分割統治法は、(①:c)というアルゴリズムであるが、(②:b) 。マージソートは(③ :b)というソートアルゴリズムであるが、n 個のデータに対しては、(④:a) 。

①:**a.** すべての解を効率良く列挙する
b. アルゴリズムの実行途中において全体的なことは考えず、局所的に最良の解を選択する
c. 入力をいくつかの部分問題に分割し、各部分問題を再帰的に解く
d. 問題を部分問題から解き、その解を記録しておいて再利用する

②:**a.** 再帰とともに用いられることはない
b. 分割、統治、組合せという3つのステップで構成される
c. 入力は必ず2つの部分問題に分割される
d. 時間計算量は、入力サイズ n とすると必ず O(n \log n) となる

③:**a.** 配列の左からデータを順番に処理する
b. データをほぼ同じサイズの2つの集合に再帰的に分割する
c. 入力を基準値を用いて2つの集合に分割し、再帰的にソートを実行する
d. 配列以外のデータ構造を利用する

④:**a.** 時間計算量が常に O(n \log n) である
b. 最悪時間計算量はクイックソートの最悪時間計算量と同じである
c. つねに挿入ソートより高速に実行できる
d. 入力により計算時間が異なる。

情報科学の分野で用いられるグラフは(①:b)であり、日常では(②:c,d)などとして用いられている。また、このグラフは、(③:d)。グラフを格納するための代表的なデータ構造としては、隣接行列と隣接リストがあるが、(辺の数)が少ないグラフを格納する場合は、記憶領域の面では(④:b)である。また、2つの頂点間に辺があるかないかを頻繁に調べる場合は、(⑤:a)である。

① a. y = x^2 などの関数の値を2次元平面上にプロットしたもの
b. データの関係を視覚的に表す抽象概念
c. 全順序関係をもつデータを特定の順序で記憶するためのデータ構造
d. O(1) 時間で探索を実行するためのデータ構造

② a. 電車の時刻表
b. データ分布を表す円グラフ
c. 高速道路の路線図
d. ネットワークの配線図

③ a. 頂点の集合と2つの頂点を結ぶ辺の集合で構成される
b. 2頂点間に必ず辺が存在する
c. 必ず根とよばれる頂点がある
d. 辺の数が m の場合、頂点数は m(m-1)/2 以下である

④ a. 隣接行列のほうが有利
b. 隣接リストのほうが有利
c. 隣接行列でも隣接リストでも同じ
d. 隣接行列や隣接リストでは不十分

⑤ a. 隣接行列のほうが有利
b. 隣接リストのほうが有利
c. 隣接行列でも隣接リストでも同じ
d. 隣接行列や隣接リストでは不十分

グリーディ法は (①:**b**) というアルゴリズムであるが (②:**b**) 手法である。動的計画法は (③:**d**) というアルゴリズムであるが (④:**d**)。

①
- **a.** すべての解を効率良く列挙する
- **b.** アルゴリズムの実行途中において全体的なことは考えず、局所的に最良の解を選択する
- **c.** 入力をいくつかの部分問題に分割し、各部分問題を再帰的に解く
- **d.** 問題を部分問題から解き、その解を記録しておいて再利用する

②
- **a.** つねに最適な解が得られる
- **b.** つねに最適な解が得られる問題もあるが、そうでない問題もある
- **c.** 最適な解は得られない
- **d.** どんな問題にも適用できる

③
- **a.** すべての解を効率良く列挙する
- **b.** アルゴリズムの実行途中において全体的なことは考えず、局所的に最良の解を選択する
- **c.** 入力をいくつかの部分問題に分割し、各部分問題を再帰的に解く
- **d.** 問題を部分問題から解き、その解を記録しておいて再利用する

④
- **a.** 一度計算した値はすぐに削除する
- **b.** 0-1 ナップサック問題に対してつねに正しい解が得られるとは限らない
- **c.** どんな問題にも適用できる手法である
- **d.** 処理の高速化のための手法である

バックトラック法は(① :a)というアルゴリズムである。また、分枝限定法は、バックトラック法に(②:**b**)という性質を追加した方法である。この分枝限定法では(③ : a,c) 。

①:
- **a.** すべての解を効率良く列挙する
- **b.** アルゴリズムの実行途中において全体的なことは考えず、局所的に最良の解を選択する
- **c.** 入力をいくつかの部分問題に分割し、各部分問題を再帰的に解く
- **ddd.** 問題を部分問題から解き、その解を記録しておいて再利用する

