データ構造とアルゴリズムII

第3回
動的計画法 2

105

動的計画法のアルゴリズム

1. 最適解の構造を特徴づける（部分問題最適性）
2. 最適解の値を再帰的に定義する
3. ボトムアップに最適解の値を求める
4. 最適解を構成する

106

例題

連鎖行列積

107

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい

108

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に寄らない
    - $(A_1(A_2(A_3A_4)))$
    - $(A_1((A_2A_3)A_4))$
    - $((A_1A_2)(A_3A_4))$
    - $((A_1(A_2A_3))A_4)$
    - $(((A_1A_2)A_3)A_4)$

109

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に依存しない
    - $(A_1(A_2(A_3A_4)))$: 右から
    - $(A_1((A_2A_3)A_4))$: 中右左の順
    - $((A_1A_2)(A_3A_4))$: 左右を先に
    - $((A_1(A_2A_3))A_4)$: 中左右の順
    - $(((A_1A_2)A_3)A_4)$: 左から

110

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に依存しない
  - 計算量は括弧付けの順番に依存する

$A(p\times q$行列$)$ と $B(q\times r$行列$)$ の積では，pqr回の掛け算が実施される

111

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に依存しない
  - 計算量は括弧付けの順番に依存する

(例) $A_1(10\times 100$行列$)$ $A_2(100\times 5$行列$)$ $A_3(5\times 50$行列$)$

$((A_1A_2)A_3)$: $10\times 100\times 5+10\times 5\times 50$

$(A_1(A_2A_3))$: $100\times 5\times 50+10\times 100\times 50$

112

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に依存しない
  - 計算量は括弧付けの順番に依存する

(例) $A_1(10\times 100$行列$)$ $A_2(100\times 5$行列$)$ $A_3(5\times 50$行列$)$

$((A_1A_2)A_3)$: $10\times 100\times 5+10\times 5\times 50$ $= 7{,}500$

$(A_1(A_2A_3))$: $100\times 5\times 50+10\times 100\times 50$ $=75{,}000$

113

問題

- 行列積 $A_1A_2\ldots A_n$ の計算量を小さくしたい
  - 演算結果は括弧付けの順番に依存しない
  - 計算量は括弧付けの順番に依存する

→ **$A_1A_2\ldots A_n$ の乗算回数を最小化する括弧付け**を決定する

- A_i の次元は $p_{i-1}\times p_i$

114

括弧付けの個数

$P(n)$：N個の行列に対する括弧付けの個数

$$P(n)=\begin{cases}1 & (n=1)\\ \sum_{k=1}^{n-1}P(k)P(n-k) & (n\geq 2)\end{cases}$$

