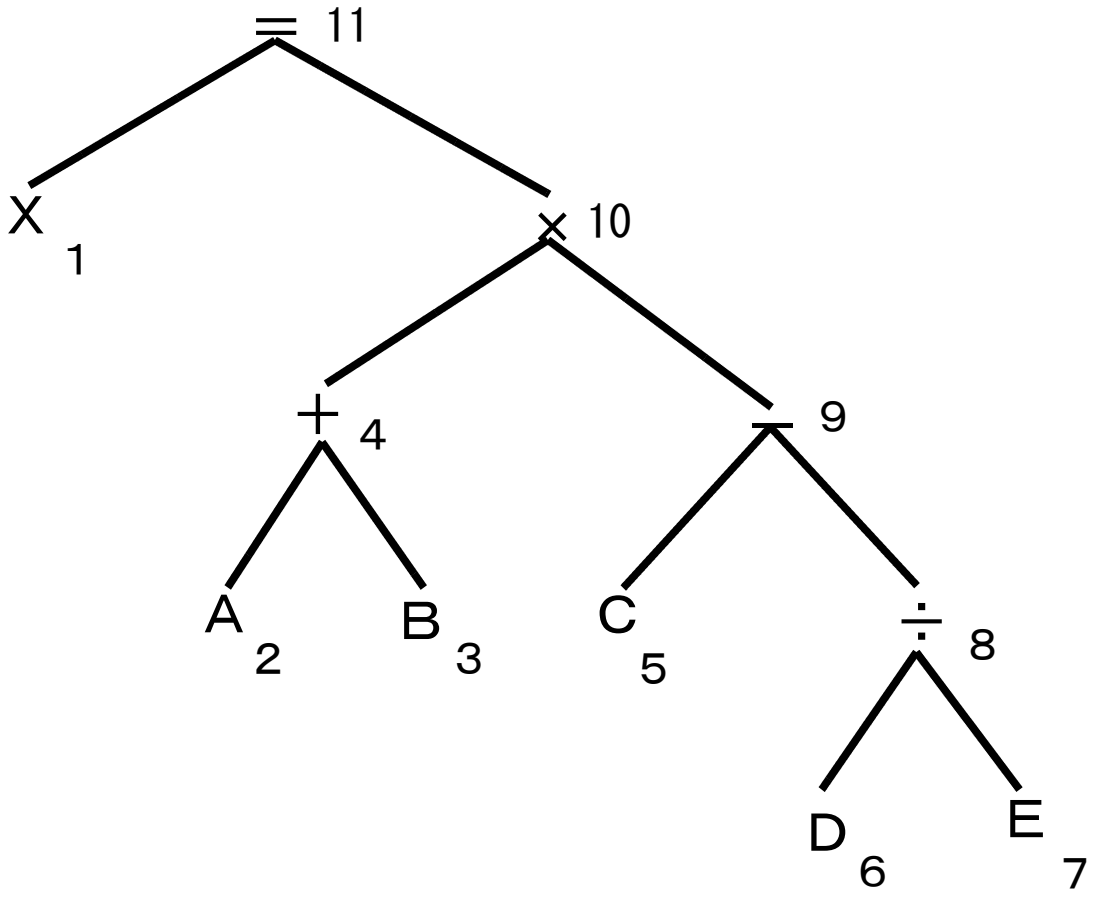
①　ＡＢを＋で結合する。
②　ＤＥを÷で結合する。
③　Ｃと÷を－で結合する。
④　＋と－を×で結合する。
⑤　Ｘと×を＝で結合する。

完成した二分木に後行順ならいを実行すると、図の二分木に示すノードの順番になる。

このノードの順番に変数、演算子を並べると答えが次のように求まる。

Ｘ Ａ Ｂ ＋ Ｃ Ｄ Ｅ ÷ － × ＝

となり、求める答えはアとなる。



◇問19解答　ア

問20問題

逆ポーランド表記法(後置表記法)で,

ＥＦ－Ｇ÷ＣＤ－ＡＢ＋÷＋

と表現される式はどれか。

ア　((Ａ＋Ｂ)＋(Ｃ－Ｄ))÷Ｇ－(Ｅ÷Ｆ)
イ　((Ａ＋Ｂ)÷(Ｃ－Ｄ))＋Ｇ÷(Ｅ－Ｆ)
ウ　((Ｅ－Ｆ)÷Ｇ)＋((Ｃ－Ｄ)÷(Ａ＋Ｂ))
エ　((Ｅ－Ｆ)÷Ｇ)÷((Ｃ－Ｄ)＋(Ａ＋Ｂ))

