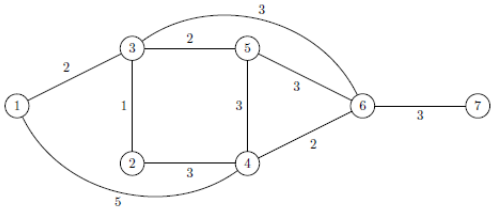
8
(1) グリーディ法とは，各ステップにおいてその場での最善（局所的な最適解）を選択し続けることで，最終的に全体としての最適解を得ようとするアルゴリズムの設計指針である．
(3) ナップザック問題とは，容量制限のあるナップザックに対して，入れる荷物の合計重量が容量を超えない範囲で，荷物の合計価格を最大化する組み合わせを求める組み合わせ最適化問題である．
(4) 0-1 ナップザック問題とは，各荷物を「入れる（1）」か「入れない（0）」かのいずれかしか選べない，つまり荷物を分割して入れることができないという制約を持つナップザック問題である．.
(12) 荷物リストに対して容量 7kg の 0-1 ナップザック問題として動的計画法を適用すると，ブレンド（5kg）とモカ（2kg）を選択した場合に合計価値が最大値の 7000 円となる．

番号 i	種類 x_i	重さ w_i [kg]	価格 v_i [円]
1	モカ	2	2000
2	キリマンジャロ	1	1000
3	コロンビア	4	2000
4	ブレンド	5	5000
5	マウンテン	3	3000

9
(1) バックトラック法とは，解の候補を次々と試していき，その道筋では解が得られないと判断した時点で一歩手前に戻って別の選択肢を試すという，再帰的な試行錯誤による探索アルゴリズムである．
(3) 8 クイーン問題の解探索を実現するためには，各行に 1 つずつクイーンを配置しながら条件を確認し，行き詰まったら前の行に戻るバックトラック法を適用するのが最適である．
(4) 8 クイーン問題の枝刈りは，ある行にクイーンを置いた時点で，すでに配置済みのクイーンと攻撃し合う位置にある後続の探索ルートをすべて破棄し，無駄な探索を省略するという考え方で実行すればよい．
(5) 分岐限定法とは，バックトラック法に「解の質の評価」を組み合わせ，各ノードで得られる解の見込み（下界値や上界値）を計算し，現在の暫定解より良くなる見込みがない分岐を切り捨てることで探索を効率化する手法である．
(6) 分岐限定法の手順の中で行う解探索のための操作は「分枝（Branching）」と「限定（Bounding）」と呼ばれ，問題を部分問題に分割してツリー構造を作る操作と，各部分問題の可能性を評価して探索の要否を判断する実行の意味を持つ．

１０
(1) 隣接行列とは，グラフの頂点間の隣接関係を正方行列で表したデータ構造であり，頂点 i と j の間に枝がある場合に (i,j) 要素を 1（あるいは重み），ない場合に 0 とするものである．
(3) 隣接リストとは，グラフの各頂点に対して，その頂点に隣接している頂点のリストを保持する形式のデータ構造である．
(7) 隣接行列を用いて全ノードの接続を確認する場合，全要素を走査する必要があるため，頂点数を V とすると時間計算量は O(V^2)となる．
(8) 隣接リストを用いて全ノードの接続を確認する場合，全頂点のリストに含まれる枝の総数を走査するため，頂点数を V ，枝数を E とすると時間計算量は O(V+E)となる．
(9) 縦型探索（深さ優先探索）は，探索可能な限り一つの道を深く進み，行き止まりに達したら一歩戻って別の枝を探索するという，スタックや再帰を利用した探索手法である．
(10) 横型探索（幅優先探索）は，始点に近い頂点から順に，同じ深さの頂点をすべて訪問してから次の深さへと進む，キューを利用した探索手法である．
(13) 深さ優先探索でスタックを用いるのは，最後に発見した未訪問の隣接頂点を優先的に探索するという「後入れ先出し（LIFO）」の性質が，探索の「深く進む」挙動と一致するためである．
(14) 最短経路問題とは，ネットワーク上の 2 つの頂点間を結ぶ経路のうち，経路上の枝の重みの総和が最小となる経路を求める問題である．
(15) ダイクストラ法は，始点からの最短距離が確定した頂点の集合を順次広げていくことで，各頂点への最短経路を効率的に決定するアルゴリズムである．
(17) 頂点 v_1 を始点としてダイクストラ法を実行すると，各頂点への最短距離は v_2: 3, v_3: 2, v_4: 5, v_5: 4, v_6: 5, v_7: 8$ と順次確定する．



(19) ハミルトン閉路問題とは，グラフのすべての頂点をちょうど一度だけ通り，かつ出発点に戻ってくるような閉路が存在するかどうかを判定する問題である．
(21) オイラーグラフとは，グラフのすべての枝をちょうど一度だけ通り，かつすべての頂点を通過して出発点に戻る閉路（オイラー閉路）が存在するグラフのことである．

１１
(3) ホーナーの方法のアイデアは，多項式を と展開せずに のように括り出すことで，乗算回数を最小化することである．
(5) 行列の連続積の計算順序を検討すべき理由は，行列のサイズによって乗算回数が劇的に変化するためであり，例えば の計算では結合順序次第で演算コストが数倍以上変わるからである．
(7) 行列の連続積の問題の定義は，与えられた複数の行列の積を計算する際に，スカラー乗算の総回数が最小となるような括弧の付け方（計算順序）を決定することである．
(10) ストラッセンの行列積アルゴリズムのアイデアは，通常の行列積で必要な 8 回の乗算を，特殊な加減算を組み合わせることで 7 回に減らし，再帰的に適用することで全体の計算量を より小さくすることである．
(12) 行列の大きさの定義に基づき((A1A2)A3)A4 を計算する場合，まず A1A2 で 10 回，A1 A2$で 10×2×10 = 200 回，その結果と A3$の積で 10×10×20 = 2000 回，さらに A4 との積で 10×20× 4 ＝800 回を要し，合計で 3000 回の演算となる．