総当りには$\Omega(4^n/n^{3/2})$必要

115

括弧付けの個数

$P(4)$：4個の行列に対する括弧付けの個数

$(A_1(A_2(A_3A_4)))$ $((A_1(A_2A_3))A_4)$

$(A_1((A_2A_3)A_4))$ $((A_1A_2)(A_3A_4))$ $(((A_1A_2)A_3)A_4)$

116

括弧付けの個数

$P(4)$：4個の行列に対する括弧付けの個数

$(A_1(A_2(A_3A_4)))$
$(A_1((A_2A_3)A_4))$
$((A_1A_2)(A_3A_4))$
$(((A_1A_2)A_3)A_4)$
$((A_1(A_2A_3))A_4)$

$P(1)$ $P(3)$　$P(2)$ $P(2)$　$P(3)$ $P(1)$

括弧付けの個数

$P(4)$：4個の行列に対する括弧付けの個数

$(A_1(A_2(A_3A_4)))$
$(A_1((A_2A_3)A_4))$
$((A_1A_2)(A_3A_4))$
$(((A_1A_2)A_3)A_4)$
$((A_1(A_2A_3))A_4)$

$P(1) \times P(3) + P(2) \times P(2) + P(3) \times P(1)$

$\rightarrow \sum_{k=1}^{n-1} P(k)P(n-k)$

アルゴリズムのステップ1: 最適解の構造を特徴づける

$A_iA_{i+1}\ldots A_j$の最適括弧付けが

- A_kとA_{k+1}で分割すると仮定する

$((A_i \; A_{i+1}\ldots A_k)(A_{k+1}\ldots A_j))$

$A_iA_{i+1}\ldots A_j$の最適括弧付けが

- A_kとA_{k+1}で分割すると仮定する

$((A_i \; A_{i+1}\ldots A_k)(A_{k+1}\ldots A_j))$

部分括弧付けは$A_i\ldots A_k$の**最適括弧付け**となっている

部分括弧付けは$A_{k+1}\ldots A_j$の**最適括弧付け**となっている

なぜならば

- より良い括弧付けがあればそれを使って$A_iA_{i+1}\ldots A_j$をより良くできる

$((A_i \; A_{i+1}\ldots A_k)(A_{k+1}\ldots A_j))$

より良い括弧付け

なぜならば

- より良い括弧付けがあればそれを使って$A_i \ldots A_j$をより良くできる

$$\left(\left(A_i \;\; A_{i+1} \ldots A_k \right) \left(A_{k+1} \ldots A_j \right) \right)$$

より良い括弧付け

矛盾

$A_iA_{i+1}\ldots A_j$ が最適括弧付けであることと

問題($A_iA_{i+1}\ldots A_j$)の最適解に
部分問題($A_i\ldots A_k$ と $A_{k+1}\ldots A_j$)の最適解が
含まれる

||

部分問題最適性

アルゴリズムのステップ2.
最適解の値を
再帰的に定義する

最適解の値

$m[i, j]$ = $A_i\ldots A_j$の計算に必要なスカラ乗算の最小回数

$$\left(A_i \;\; A_{i+1} \ldots A_k \quad A_{k+1} \ldots A_j \right)$$

最適解の値

$m[i, j]$ = $A_i\ldots A_j$の計算に必要なスカラ乗算の最小回数

最適解は A_k と A_{k+1}で分割すると仮定する

$$\left(\left(A_i \;\; A_{i+1} \ldots A_k \right) \left(A_{k+1} \ldots A_j \right) \right)$$

最適解の値

$m[i, j]$ = $A_i\ldots A_j$の計算に必要なスカラ乗算の最小回数

最適解は A_k と A_{k+1}で分割すると仮定する

$$\left(\left(A_i \;\; A_{i+1} \ldots A_k \right) \left(A_{k+1} \ldots A_j \right) \right)$$

$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1}p_kp_j$$

未知の値 k を求める

$$m[i,j] = \begin{cases} 0 & (j=i) \\ \min_{i \leqq k < j}(m[i,k] + m[k+1,j] + p_{i-1}p_k p_j) & (j>i) \end{cases}$$

全ての k を調べて最良のものを選択する

アルゴリズムのステップ3：（ボトムアップに）最適解の値を求める

再帰だと指数時間が必要

部分問題の数が少ない

- $1 \leqq i \leqq k \leqq n$ を満たす部分問題の個数は
  - ${}_nC_2+n = \Theta(n^2)$ 通り
- 再帰の途中で同じ部分問題が繰り返し表れる

||

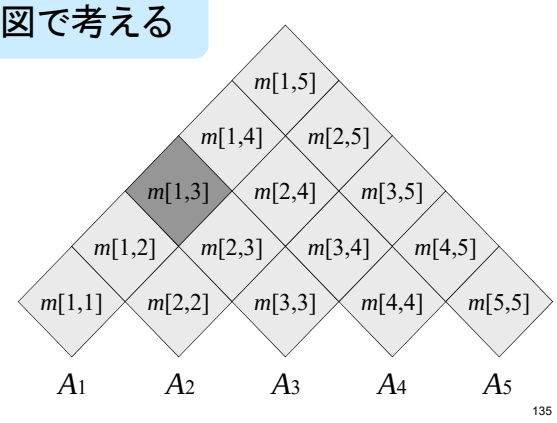
部分問題重複性

そこでボトムアップに

MATRIX-CHAIN-ORDER(p)

