データ構造とアルゴリズムII

第5回
貪欲アルゴリズム

16. 貪欲アルゴリズム

貪欲アルゴリズム

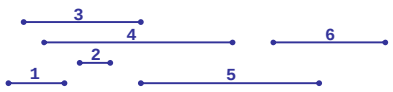
- 貪欲法では，選択をする時点で，最も良さそうに見える選択肢にコミットする
  - 日常生活では一般的
    - 分かれ道で，とりあえず目的地に近づく方を選ぶ
  - 局所的に最適な選択が，大域的な最適解を与えることを期待する
  - 常にうまくいくとは限らないが，実際に最適解が得られる問題が存在

活動選択問題

- テーマパークでパスポートを購入した
- どのライドも乗り放題
- 各ライド毎に，開始時刻と終了時刻が与えられている
- 目的は，なるべく多くのライドに乗ること
  - ライドの面白さとか，乗っている時間の長さとかは気にしない．開始時刻が異なれば，異なるライドだと仮定する．

活動選択問題

- S: n 個の活動の集合
  - s_i = 活動 i の開始時刻
  - f_i =活動 i の終了時刻
  - 目的は，両立可能な（時間に重なりのない）S の要素数最大の部分集合 A を求めること



■ 一般性を失うことなしに以下を仮定:

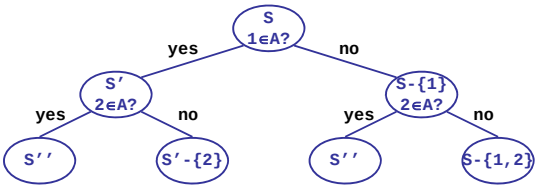
$f_1 \le f_2 \le \ldots \le f_n$

活動選択問題：部分問題最適性

- k をAの中で最小（開始時刻が最も早い）活動とすると，A - $\{k\}$ は $S' = \{i \in S: s_i \ge f_k\}$に関する最適解である
  - 言い換えると，活動kが選択されたなら，残りの問題は S 中で kと両立可能な最適解を求めることに帰着される．
  - 証明: もし，以下が満たされる S' の最適解Bが存在するとする: $|B| > |A - \{k\}|$,
    - $B \cup \{k\}$ は両立可能
    - $|B \cup \{k\}| > |A|$ となり，Aが最適解であるという仮定に矛盾

活動選択問題：厳密解を得るための再帰手続き

- 以下のように再帰的に問題を解けば，厳密な最適解が得られることは明らか



228

貪欲な選択

- 実は，最適解を得るために，どのライドを選ぶべきかは簡単に決められる ⇒ 最も早い終了時刻を持つライドを選ぶべき
  - 局所的に最適な選択＝大局的最適解を導く選択
  - 定理 16.1: 活動選択問題 S（終了時刻順にソートされている）において，以下の最適解Aが存在する：$\{1\} \in A$
    - 証明の概略: もし1を含まない最適解Bが存在したとすると，Bの最初の要素を1と入れ替えたものも，両立可能な解となり，これはBと同じ要素数なので最適．

229

活動選択問題：貪欲法

アルゴリズムは簡単：

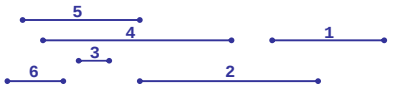
- 活動を終了時刻の早い順にソート
- 最初の活動を選択
- 次に，最初の活動の終了後にスタートする活動中で，終了時刻が最も早いものを選択
- 上記の手続きを，活動がなくなるまで繰り返す

- 以下の適応型の戦略でも大丈夫：今から乗れる，最も早く終わるライドを選ぶ

230

演習：活動選択問題

- 以下の手続きで最適解が得られるか?
  - 活動を開始時間が遅い順にソート
  - 最後の活動を選択
  - 次に，最後の活動の開始時間より前に終了する活動中で，開始時刻が最も遅いものを選択
  - 上記の手続きを，活動がなくなるまで繰り返す



231

ナップサック問題

- 泥棒がある店に盗みに入っている．色々な商品があり，価値および重さは様々である．持っているナップザックが運べる最大重量に収まるだけの商品しか盗めない．利益を最大化するには，どのように商品を選べばよいか?