◇問20解説

逆ポーランド記法(後置表記法)に関する問題である。

次の２つの考え方のいずれかを利用すれば答えを求めることができる。

① ア～エの二分木を作成し、後行順で逆ポーランド記法を求める。

② 逆ポーランド記法を利用して、左から順次演算子の前の２つの変数で式を求める。

ここでは②の考え方で解く。次の手順で実行する。

① ＥＦ－からＥ－Ｆ

② (Ｅ－Ｆ)Ｇ÷から(Ｅ－Ｆ)÷Ｇ

③ ＣＤ－からＣ－Ｄ

④　ＡＢ＋からＡ＋Ｂ

⑤　（Ｃ－Ｄ）（Ａ＋Ｂ）÷から（Ｃ－Ｄ）÷（Ａ＋Ｂ）

⑥　②、⑥の結果を利用して、（（Ｅ－Ｆ）÷Ｇ）（（Ｃ－Ｄ）÷（Ａ＋Ｂ））＋から次の式が求まる。

（（Ｅ－Ｆ）÷Ｇ）＋（（Ｃ－Ｄ）÷（Ａ＋Ｂ））

求める答えはウとなる。

◇問20解答　ウ

問21問題

Ａ＝１，Ｂ＝３，Ｃ＝５，Ｄ＝４，Ｅ＝２のとき、逆ポーランド表記法で表現された式ＡＢ＋ＣＤＥ／－＊の演算結果はどれか。

ア　－１２
イ　２
ウ　１２
エ　１４

◇問21解説

逆ポーランド記法に関する問題である。

ＡＢ＋ＣＤＥ／－＊に値を代入すると、１３＋５４２／－＊となる。

演算は次のようになる。

（１＋３）×（５－４／２）＝４×（５－２）
＝４×３
＝１２

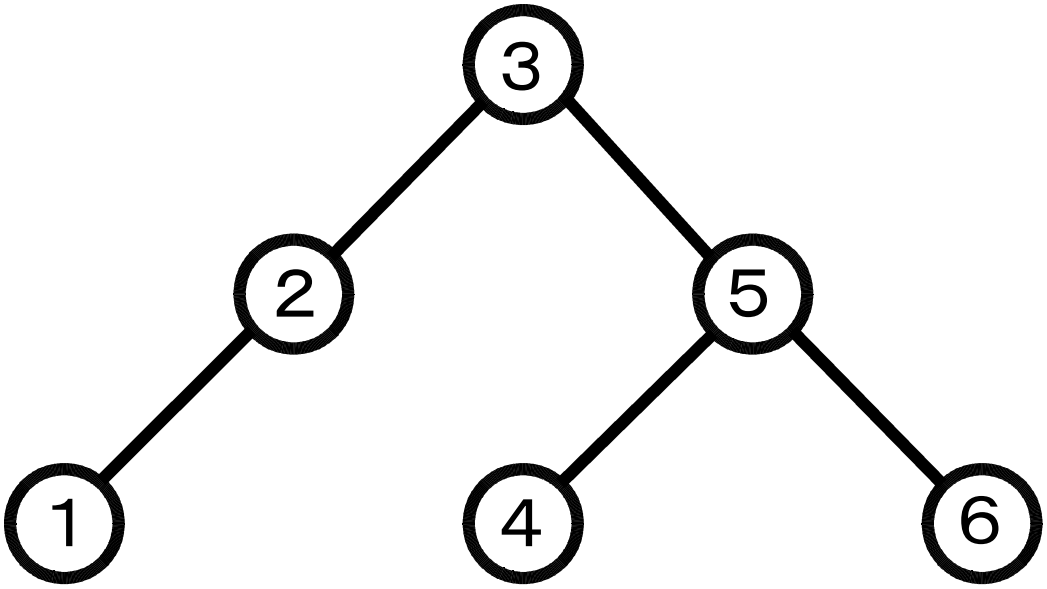
求める答えは１２となる。

◇問21解答　ウ

問22問題

二分木を入力するためのテキスト表現として、（左部分木の節番号またはテキスト表現、節番号、右部分木の節番号またはテキスト表現）と記述する方法を採用した。

部分木が空の時はｘを書く。図のように節に番号を付けたとき、テキスト表現として正しいものはどれか。

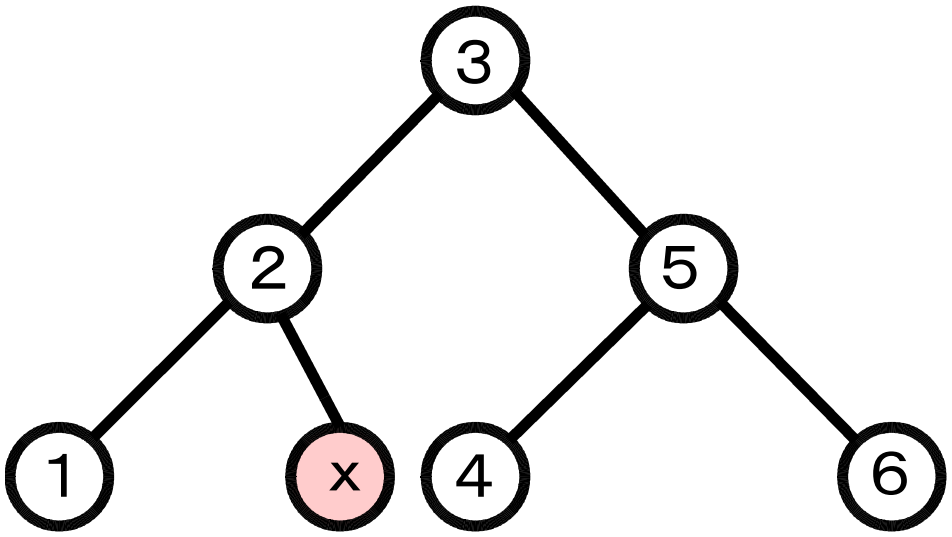


ア　((１、２、３)、(４、５、６))
イ　((１、２)、３、(４、５、６))
ウ　((１、２、３)、ｘ、(４、５、６))
エ　((１、２、ｘ)、３、(４、５、６))