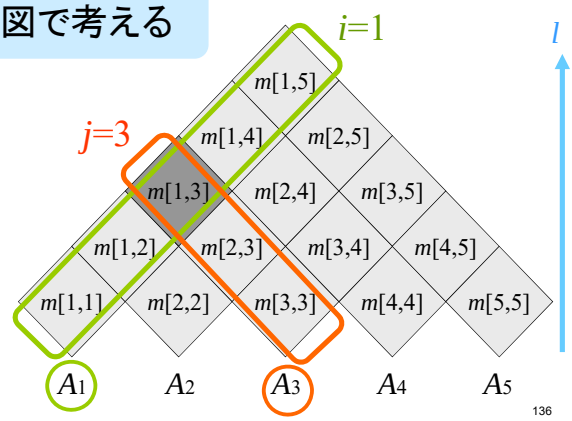
```
n ← length[p] − 1
for i ← 1 to n                          } 初期値を設定
    do m[i, i] ← 0
for l ← 2 to n                          } 連鎖長がlのものでループ
    do for i ← 1 to n − l + 1
        do j ← i + l − 1
           m[i, j] ← ∞
           for k ← i to j − 1
               do q ← m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j   } m[i, j]を求める
                  if q < m[i, j]
                     then m[i, j] ← q
                          s[i, j] ← k
```