232

ナップザック問題：定義

- *0-1* ナップザック問題:
  - n 個の財, i番目の財の価値は v_i で重さは w_i .
  - 重さの合計がW 以内で，価値を最大化する財の組合せを求める
    - v_i, w_i, および W はすべて自然数
    - “0-1” の意味は，財は取るか取らないかのどちらか
- 変形として，有理ナップザック問題：
  - 財を分割して，一部を取ることが可能
  - 価値／重さは比例配分

233

部分問題最適性

- 最適解から，財 j を取り除いた場合，残りの解は，j 以外の財に関して，容量$W - w_j$ に関する最適解に対応
- 0-1でも有理でも成立

234

ナップザック問題の貪欲解法

- v_i / w_i，すなわち単位重量当たりの価値で財をソートする
- 有理の場合は最適解が得られる（全部入るなら入れる，全部入らないなら，容量一杯まで入れる）
- 0/1の場合はダメ．例えば，財1を取ると，半端に余って残りの財が入らないが，財2, 3でちょうど一杯で，こっちの方が価値の合計が高い場合

235

演習：ナップザック問題

- *0-1* ナップザック問題:
  - n 個の財, i番目の財の価値は v_i で重さは w_i .
  - 重さの合計がW 以内で，価値を最大化する財の組合せを求める
    - v_i, w_i, および W はすべて自然数
    - “0-1” の意味は，財は取るか取らないかのどちらか
- この問題を動的計画法で解くにはどうしたらよいか?

236

ナップザック問題：ヒント

- C[i][w]: i番目までの財を使って，重さの合計がw以下の最適解の価値の合計
- C[i+1][w] = max (v_{i+1}+C[i][w－w_{i+1}], C[i][w])
- 以下の表を埋める

i	v	w	w=0	1	2	3	4	5
0			0	0	0	0	0	0
1	4	2	0					
2	5	2	0					
3	2	1	0					
4	8	3	0					

237

ナップザック問題：解答

- C[i][w]: i番目までの財を使って，重さの合計がw以下の最適解の価値の合計
- C[i+1][w] = max (v_{i+1}+C[i][w－w_i], C[i][w])
- 以下の表を埋める

item[N+1]　　C[N+1][W+1]

	v	w	0	1	2	3	4	5
0			0	0	0	0	0	0
1	4	2	0	0	4	4	4	4
2	5	2	0	0	5	5	9	9
3	2	1	0	2	5	7	9	11
4	8	3	0	2	5	8	10	13

238

ハフマン符号化

- データを効率的に圧縮する方法
- データは文字列であることを仮定
- どの文字列に，どのコードを割り当てるかを決める：コードの長さは可変

Alphabet:	a	b	c	d	e	f
Frequency in a file	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

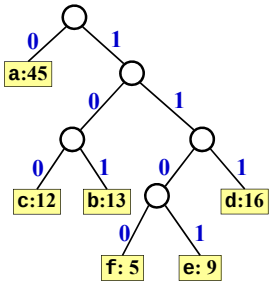
file length 1 = 300;　file length 2 = 224

Compression ratio = (300–224)/300·100% ≈ 25%

239

接頭語符号と対応する木

- 接頭語符号：どの符号も，他の符号の接頭語（符号の最初の一部）にならない
- 木を使って復号可能
- ハフマン符号は接頭語符号の一種



240

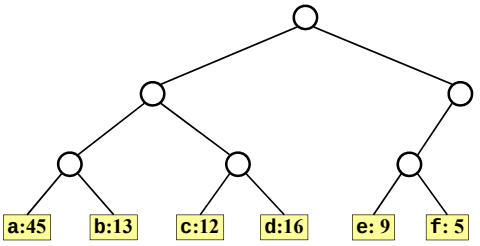
最適な符号化

- アルファベットCに対して，その符号化を表す木 T を考える．f(c) を，c∈Cのファイル中の頻度，$d_T\ (c)$を，cを表すノードのT中の深さ（すなわち符号cの長さ）とする
- 符号化したファイルのサイズは以下で与えられる：$\sum_{c \in C} f(c) d_T\ (c)$.
- 上記を最小化するような符号化を求めたい

241

性質1

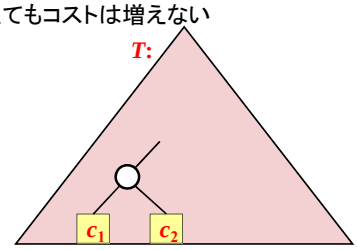
- 最適符号化に対応する木では，葉以外のノードは，二つの子ノードを持つ



242

性質2 (補題16.2)

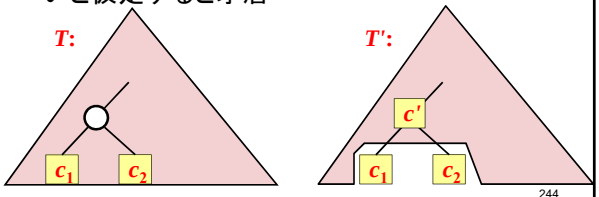
アルファベット中で，最も頻度が小さい文字をc_1, c_2とすると，これらの符号長が同じで，かつ，これらの符号が最後の一ビットを除いて同じとなる最適符号化方法が存在する: 任意の木で，これらの文字を最も深いノードに置き換えてもコストは増えない