行列の記号	行	列
A_1	10	2
A_2	2	10
A_3	10	20
A_4	20	4

(13) 動的計画法を用いて表の行列積の最小演算回数を求めると， の順序で計算した場合に で 800 回， で 80 回， で 80 回の計 960 回となり，これが最も少ない演算回数である．

１２
(1) 文字列照合とはどのよう問題か簡潔に説明せよ
作成中の文章から指定した検索文字列を探してり，置換したりする操作.
(3) ホール・スプール法のアルゴリズムとしてのアイデアを簡潔に説明せよ。
ホール・スプール法は、テキストとパターンを重ね合わせ、不一致が生じた場合にパターンを右に 1 つずつずらして比較を繰り返すという、素朴な全探索のアイデアに基づいています。
(4) ホール・スプール法の時間計算量を示せ。
比較対象となるテキストの長さ n，パターンの長さ m とすると、時間計算量は O(nm)
(7) ボイヤー・ムーア法のアルゴリズムとしてのアイデアを簡潔に説明せよ。
ボイヤー・ムーア法は、パターンの末尾から先頭に向かって逆方向に比較を行い、不一致が生じた際に「不一致文字」や「一致した接尾辞」の情報を用いて、可能な限り大きくパターンを右にスキップさせること効率化を図るアイデアに基づいています。

13
(1) 問題の複雑さとは，その問題を解くために必要な計算資源の最小量を表す尺度であり，通常は入力サイズ n に対する最悪時間計算量のオーダーを用いて定義されます．
(2) アルゴリズムの質が異なる場合でも，それぞれの最悪時間計算量を O 記法で表現することで，入力サイズ n が十分に大きくなった際の実行時間の増加率を共通の尺度と

して比較します.
(3) 貪欲法で最適解が求まる分割ナップザック問題は $O(n ¥log n)$ 程度で解ける「易しい問題」ですが，部分和問題は指数時間を要すると考えられている「難しい問題」であり，両者の複雑さには大きな隔たりがあります.
(4) 問題のクラスは，決定性または非決定性のチューリングマシンで解く際に必要となる「時間（計算ステップ数）」や「空間（メモリ使用量）」といった計算資源の量を尺度として分類できます.
(5) クラス P は，決定性アルゴリズム（現実的なコンピュータ）を用いて，入力サイズの多項式時間内に解くことができる問題の集合です.
(6) クラス NP は，ある解の候補が正しいかどうかを多項式時間内で検証できる問題，あるいは非決定性アルゴリズムを用いれば多項式時間内で解ける問題の集合です.
(8) 分割問題は「集合を和が等しい 2 つの部分集合に分けられるか」という問いですが，これは「目標値を全要素の和の半分とした部分和問題」の特殊ケースとして帰着させることが可能です.
(9) NP 完全とは，その問題がクラス NP に属し，かつ全ての NP に属する問題から多項式時間で帰着可能であるという，NP の中で最も難しい問題の性質を指します.
(10) NP 困難とは，全ての NP に属する問題から多項式時間で帰着可能であるという性質を持ち，必ずしもクラス NP に属している必要はない（NP と同等以上に難しい）問題の性質を指します.
(11) NP 困難な問題のうち，クラス NP に含まれるものが NP 完全と呼ばれ，包含関係としては NP 完全問題は NP と NP 困難の共通部分に位置します.
(12) 例えば「ハミルトン閉路が存在するか」という判定問題（NP 完全）は「Yes/No」を出力しますが，その最適化版である「巡回セールスマン問題」（NP 困難）は「最小の移動距離」という具体的な数値を出力します.
(13) クラス NP に属する問題の複雑さは，効率的な解法が見つかっていないため指数時間が必要と考えられている一方で，解の妥当性チェックだけは多項式時間で済むという非対称的な特徴に基づいています.
(14) 3-SAT（3 次充填可能性問題）は代表的な NP 完全問題であり，与えられた論理式を真にする変数の割り当てが存在するかどうかを判定する問題です.
(15) ナップザック問題（最適化版）は代表的な NP 困難問題であり，容量制限のあるバッグに価値が最大となるように荷物を詰め込む組み合わせを決定する問題です.
(16) 現時点で NP 完全な問題を多項式時間で解くアルゴリズムが一つも見つかっていないことや，解の発見と検証では明らかに発見の方が困難であるという直感的な経験則から $P \neq NP$ と考えられています.
(17) P=NP が証明された場合，「RSA 暗号などの現代の暗号体系が容易に解読される」というセキュリティ上の崩壊と，「創薬や物流最適化などの複雑な課題が瞬時に解決可能になる」という技術革新の 2 点の影響が考えられます.
(18) 停止性判定問題とは，あるプログラムが任意の入力に対して有限時間内に終了するか，あるいは無限ループに陥るかを判定する問題であり，アルゴリズム的には解くことが不可能な「計算不能な問題」として知られています.

27 週
ランレングス法，くり返し同じパターンが表れる場合に有効
エントロピー
情報エントロピーが大きい状態 -> 結果の予想がつかない，情報として曖昧
期待値みたいなもの，単位 bit$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$
すべての事象が同じ確率の場合$H(X) = \log_2 n$
，等確率のサイコロの場合，$\log_2 6 = 2.585$
平均符号長: 符号したときの平均の長さ，エントロピーより短く符号化できない
ハフマン符号化はできるだけ平均符号長をエントロピーに近づけるための方法