②:
- **a.** 解の列挙をさらに増やす
- **b.** 不必要な解の列挙を省略する
- **c.** 入力そのものを限定する
- **d.** 近似的な解を求める

③:
- **a.** 入力によりアルゴリズムの実行時間は大きく異なる
- **b.** どのような入力に対してもアルゴリズムの実行時間は同じである
- **c.** 枝刈りの条件の決め方により実行時間が変化する
- **d.** 暫定解を記録しなくても実行時間は変わらない

- 多項式の値を求めるホーナーの方法は、n+1 個の係数をもつ多項式に対して、時間計算量が(①:**b**)である 。
- 2つの n × n 行列の積を求める基本的なアルゴリズムの時間計算量は(②:**a**)であるが、ストラッセンの行列積アルゴリズムを用いることにより、(③:**c**)で行列積を求めることができる 。
- また、n個の行列の最適な連続積を求める動的計画法を用いたアルゴリズムは、時間計算量が(④:**a**)である 。

① の選択肢
- a. O(n2)
- b. O(n)
- c. O(logn)
- d. O(1)

② の選択肢
- a. O(n3)
- b. O(n2)
- c. O(nlog27)
- d. O(n)

③ の選択肢
- a. O(n3)
- b. O(n2)
- c. O(nlog27)
- d. O(n)

④ の選択肢
- a. O(n3)
- b. O(n2)
- c. O(nlog27)
- d. O(n)

文字列照合を行う基本的なアルゴリズムは、(①:**a,d**)するが、長さ n のテキストと長さ m のパターンを入力とする場合、パターンがテキスト中に存在しなければ、その最良時間計算量は (②:**d**) であり、最悪時間計算量は (③:**a**) である。
ボイヤー・ムーア法とよばれる文字列照合アルゴリズムは、1つ目のアイデアにより、(④**b,c**) するが、これにより不一致が起こった場合にパターンを最大 (⑤: a) 文字分だけ右にずらしてつぎの比較を行うことができる。また、2つ目のアイデアも同時に用いて、テキスト中の文字の種類数を定数と考えると、パターンがテキスト中に存在しない場合のボイヤー・ムーア法の最良時間計算量は (⑥ :e) であり、最悪時間計算量は (⑦:**d**) である。

①
- a. テキストとパターンをパターンの左端から比較
- b. テキストとパターンをパターンの右端から比較
- c. テキストの右端から比較を開始
- d. テキストの左端から比較を開始

②・③・⑥・⑦
- a. O(mn)　d. O(n)
- b. O(n+m)　e. O(n/m)
- c. O(m)　f. O(1)

④
- a. テキストとパターンをパターンの左端から比較
- b. テキストとパターンをパターンの右端から比較
- c. 不一致が起こったテキスト中の文字の情報を利用
- d. 不一致が起こったパターン中の文字の情報を利用

⑤
- a. m　b. n　c. n/m　d. mn

問題のクラスにおいて、クラス P は (① :a,d) 問題のクラスであり、クラス NP は (② :b) 問題のクラスである 。
ある問題 A が問題 B に O(n) 時間で帰着でき、問題 B を O(n^2) 時間で解くアルゴリズムが存在するとき、(③ :a) 。
問題 A が (④:b)、(⑤:d) の 2 つの条件を満たすとき、その問題を NP 完全とよぶ。この NP 完全問題は (⑥a,b,c) 。

①:
- **a.** 入力サイズの多項式に比例する時間で解ける
- **b.** 入力サイズの指数に比例する時間で解ける
- **c.** クラス NP を含む
- **d.** クラス NP に含まれる

②:
- **a.** 入力サイズの多項式に比例する時間で解ける
- **b.** 入力サイズの多項式に比例する時間で解が正しいかどうかを確かめることができる
- **c.** クラス P を含む
- **d.** クラス P に含まれる

③:
- **a.** 問題 A は O(n^2) 時間で解ける
- **b.** 問題 A は O(n) 時間で解ける
- **c.** 問題 B は O(n) 時間で解ける
- **d.** 問題 A と問題 B の難しさは同じである

④:
- **a.** 問題 A はクラス P に属する
- **b.** 問題 A はクラス NP に属する
- **c.** 問題 A はクラス P に属さない
- **d.** 問題 A はクラス NP に属さない