243

性質3 (補題16.3)

- アルファベット中で，最も頻度が小さい文字をc_1, c_2とする．c_1, c_2を除いて新たなアルファベットc' ($f(c') = f(c_1) + f(c_2)$ が成立) を加えた場合，以下のT'が最適符号化となる：そうでないと仮定すると矛盾



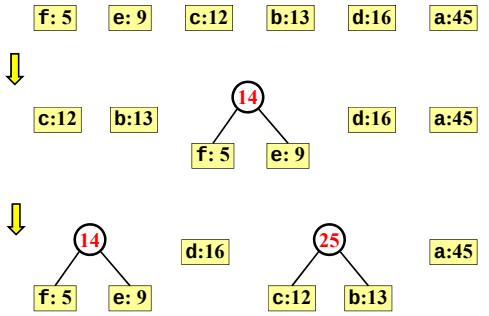
244

ハフマン符号化アルゴリズム

- もっとも頻度の小さいアルファベットを木の最深部の葉とする．これらをまとめて一つのアルファベットに置き換える．上記の処理を繰り返す．

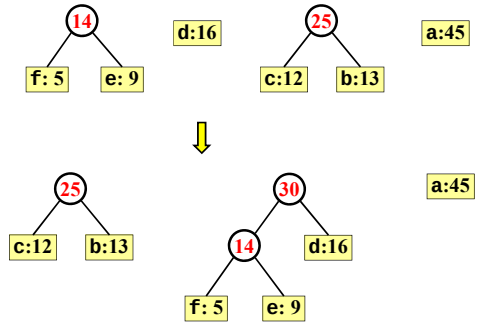
245

実行例



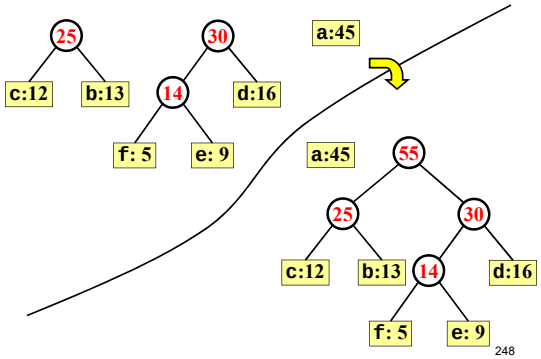
246

実行例



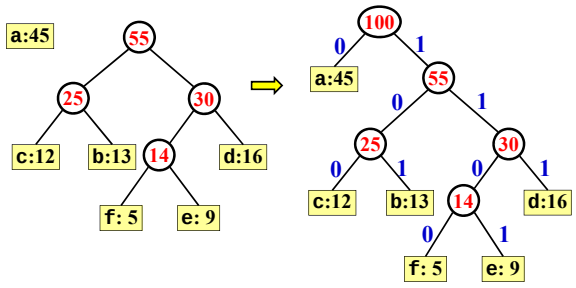
247

実行例



248

実行例



249

マトロイド：貪欲法の理論的背景

- マトロイド (matroid) とは，行列 (matrix) のようなもの (-oid) という意味

$$A = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, X = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$$

$$\mathcal{F} = \left\{ \emptyset, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\}, \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right\}, \ldots \right\}$$

行列 A の列ベクトルの集合を X，さらに，$\mathcal{F}$ を，X の部分集合で，要素が線形独立なものの族 (集合の集合) とすると，以下の3つの性質が成立する

1. $\emptyset \in \mathcal{F}$,
2. $X' \in \mathcal{F}$ かつ $X'' \subset X'$ ならば，$X'' \in \mathcal{F}$,
3. $X', X'' \in \mathcal{F}$ かつ $|X'| > |X''|$ ならば，$X'' \cup \{x\} \in \mathcal{F}$ となる $x \in X' \setminus X''$ が存在する．

マトロイド：定義

Definition (マトロイド)

有限集合 X および X の部分集合族 $\mathcal{F}$ が以下の3つの条件を満たすときに $(X, \mathcal{F})$ をマトロイドと呼ぶ

1. $\emptyset \in \mathcal{F}$,
2. $X' \in \mathcal{F}$ かつ $X'' \subset X'$ ならば，$X'' \in \mathcal{F}$,
3. $X', X'' \in \mathcal{F}$ かつ $|X'| > |X''|$ ならば，$X'' \cup \{x\} \in \mathcal{F}$ となる $x \in X' \setminus X''$ が存在する．

