アルゴリズム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）配列からデータを削除：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

テキスト

AI 生成コンテンツは誤りを含む可能性があります。　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．

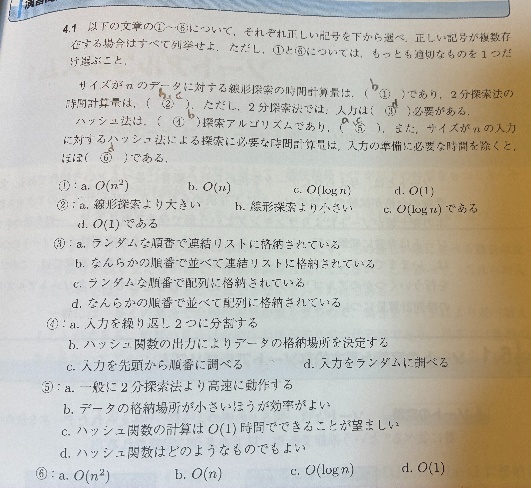
（７）ハッシュ関数に求められる特性

　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

（１）ソートはどのような操作か

　与えられたデータを決められた順に並べる操作

（２）選択ソート考え方

　まず，最大のデータを見つけ，それをソートの対象から除外する．これらの操作をn-1回繰り返し，除外した最大データを順番に並べると，アルゴリズムの終了時のデータはソートされたデータになっている．

（３）選択ソート，最悪時間計算量：最良時間計算量と等しくO().

（４）挿入ソート考え方

　手札を並び替える動作と一緒．左から順に整列済みの部分列を作り，新しい要素をその中の適切な位置に挿入する．

（５）挿入ソート，最悪時間計算量：O()．最良はO(n)．

（６）ヒープソートを構成する2つの操作

・push\_heap(H,x)：ヒープHに対してデータxを格納する．

・delete\_maximum(H)：ヒープHから最大の値を持つデータを削除し，取り出したデータを出力する．

（７）ヒープソートはどのようなデータ構造を用いたソート方法か

　値の大きいデータを優先順位が高いものとして，先に取り出せるようにしている．

（８）ヒープに求められるデータ構造としての性質

　ヒープは以下の二つの性質が成り立つ2分木のこと

・2分木の最大レベルをImとすると0＜k＜Im-1を満たす各レベルkには個の節点が存在し，Imに存在する葉は左詰めされている．-1

・各節点に保存されるデータは，子に保存されるデータよりも大きい時．-2

（９）ヒップのデータの追加の手順

テキスト

AI 生成コンテンツは誤りを含む可能性があります。①データxを格納する節点を作成し1を満たすような葉として追加．

②・親節点のデータが大きければ2を満たしているため終了する

　・親節点のデータが小さければ，親子の節点間のデータを交換し皮革の操作を繰り返す

（１０）ヒップのデータの追加の手順

　末尾に追加→親と比較→大きければ上にあげる→条件を満たすまで繰り返す．

（１１）ヒープから最大値取り出し時の手順

　根から最大値を取り出し，右端の葉のデータを格納．移動したデータの節点と，その子の節点の比較．子節点が存在しなければ終了．子節点が1つの場合は，このデータが小さい場合は終了．大きい場合はデータを交換し比較を繰り返す．子節点が2つの場合，両方とも小さければ終了．片方が大きかったら，大きいほうのデータと交換し比較を続ける．

（１２）図のヒープから最大値取り出し時の手順

　まず，39というデータを根から取り出し，右端の葉の5というデータが格納される．このとき節点に保存されるデータがこのデータの値より小さいため，それぞれ比較をする．この図の節点では24が一番大きいので，5とデータを交換する．交換後の節点5の子は1と2でそれぞれ親の値が大きくなっているので操作はここで終了．

（１３）ヒープを表す配列に対するデータの取り出し手順

　まず，根（最大値）を取り出し，配列の末尾のデータを先頭に移す．子ノードの23.24のうち大きいほうを選び5と24を交換する．

（１４）ヒープソートの手順の構成

　配列に格納されたn個のデータについて，push\_heapをn回繰り返し，ヒープを表す2分木をつくる．これに対し，delete\_maximumをn回繰り返し，データを取り出した順に並べる．