図で考える



135

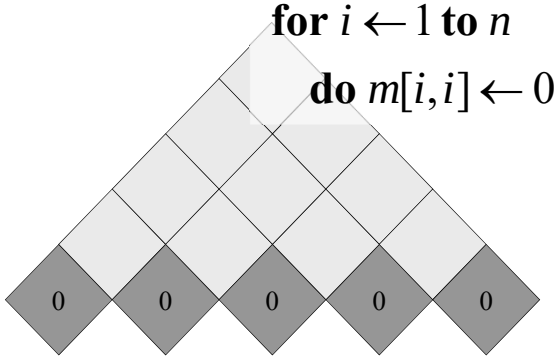
図で考える



136

for $i \leftarrow 1$ **to** n

do $m[i,i] \leftarrow 0$

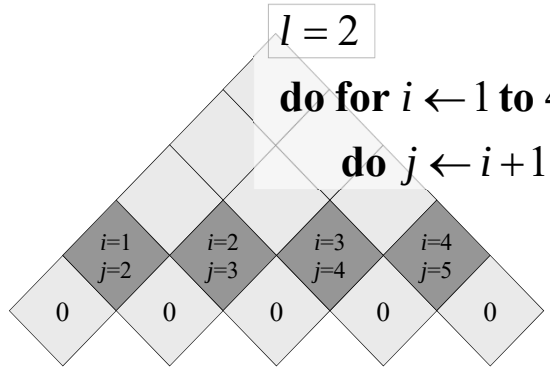


137

$l = 2$

do for $i \leftarrow 1$ **to** 4

do $j \leftarrow i + 1$



138

$l = 1, i = 1, j = 2$

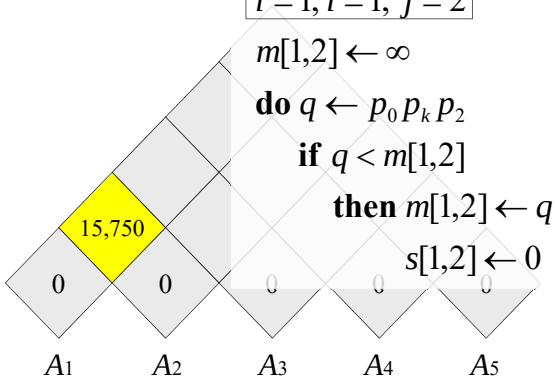
$m[1,2] \leftarrow \infty$

do $q \leftarrow p_0 p_k p_2$

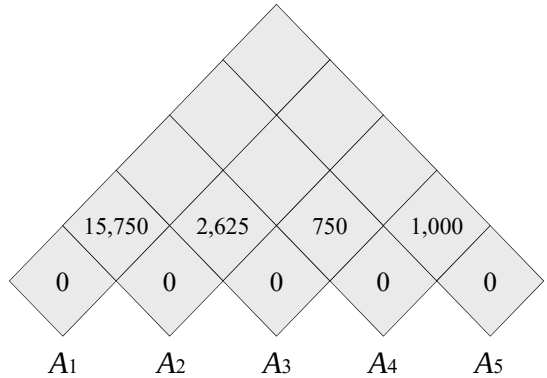
if $q < m[1,2]$

then $m[1,2] \leftarrow q$

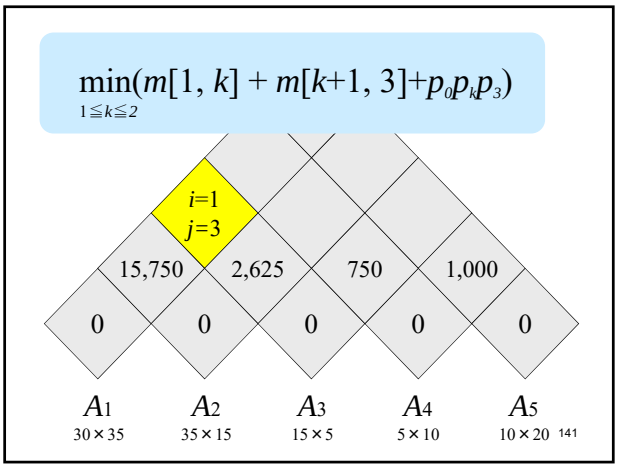
$s[1,2] \leftarrow 0$

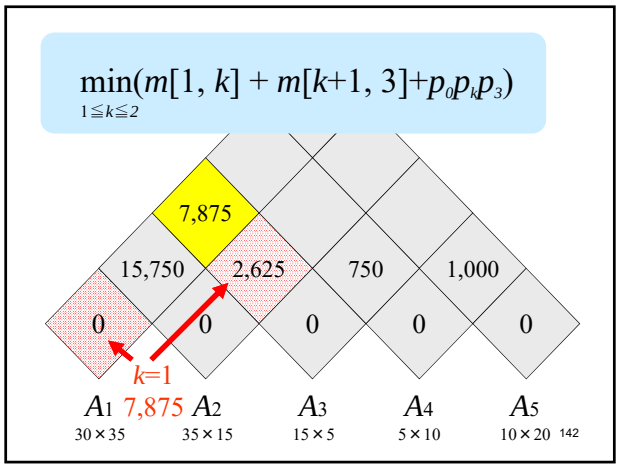


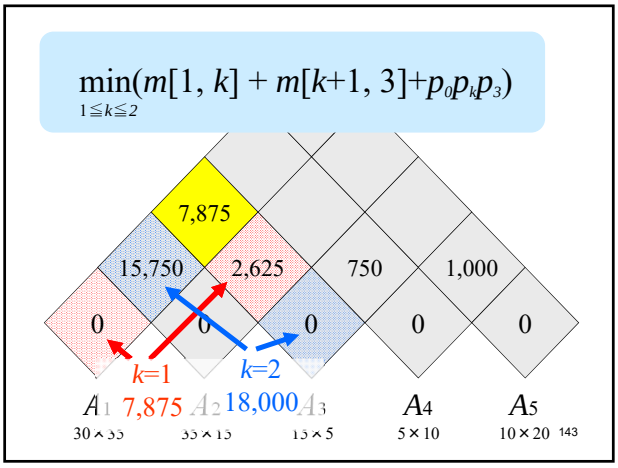
