1章

（１）アルゴリズムの定義

与えられた問題の正しい答えを求めるための「うまいやり方」

（２）アルゴリズムを比較するための基準

アルゴリズムの評価基準

（３）3の倍数かどうかの判断

　・コンピュータ：早く実行できる．

　・数学的：N桁の整数を3でわり，あまり0の時，各桁の和が3で割り切れたら．

質の違い：早く実行できるアルゴリズムのほうが優れている（計算効率を意識，性質を用いた思考の工夫）

（４）計算資源の観点からアルゴリズムの性能を比較する際の評価指標を2点

　時間計算量（実行時間の速さ），空間計算量（メモリの大きさ）

（５）時間計算量について，アルゴリズムの性能を比較する際の観点3点

　最良時間計算量，最悪時間計算量，漸近的な時間計算量．

（６）次官計算量の性能の比較の際の評価指標：小さければ小さいほどいい

（７）オーダ記法使われる理由

　入力サイズが大きくなっても実装方法や定数に依存せず，計算量を比較できるから

（８）テニスボール問題不良品見つけるまでの時間の回数：

（９）テニスぼるの数が10から1000になっても最大実行時間が170秒で収まる理由

　1.2は1つずつ調べる方式だが，1.3は1回の操作で半分に減らせるので，捜査の繰り返し回数が1.3はすくないため，実行時間も短くなる．

（１０）A，　B，，　C，

（１１）log＜＜n＜nlogn＜＜＜・・＜＜n!

（１２）入力サイズnが大きくなると実行時間はどんな観点で比較できるか

漸近的な時間計算量（オーダ記法）

（１３）O(n)

（１４）最悪計算時間考え方

　iは1からn-1まででループ回数は(n-1)回

　jはi+1からnまででループ回数は(n-i)

内側ループの回数を全部足すと，.各ループで行う処理はO(1)．よって，O()＝O()．

2章

（１）削除：最悪時間計算量：O(n)

（２）追加：最良時間計算量：O(1)，最悪時間計算量：O(n)

（３）連結リストの考え方

1つのデータをレコードという格納場所で管理し，各レコー3ドをポインタで指すことで，データの列を作っている．

（４）連結リストを実現するためのデータ構造の考え方

各要素をノードとして持ち、ポインタで順につなぐことで構成されるため、配列のように要素を移動させる必要がない。新しいノードを作成し、そのポインタ部を旧先頭ノードに向け、head を新しいノードに更新するだけで追加できる．

（５）連結リストのテータの追加・削除の特徴

連結リスト先頭に追加・削除：**O(1)**．途中や末尾に追加・削除： **O(n)**．したがって、連結リストは「探索は遅いが、先頭での追加や削除は高速に行える」という特徴を持つデータ構造である。

（６）スタックのデータ構造としての特徴．

スタックは処理要求の遅いものからの順序でデータの格納，取り出しをする．一番上からしか出し入れできないため，計算量は定数時間O(1)．

（７）スタックの実装考え方

　格納「push」，取り出し「pop」

（８）キューの特徴

キューは処理要求がはやいものから処理し，要素の追加は末尾から行い、削除は先頭から行う。

（９）キューの実装考え方

格納「enqueue」，取り出し「dequeue」．配列の場合実行時間はどちらもO(1)．

（１０）スタック実装注意点

　配列を用いた空のスタックの準備には，十分なサイズの配列の準備，変数topの値の初期化が必要．

（１１）キューの実装注意点

　・配列を用いた空のキューの準備には，変数left，rightの初期化が必要．

　・変数rightはキューに格納されている右端のデータから一つ右の格納場所を表す．

3章

（１）2分木の特徴

　すべての節点が2個以下の個しか持たない．探索や分類に使いやすい．

（２）完全2分木の葉の数は高さをhとすると，

　完全2分木において，レベル0の節点数は1．レベルが1上がるごとに節点数は2倍になる．よってレベルがkの時節点数は個．ここで，木の高さは，木に含まれる葉のレベルの最大値に1加えたものだから，．計算量は，オーダ記法より，主要項を取り，係数を削除すると)となる．

（３）完全2分木の高さは，葉の数をmとすると，

　葉の数がm個の時，高さをhとすると．よって．計算量はオーダ記法より，主要項を取り，係数を削除するととなる

（４）完全2分木の節点数は，高さをhとすると．オーダ記法より主要項を取り，係数を削除するととなる．

（５）完全2分木の節点数は，高さをhとすると

レベルがkの節点数は．高さをhとすると，節点数は各レベルの節点数の和より，

この式は初項が1，公比2の等比数列の和である．これを変形して．

（６）完全2分木の高さは，節点の数をnとすると，．オーダ記法より主要項を取り，係数を削除すると，.

（７）2分木を配列で表現できる説明

　まず，木の根の節点を1とする．番号iをもつ節点の子を左が2i，右が2i+1とする．次に，iの節点のデータをT[i]に格納する．よってiの節点の子は，T[2i]と，T[2i+1]に格納される．また，iの節点の親データは[i/2]でT[k]に格納される．

（８）n分後の細胞数：10×

（９）n分後の細胞数を求めるアルゴリズム

　nが0の時最初の細胞数は10．n＞0のとき，1分前の細胞数を2倍して求める．

（１０）和の計算を再機木を用いて求めれる説明

　再帰木の高さは繰り返し関数が呼び出される回数を表している．また，再帰を終了するのに必要な時間計算量を表している．よって，再帰アルゴリズムの時間計算量は，再帰木のすべての節点の和に等しくなる．