◇問22解説

二分木に関する問題である。

部分木が空の場合をｘとして、二分木を作成すると図のようになる。



図の二分木を

（左部分木の節番号またはテキスト表現、節番号、
右部分木の節番号またはテキスト表現）

で表すと、

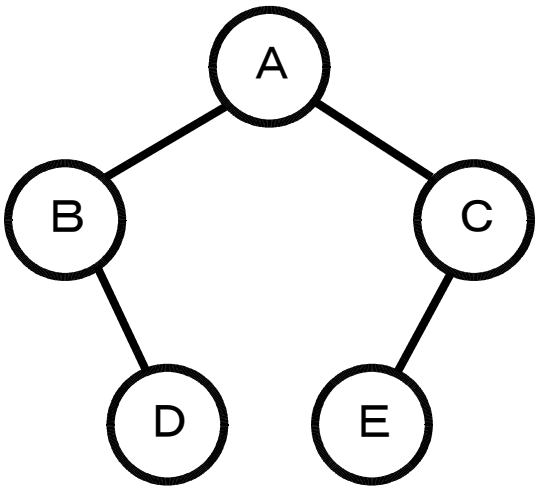
（（１、２、ｘ）、３、（４，５、６））

となる。求める答えはエとなる。

◇問22解答　エ

問23問題

２分木の節に付けられた記号を印字する。印字の順序は各節について、その節の左部分木、その節、その節の右部分木の順である。図の２分木について印字した結果はどれか。



ア　ＡＢＤＣＥ
イ　ＢＤＡＣＥ
ウ　ＢＤＡＥＣ
エ　ＤＢＡＣＥ

◇問23解説

二分木に関する問題である。

印字の順序は、

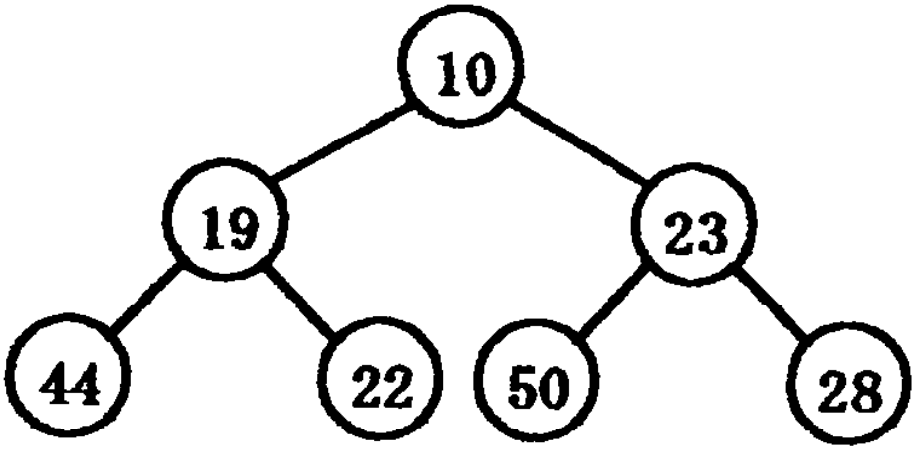
その節の左部分木、その節、その節の右部分木

であるから、ＢＤＡＥＣとなり、求める答えはウとなる。

◇問23解答　ウ

問24問題

図の２分木はヒープである。これを配列で表したものはどれか。



ア	１０	１９	２３	４４	２２	５０	２８
イ	１０	１９	２２	２３	２８	４４	５０
ウ	４４	１９	２２	１０	２３	５０	２８
エ	４４	２２	５０	２８	１９	２３	１０