139

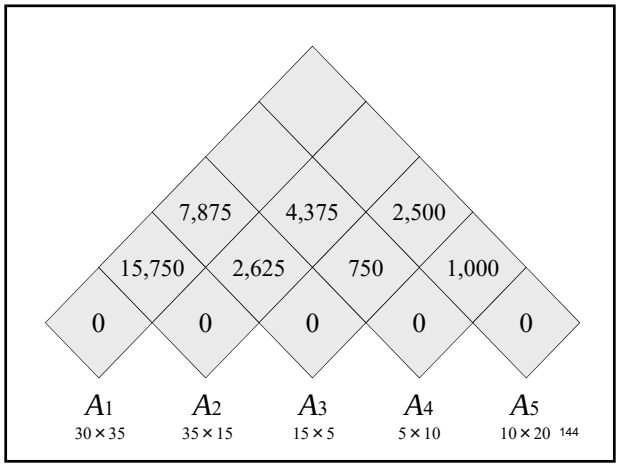


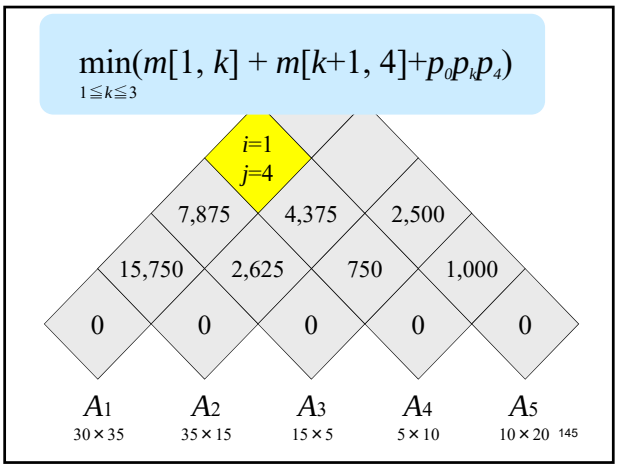
140

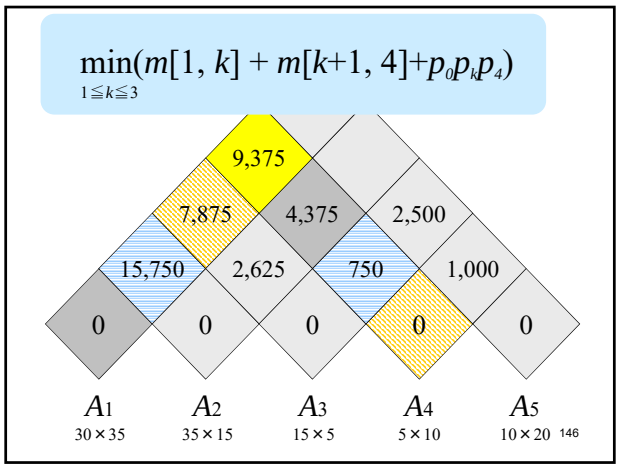

min(m[1, k] + m[k+1, 3]+p0pkp3)
1≦k≦2
i=1
j=3
15,750
2,625
750
1,000
0
0
0
0
0
A1
30×35
A2
35×15
A3
15×5
A4
5×10
A5
10×20
141

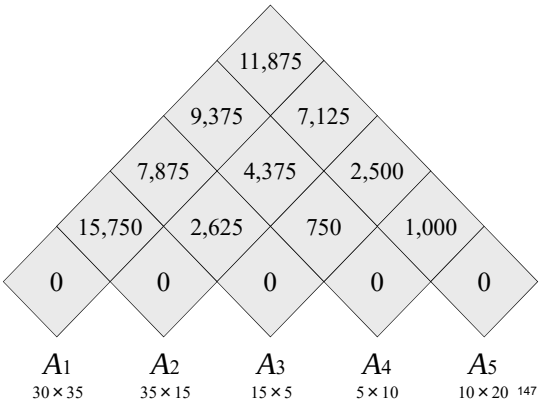

min(m[1, k] + m[k+1, 3]+p0pkp3)
1≦k≦2
7,875
15,750
2,625
750
1,000
0
0
0
0
0
k=1
A1
30×35
7,875
A2
35×15
A3
15×5
A4
5×10
A5
10×20
142


min(m[1, k] + m[k+1, 3]+p0pkp3)
1≦k≦2
7,875
15,750
2,625
750
1,000
0
0
0
0
0
k=1
k=2
A1
7,875
A2
18,000
A3
A4
5×10
A5
10×20
143