（１１）和の計算を再機木を用いて求めれる場合の計算量

　n個の和を求めるアルゴリズム全体の時間計算量をT(n)とおく．このとき，定数個の演算と時間計算量がT(n-1)となる関数の再帰的な呼び出しから構成されているため

T(n)= T(n-1)+c(n＞2のとき)，T(n)=c(n=1のとき)

（１２）和の計算を2項の数列に分けて計算する方法の漸化式

　T(n)=2T(n/2)+c(n＞2のとき)，T(n)=c(n=1のとき)

（１３）時間計算量：=c(2n-1)=O(n)

4章

（１）探索の定義：入力としてn個のデータと値xが与えられたときにデータの中からx=diとなるdiを見つける操作．

（２）線形探索，最良時間計算量，最悪時間計算量

最良時間計算量：D[0]に値がある場合でO(1)．最悪時間計算量：Dに値が含まれてない時でO(n)

（３）2分探索のアルゴリズム

　探索範囲の右端と左端を変数left，right．中央の場所を変数mid．While文で2分探索の繰り返しを表し，探索範囲が1になった時(left,rightが等しくなる)繰り返し処理を終了する.

（４）最悪時間計算量：O(logn)

（５）ハッシュ法について，ハッシュ法と線形探索・2分探索との違い

　ハッシュ：平均探索時間がO(1)と非常に高速．

　線形探索・2分探索：平均探索時間がO(n)，O(logn)とハッシュ法に比べて遅い．

（６）ハッシュ法の探索例

　データをハッシュ関数で配列の位置に割り当て，衝突が起きると次の空き場所を探して格納する．配列の最後のH[1.5n-1]で格納できなかったら，H[0]に戻る．

（７）ハッシュ関数に求められる特性

　同じデータが入力されたとき，必ず同じハッシュ値を返す必要がある．データが均等に

（８）15の探索の場合，H[15]には別のデータが格納されているためH[16]が探索され，H[16]が出力され終了する．

（９）データが格納されていた時の回避方法

　オープンアドレス法：衝突が起きたときに，次の空きを探して格納する．

チェイン法：各配列の要素をリストにし，同じハッシュ値のデータをつなげて格納する．

（１０）配列用いたハッシュ法の性質

　探索時間量の平均アルゴリズムは，O().配列のサイズが大きければ大きいほど時間計算量は小さくなる．ハッシュ法の時間計算量は入力データサイズに依存せず，定数時間で実行できる．

（１１）探索の実行時間の比較

　線形探索：データのサイズに比例して増加する．

　2分探索・ハッシュ法は非常に高速．ハッシュ法はデータ数によらずほぼ一定時間で実行できる．

5章

5 章挿入ソート、選択ソート、ヒープソート

(1) アルゴリズムにおけるソートとはどのような操作か

与えられたデータを順に並べるとういう操作

(2) 教科書p.50 アルゴリズム5.1 について、選択ソートの考え方を説明せよ

外側のループで整列済み部分と未整列部分を区別し，内側のループで未整列部分の中の最大値を探す．

(3) 教科書p.50 アルゴリズム5.1 について，選択ソートの最悪時間計算量を求めよ

n-1∑i=1×O(1)×n(n-1)/2

=O(n^2)

(4) 教科書p.51 アルゴリズム5.2 について，挿入ソートの考え方を説明せよ

挿入ソート：すでに整列されている部分列に対して，要素を適切な位置に挿入していくことで全体を順に整列させる

説明：まず，配列の整列とみなす．次の要素D[i]を取り出し，一時変数xに保存する．整列済みの部分の中で，xより大き要素を右にずらす．空いた位置にxを挿入する．これを末尾まで繰り返す．

(5) 教科書p.51 アルゴリズム5.2 について，挿入ソートの最悪時間計算量を求めよ

O(logn)

(6) ヒープソートを構成する2 つの操作を挙げて，説明せよ

ヒープ：大量のデータを特定の順序で記憶するためのデータ構造

2つの操作：プッシュヒープ

　　　　　ヒープHに対してデータxを格納する

　：デリートマキシマム

　　　　　ヒープHから最大の値をもつデータを削除し，取り出したデータを入力する

(7) ヒープソートはどのようなデータ構造の特徴を用いたソート方法であるか説明せよ

→優先順位つきの処理を実現するためのデータ構造

(8) ヒープソートに用いられるヒープに求められるデータ構造としての性質を説明せよ

→2分木の最大レベルをLとすると，０＜k<L-1を満たす各レベルには2^k個の節点が存在し，レベルLに存在する葉は，そのレベルに左詰めにされる．

→各節点に保存されるデータは，その子に保存されるデータより大きい．

(9) 教科書p.56 図5.7 について，ヒープへのデータの追加の手順を確認せよ

→末尾に新しいデータを追加する（完全二分木の形を保つため，空いている一番下の右側に入れる）

→親ノードと比較する（最大ヒープなら「親より大きいか，最小ヒープなら「親より小さいか」）

→条件を満たさない場合は親と交換する（ヒープ条件が崩れていれば，親子を入れ替える）

→根に到達するか，条件を満たすまで繰り返す

(10) 教科書p.56 図5.8 について，ヒープへのデータの追加の手順を確認せよ

(11) ヒープソートにおける、ヒープから最大値の取り出し時の手順について説明せよ。

(12) 教科書p.57 図5.9 について、ヒープから最大値の取り出し時の手順について説明せよ。

(13) 教科書p.58 図5.10 について、ヒープを表す配列に対するデータの取り出しの手順について確認せよ。

(14) 教科書p.59 アルゴリズム5.5 について、ヒープソートの手順の構成を説明せよ。

(15) ヒープソートの最悪時間計算量の求め方の考え方を説明せよ。