- マトロイドは計算機科学の様々な場面で頻出
- 貪欲法で最適解が得られる多くの問題はマトロイド構造を持っている (最小コスト被覆木，最大流量)

251

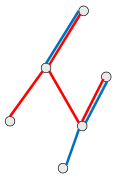
例：グラフマトロイド

- Xをあるグラフの辺の集合とする
- X’⊆Xに関して，X’が閉路を含まないなら（森なら），Fの要素であるとする
- (X, F) はマトロイドになる

252

例：グラフマトロイド

- 性質1, 2は明らか
- 3に関して，
  - X’, X”∈Fとして，|X’|>|X”| とする
  - それぞれの森に含まれる木の数を考えると，|X’|の方が少ない (隣接する辺が含まれない頂点は，単一の頂点からなる木だと思う)
  - X’の各木に関して，頂点を共有するX”の木を考えると，X’のある木に関して，X”の複数の木が対応している（鳩の巣原理）
  - よって，X’の辺で，X”に含まれず，かつ，X”に加えても閉路を構成しないものが，少なくとも一つ存在する



253

重み付きマトロイドと貪欲法

- Xの各要素xに関して，重みw(x)が定義されているとする
- Fが解の候補の集合として，F中で，重みの和が最小のものを求めたい
- 以下の貪欲法で最適解が得られる

L←X, S←øとする

1. Lが空なら, Sを解として返す
2. Lから，重みが最小の要素xを取り出す
3. {x}∪S∈Fなら，xをSに加える
4. 1に戻る

254

貪欲法の正しさ

- yを，{y} ∈Fで，重みが最小のものとする
- yを含む最適解が存在する
- なぜならyを含まない最適解X”が存在すると仮定すると，X”の要素はすべて重みがw(y)以上．{y}とX”に関して，性質3を使うと，yを含み，|X’|=|X”|となるX’が構成でき，X’の重みの和はX”以下
- 同様に，{y, z}∈Fを満たすものの中で，重みが最小のものをzとすると，y, zを含む最適解が存在することが示せる

255

グラフマトロイドでの貪欲法

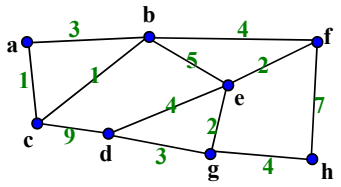
- 最小被覆木：すべての頂点を含む，重み和が最小となる木
- 貪欲法で構成可能 (Kruscalのアルゴリズム)

L←E, T←øとする

1. Lが空なら, Tを解として返す
2. Lから，重みが最小の辺eを取り出す
3. {e}∪Tが閉路を含まないなら，eをTに加える
4. 1に戻る

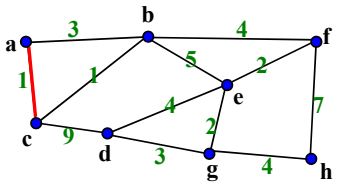
256

Kruskalのアルゴリズムの実行例



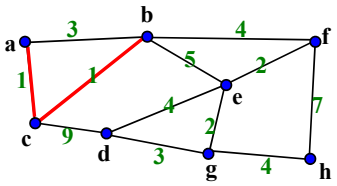
257

Kruskalのアルゴリズムの実行例



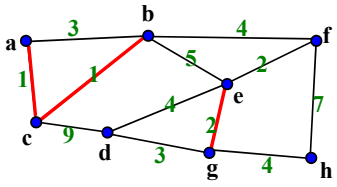
258

Kruskalのアルゴリズムの実行例



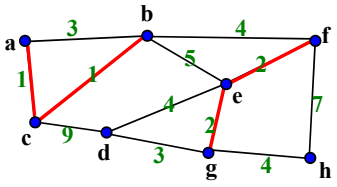
259

Kruskalのアルゴリズムの実行例



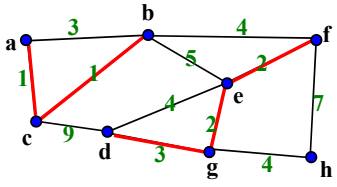
260

Kruskalのアルゴリズムの実行例



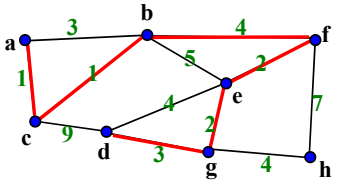
261

Kruskalのアルゴリズムの実行例



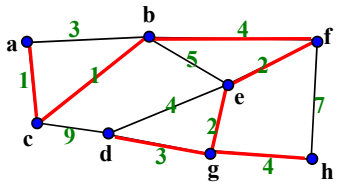
262

Kruskalのアルゴリズムの実行例



263

Kruskalのアルゴリズムの実行例



MST cost = 17

264