⑤:
- **a.** 問題 A はクラス NP に属する任意の問題に多項式時間で帰着可能である
- **b.** 問題 A はクラス P に属する任意の問題に多項式時間で帰着可能である
- **c.** クラス P に属する任意の問題を多項式時間で問題 A に帰着可能である
- **d.** クラス NP に属する任意の問題を多項式時間で問題 A に帰着可能である

⑥:
- **a.** つねに NP 困難問題でもある
- **b.** クラス NP に必ず属する
- **c.** クラス NP に属する問題の中でもっとも難しい問題だと考えられている
- **d.** 入力サイズの多項式時間で計算でき

1. 分割統治法とは、問題をいくつかの部分問題に分割して解いた後、その解を再構成して全体の解を得るという手法であり、分割、統治、組合せという3つのステップで構成される。
2. 分割統治法における分割のステップでは、各部分問題は再帰的に解かれることが多いので、分割統治法のアルゴリズムは再帰を用いることが一般的である。
3. n 桁の整数どうしの掛け算を基本的な方法で計算すると、$O(n^2)$ 時間が必要であるが、分割統治法を用いると、$O(n^{\log_2 3}) = O(n^{1.59})$ 時間で計算することができる。
4. 分割統治法の代表例であるマージソートは、入力のデータを2つに分割し再帰的にソートを行うソートアルゴリズムである。マージソートでは、サイズが $\frac{n}{2}$ の2つのソート済みの列を $O(n)$ 時間でマージする処理を関数 merge により実現することにより、n 個のデータに対して **$O(n \log n)$** 時間でソートを実行することができる。

---

1. グリーディ法（貪欲法）は、アルゴリズムの実行の各ステップにおいて、将来のことを考えずに、その時点で最良と思われる解を選択する手法である。
2. グリーディ法はつねに最適な解を導くとは限らない。例えば、**0-1** ナップサック問題に対しては、グリーディ法では最適な解が得られない場合がある。
3. 荷物がn個の分割ナップサック問題に対してはグリーディー法によりO(nlogn)時間で正しい解を求めることができる
4. 動的計画法（Dynamic Programming; DP）は、問題をいくつかの部分問題に分割し、各部分問題の解を記録（メモ）しておき、それらを再利用することで全体の解を求める手法である。
5. 動的計画法を用いると、0-1 ナップサック問題に対して最適な解を確実に求めることができる。重さの上限を W、荷物の個数を n とすると、動的計画法による時間計算量は **$O(nW)$** となる。

---

1. バックトラック法とは、解の列挙が必要な問題に対して効率良く系統的に解を列挙しチェックを行う方法である。このバックトラック法における解の列挙は、列挙木とよばれる木を用いて表される。
2. 入力サイズが n の部分和問題に対するバックトラック法を用いたアルゴリズムの時間計算量は、**$O(n2^n)$** である。
3. 分枝限定法はバックトラック法に加えて用いられる手法であり、枝刈りという操作により不必要な列挙の操作を行わないようにする手法である。ここで枝刈りとは、バックトラック法により作成される列挙木の各節点が表す選択肢において、暫定解などからその選択により問題の出力となる解が得られないと判定される場合に、それ以上の列挙の操作を中止し、後戻りする操作である。
4. 部分和問題や 0-1 ナップサック問題に対する分枝限定法を用いたアルゴリズムの最悪時間計算量は、漸近的には分枝限定法を用いないバックトラック法と同じである。しかし、分枝限定法を用いたアルゴリズムの実際の実行時間は、単なるバックトラック法のアルゴリズムと比較して、非常に高速である。