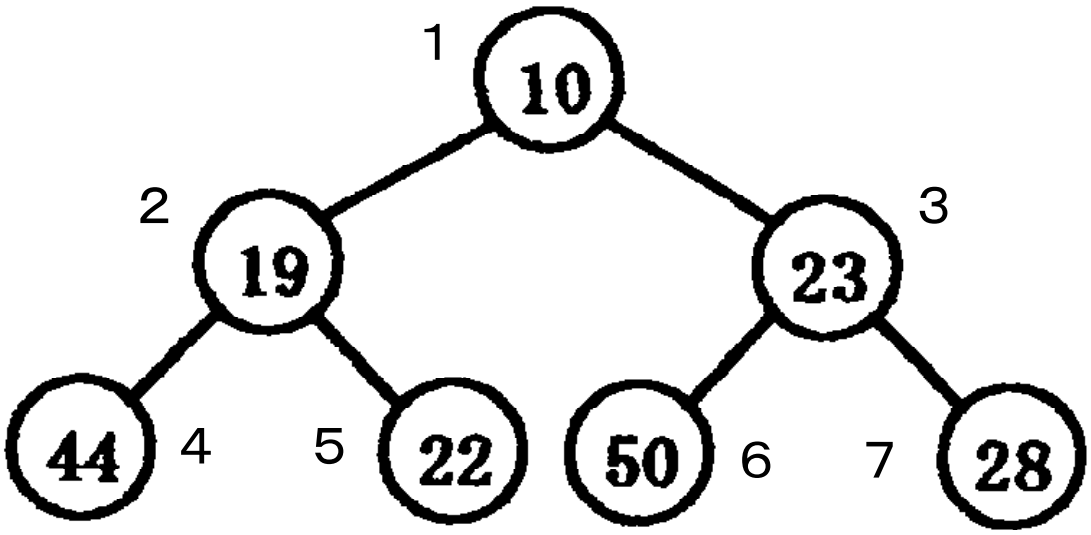
◇問24解説

ヒープ木に関する問題である。

ヒープ木は根の部分を添字の１として、各レベルの左から右に向かって添字の番号を付ける。

下位のレベルの値は上位のレベルの値より大きい(または小さい)の特徴がある。

ヒープ木の節に番号を付けると図のようになる。



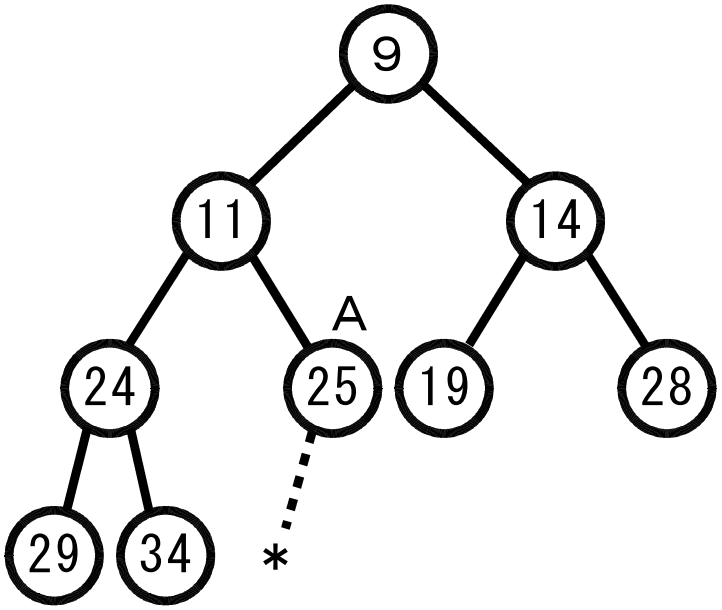
ヒープ木の値を番号順に並べると次のようになる。

１０→１９→２３→４４→２２→５０→２８

求める答えはアである。

◇問24解答　ア

問25問題



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とき