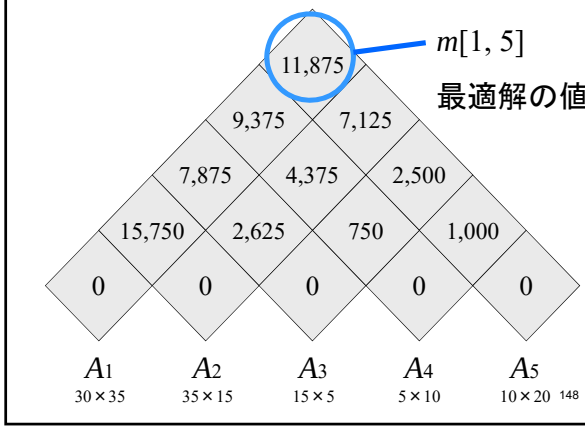

7,875
4,375
2,500
15,750
2,625
750
1,000
0
0
0
0
0
A1
30×35
A2
35×15
A3
15×5
A4
5×10
A5
10×20
144


min(m[1, k] + m[k+1, 4]+p0pkp4)
1≦k≦3
i=1
j=4
7,875
4,375
2,500
15,750
2,625
750
1,000
0
0
0
0
0
A1
30×35
A2
35×15
A3
15×5
A4
5×10
A5
10×20
145


min(m[1, k] + m[k+1, 4]+p0pkp4)
1≦k≦3
9,375
7,875
4,375
2,500
15,750
2,625
750
1,000
0
0
0
0
0
A1
30×35
A2
35×15
A3
15×5
A4
5×10
A5
10×20
146



147



148

アルゴリズムの評価

- 計算量
  - $O(n^3)$
  - $\Omega(n^3)$ →練習問題 15.2-4
- メモリ消費量
  - $\Theta(n^2)$

149

アルゴリズムのステップ4: 最適解を構成する

150

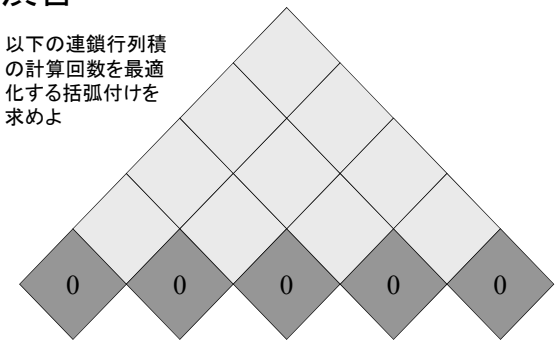
- $s[i, j]$ から最適解を構成する

```
PRINT-OPTIMAL-PARENS (s, i, j)
if i = j
  then print "A"+i
  else print "("
       PARENT – OPTIMAL – PARENS(s,i,s[i, j])
       PARENT – OPTIMAL – PARENS(s,s[i, j]+1, j)
       print ")"
```