（１５）ヒープソート最悪時間計算量の考え方

　最良，最悪ともにO(nlogn)．N個のデータが格納されたヒープに対する時間計算量はO(logn)だが，データサイズが1からnまでの場合は，

6章

（１）クイックソート，ソートの考え方．

　まず，分割の基準となる値（基準値）を入力データから適当に選び，次に，入力データを基準値よりも大きいか小さいかで2分割する，このとき基準値よりも小さいデータは基準値よりも左に，大きいデータは右に置く．その後また同じ手順を繰り返して分割した，集合のデータが1つになるまで繰り返すとすべてのデータがソートされる．

（２）図のソートの考え方

　基準値を11と設定し，それぞれ基準値よりも大きいほうと小さいほうに分割する．また，分割したそれぞれの集合にも基準値を定め，同じように分割する．分割した集合のデータが1つになるまで繰り返す．

（３）クイックソートの再帰木のソートの考え方：(2)といっしょ

（５）クイックソートの手順が6.1の流れになる理由

　Partitionで1回並び替えの区切りを作り，そのあと大きいほう小さいほうをそれぞれ独立して整列させる．この処理を再帰的に繰り返すことで，最終的に全体が整列されるから．

（６）partitionに求められる処理：基準値を決めて配列を大きい順，小さい順に分けること．

（７）partitionの実行例：基準値11とし，右端の6と交換する．とiとjの位置を決めそれらを交換する．この操作をi＞jとなるまで繰り返し最後にiと右端のデータを交換する．

（８）分割操作の考え方

　基準値を中心に小さい値を左に，大きい値を右に整理して基準値を正しい位置に置く処理．

（９）クイックソートの最悪時間計算量がO(n^2)になるときの入力例：基準値を先頭や末尾から選ぶとき

（１０）クイックソート，最悪時間計算量求め方考え方

　再帰木の高さがn，節点の時間計算量がcn，c(n-1),・・cのとき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が表す計算量に等しいので，その和を求めると，

（１１）クイックソートの最悪時間計算量：

（１２）クイックソートが最良時間計算量で動作するときの再帰木の構成

　再帰木の各レベルの節点に含まれる時間計算量の和が等しい時．

（１３）クイックソート，最良時間計算量：O(logn×cn)=O(nlogn)

（１４）ソートアルゴリズム性能比較

　選択ソートと，挿入ソートアルゴリズムは，時間計算量のに比例する実行時間になっていて，時間計算量がO(nlogn)のソートアルゴリズムがすべて1秒以下で実行できることと比較すると実行にとても時間がかかるとわかる．また，同じO(nlogn)という時間計算量を持つ3つのソートアルゴリズムの中で，クイックソートがもっとも高速．

（１５）ソートに求められる安定性

　・与えられたデータを決められた順番に並べる．・同じ値のデータは，入力の順番通りに並べる．この二つの条件を満たすこと．

章末問題

1-1　①アルゴリズムの時間計算量：アルゴリズムの評価基準，入力サイズの関数を用いて表される．

2-1　①配列：格納するデータサイズをあらかじめ決める．任意の場所に対してO(1)で読み書きできる．

　　②連結リスト：データサイズの変更できる．先頭のデータ削除はO(1)．1つのデータをレコードで管理．

　　③スタック：LIFOの順序．処理要求の順番が遅いものから．

　　④キュー：処理要求の速いものから．FIFOの順序・

　　⑤push，pop計算時間用：O(1)

　　⑥enqueueとdequeueの時間計算：O(1)

2-2　スタックを表す配列sを初期化：top=-1，キューの初期化：left=0，right=0．

2-3一回目：8，2回目：3，3回目：1．Popは一番右を出力

2-4一回目：4，2回目：3，3回目：8．Dequeueは一番左を出力

3-2　(1)完全3分木，高さhの時の葉の数と節点数

　レベルがhの時の節点数は．葉の数はレベルがh-1の時だから，．すべての節点数は

　　(2)節点数がnのときの木の高さ：O(logn)

3-3A：O(n)，B：O(logn)

5-3　delete\_maximumは最大値を出力する

5-4　(1)T[4]＜T[8]で親≧子のヒープ条件を満たしてないからヒープを表す配列でない

　　(2)T[i]＞T[j]よりT[i]の親T[]は大きくなる．よって降順のデータはヒープ．