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７を追加したとき、Ａの位置にくる要素はどれか。

ア　７
イ　９
ウ　１１
エ　２５

◇問25解説

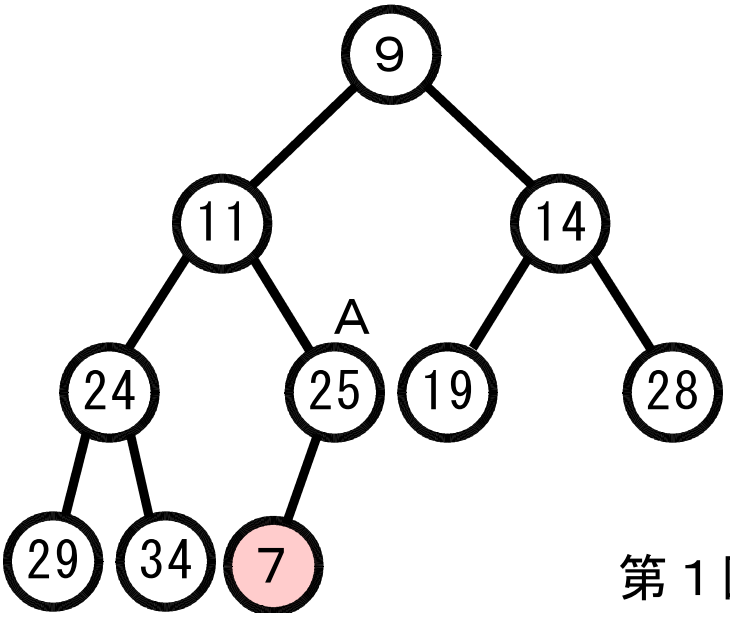
ヒープ木の挿入に関する問題である。

＊の位置に７を追加した図は、第１図の二分木になる。

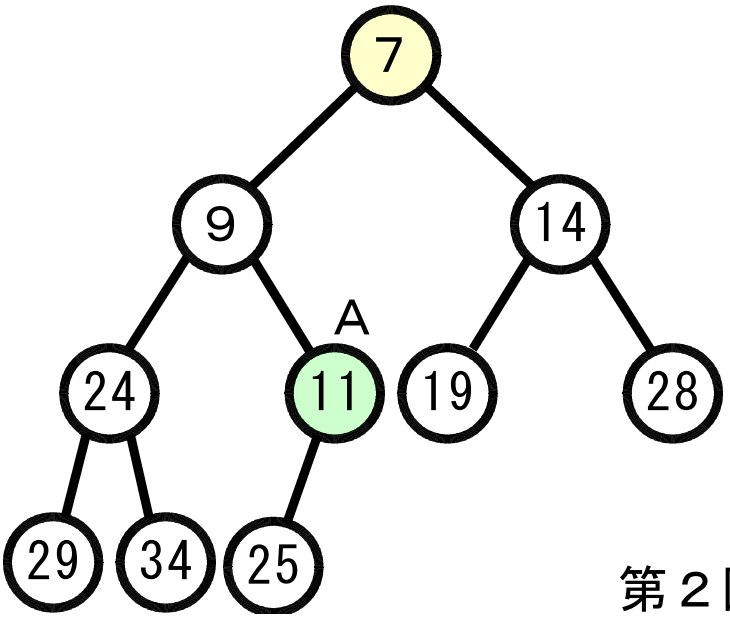
ヒープの規則を用いて、ノードの値を交換すると次のようになる。

①　７と２５を交換する。
②　７と１１を交換する。
③　７と９を交換する

交換後のヒープ木は第２図のようになる。



第１図



第２図

Ａのノードにくる値は１１となる。求める答えはウである。

◇問25解答　ウ