151

演習

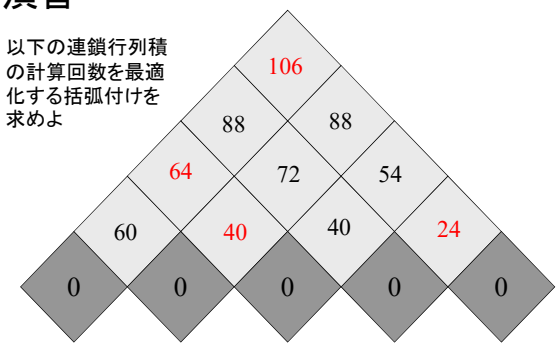
- 以下の連鎖行列積の計算回数を最適化する括弧付けを求めよ



152

演習

- 以下の連鎖行列積の計算回数を最適化する括弧付けを求めよ



153

15.3：動的計画法の基本要素

- 部分構造最適性
- 注意点
- 部分問題重複性

154

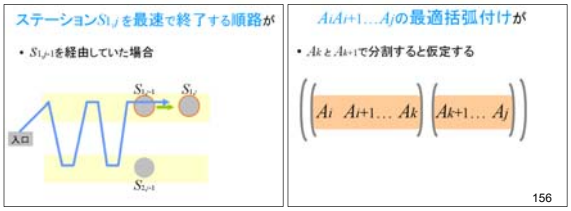
部分構造最適性を発見する手順

1. 問題の解がある選択から構成されていることを観察する.
2. 最適解を導く選択が与えられていると仮定する.
3. 選択から生ずる部分問題を決定する.
   部分問題の空間を特徴づける.
4. 部分問題の解が最適解であることを決定する.

155

例題で考える

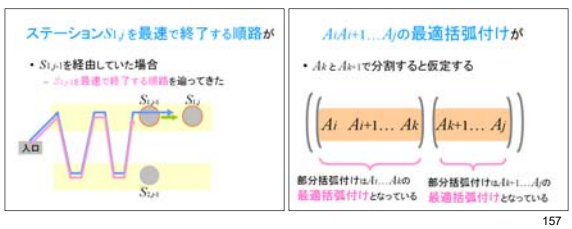
1. 問題の解がある選択から構成されていることを観察する.
2. 最適解を導く選択が与えられていると仮定する.
3. 選択から生ずる部分問題を決定する.
   部分問題の空間を特徴づける.



156

例題で考える

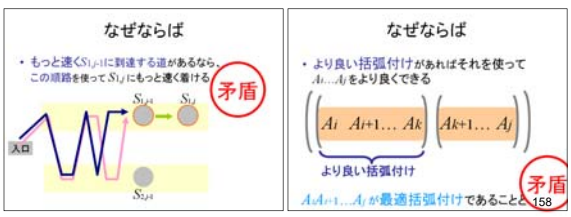
4. 部分問題の解が最適解であることを決定する.
   切り貼り法を使って証明する
   （最適ではないと仮定して矛盾を導く）



157

例題で考える

4. 部分問題の解が最適解であることを決定する.
   切り貼り法を使って証明する
   （最適ではないと仮定して矛盾を導く）



158

部分構造最適性を発見する手順

1. 問題の解がある選択から構成されていることを観察する.
2. 最適解を導く選択が与えられていると仮定する.
3. 選択から生ずる部分問題を決定する.
   部分問題の空間を特徴づける
4. 部分問題の解が最適解であることを決定する.

どのように？

159

部分問題の空間の特徴づけを得る方法

- 最初はできるだけ簡単に仮定する
- 必要に応じて拡張する

組み立てラインスケジューリングの場合
- $S_{1,j}$と$S_{2,j}$を仮定

→ うまく行く

160

部分問題の空間の特徴づけを得る方法

- 最初はできるだけ簡単に仮定する
- 必要に応じて拡張する

連鎖行列積の場合
- $A_1A_2\ldots A_j$ を部分問題の空間として仮定していれば…
  A_k と A_{k+1} で分割するとき
  部分問題 $A_1\ldots A_k$ と $A_{k+1}\ldots A_j$ を生じる

スタートのiも動かさないとダメ!

部分問題の空間に含まれない

161

特徴量

1. 元々与えられた問題の最適解が利用する部分問題の個数
2. 最適解が利用する部分問題の候補の個数

	1. 部分問題の個数	2. 候補の個数
組み立てラインスケジューリング	1 ($S_{1,j}$ もしくは$S_{2,j}$)	2 ($S_{1,j}$ と$S_{2,j}$)
連鎖行列積	2 ($A_i\ldots A_k$ と $A_{k+1}\ldots A_j$)	$j-i$ ($k=i,i+1,\ldots,j-1$)

162

計算時間

1. 部分問題の総数
2. 各部分問題で考慮すべき候補の数

問題の複雑度は，これらの積で与えられる

	1. 部分問題の総数	2. 候補数	合計
組み立てラインスケジューリング	$\Theta(n)$	2	$\Theta(n)$
連鎖行列積	$\Theta(n^2)$	高々n	$O(n^3)$