---

1. グラフとは、データの関係を視覚的に表す抽象概念であり、頂点および辺の集合で構成される 。また、グラフに含まれる2つの頂点は、一般に頂点の列により結ばれており、この頂点の列のことを頂点間の経路とよぶ 。
2. アルゴリズム中でグラフを格納する方法としては、隣接行列と隣接リストという2つのデータ構造が一般的に使われている 。隣接行列は2次元配列を用いて各辺の情報を格納するデータ構造であり、辺が多いグラフを格納する場合に向いている 。隣接リストは、各頂点に対して、その頂点に隣接する頂点の情報を連結リストを用いて格納するデータ構造であり、辺の数が少ないグラフを格納する場合に向いている 。
3. グラフの探索とは、始点となる頂点を指定し、始点からすべての頂点を1回ずつ調査する操作である 。このグラフの探索については、幅優先探索と深さ優先探索という2つの方法が知られており、頂点数が n、辺の数が m の場合、どちらの探索方法も、グラフが隣接行列で表される場合は **$O(n^{2})$**、隣接リストで表される場合は **$O(n+m)$** という時間計算量で実行できる 。
4. 最短経路問題とは、与えられたグラフとグラフ中の2つの頂点間の最短の経路を求める問題である 。この最短経路問題に対しては、ダイクストラ法という効率の良いアルゴリズムが知られており、このアルゴリズムを用いることにより、頂点数 n のグラフに対して **$O(n^{2})$** 時間で最短経路を求めることができる 。

---

1. 次の多項式の計算を基本的なアルゴリズムで行うと、$O(n^2)$ 時間が必要だが、動的計画法によるアルゴリズムを用いると、$O(n)$ 時間で計算することができる。また、漸近的には動的計画法の時間計算量と同じだが、実際の実行時間をさらに改善するアルゴリズムとして、ホーナーの方法とよばれるアルゴリズムがある。
2. n 個の行列の連続積を求める場合、積の結合則よりどのような順番で連続積を求めても答えは同じだが、計算時間には大きな違いがでる。動的計画法によるアルゴリズムを用いると、計算時間がもっとも短くなる行列積の計算順序を **$O(n^3)$** 時間で求めることができる。
3. $n \times n$ の2つの行列の積を基本的なアルゴリズムで計算すると、$O(n^3)$ 時間が必要であるが、分割統治法を用いた再帰的アルゴリズムであるストラッセンの行列積アルゴリズムを使うと、**$O(n^{\log_2 7}) = O(n^{2.81})$** 時間で行列積を求めることができる。

---

1. 文字列照合とは、与えられたテキストの中から、指定されたパターンと一致する部分を見つけ出す操作である。
2. 文字列照合を行う基本的なアルゴリズム（ちからまかせ法）では、パターンの先頭から順に比較を行うが、不一致が起こった場合に 1 文字ずつしかずらすことができないため、最悪時間計算量は **$O(mn)$** となる。
3. ボイヤー・ムーア法（BM 法）は、パターンの末尾から左に向かって比較を行い、不一致文字の情報を利用して大きくスキップするアルゴリズムである。テキスト中の文字の種類数を C とすると、最良時間計算量は **$O(C + n/m)$**、最悪時間計算量は **$O(C + n)$** である。
4. ホールスプール法は、BM 法を簡略化したアルゴリズムであり、不一致が起こった際に「現在のテキストの窓の右端にある文字」の情報のみを用いてスキップ量を決定する。実装が容易でありながら、多くの場合において BM 法に近い非常に高い効率を実現できる。

---

1. 問題を解くもっとも高速なアルゴリズムの最悪時間計算量を、問題の複雑さといい、同じ複雑さの問題の集合を問題のクラスとよぶ。問題のクラスはその時間計算量によって包含関係を構成しており、この包含関係を問題のクラス階層とよぶ 。
2. 問題を解くアルゴリズムの時間計算量が入力サイズの多項式で表される問題のクラスを、クラス**P**とよぶ。また、問題に対する解が与えられたとき、その問題の解が正しいかどうかを多項式時間で確かめられる問題のクラスを、クラス**NP**とよぶ。定義より、$P \subseteq NP$ であるが、$P \neq NP$ が成り立つかどうかは、情報科学分野の最大の未解決問題の1つである 。
3. 解きたい問題Aの入力を問題Bの入力に変換し、問題Bを解くアルゴリズムにより問題Aの解を求めるという方法を、問題の帰着とよぶ。とくに、入出力の変換に必要な時間計算量が入力サイズの多項式のオーダであるとき、問題Aは問題Bに多項式時間で帰着可能であるという 。
4. **NP**困難問題とは、クラスNPに含まれるすべての問題を多項式時間で帰着可能である問題であり、**NP**完全問題とは、NP困難かつクラスNPに含まれる問題である。NP完全問題は、その定義よりクラスNPに含まれる問題の中でもっとも難しい問題だと考えられている 。
5. 停止性判定問題に代表されるようないくつかの問題は、コンピュータを用いて解くことができないことが証明されている。このような解くことができない問題のことを非可解な問題とよぶ 。