163

最適解の再構成

「最適解の値」を求めたあとに「最適解」を求める

- 組み立てラインスケジューリングの場合
  - 直前のステーションを $l_i[j]$ に記録した
  - $l_i[j]$がなくても，わずかな計算$O(n)$で再構成できる
    - $f_1[j] = f_1[j\text{-}1]+a_{1}$ ならば $S_{1,j}$ を経由
    - $f_1[j] = f_2[j\text{-}1]+t_{2,j\text{-}1}+a_{1}$ ならば $S_{2,j}$ を経由
- 連鎖行列積問題の場合
  - 直前の分割を $s[i, j]$ に記録した
  - $s[i, j]$を利用しないと，再構成には$\Theta(j\text{-}i)$が必要
  - $s[i, j]$があれば$O(1)$で計算できる

164

演習：部分和問題

- 5個の正の整数: 3, 4, 6, 7, 5が与えられる
- この部分集合の和となるような数の集合Sを求めたい
  - 例えば，7+5=12なので，12はSに含まれる．
  - どうやっても23にはできないので，23はSに含まれない
- 部分集合の数は2^n
- 部分集合をすべて数え上げずに，Sを求めることは可能?
- 最適化問題ではないので，最適性は気にしなくてもよい
- どのように部分問題を定義し，部分問題の解を拡張していけばよいか?

165

演習：部分和問題

- 最初のk個の数のみを使って得られる集合をS_kとする．
- S_{k+1}の任意の要素に関して，以下のどちらかが成立
  - S_k の要素である
  - S_k の要素に，$k+1$ 番目の数を加えたものである
- $S_0 = \{0\}$,
- $S_1 = \{0, 3\}$,
- $S_2 = \{0, 3, 4, 7\}$,
- $S_3 = \{0, 3, 4, 6, 7, 9, 10, 13\}$,
- $S_4 = \{0, 3, 4, 6, 7, 9, 10, 11, 13, 14, 16, 17, 20\}$,
- $S_5 = \{0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 25\}$

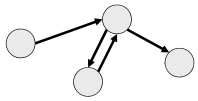
3, 4, 6, 7, 5

166

間違えて部分問題最適性を仮定しないために

- 有向グラフ $G=(V, E)$ で以下の問題を考える
  - 重みなし最短(単純)路
  - 重みなし最長単純路

単純路 = ループや多重辺を含まない



167

重みなし最短路問題

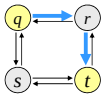
- 部分構造最適性を持つ
- $u \xrightarrow{p} v$ を $u \xrightarrow{p_1} w \xrightarrow{p_2} v$ に分解する
  - p_1 は uからwへの最短路
  - もしそうでないならば，最短路を p_1' とすれば p_1'と p_2を組み合わせてより最短な経路を作成できる

168

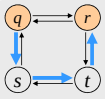
重みなし最長単純路問題

- 部分構造最適性を持たない

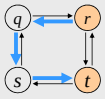
(反例)



$q \to t$ への最長路 2



部分問題
$q \to r$ への最長路 3



部分問題
$r \to t$ への最長路 3

169

重みなし最長単純路問題

- 動的計画法では解けない
- NP完全問題
  - 多項式時間では(多分)解けない

170

違いは「独立」

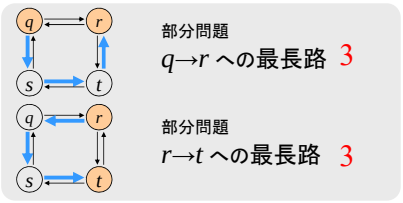
- 最短単純路：独立
- 最長単純路：独立でない

独立 =部分問題の解が別の部分問題の解に影響を与えない

171

最長単純路は「独立」か？

- 2つの最適解を組み合わせると，経路は単純でなくなる



部分問題の最適解が別の問題へ影響を与えている

172

最短路は「単純」か？

- 2つの部分問題はw以外の点を共有しないので**「単純」**

$$u \xrightarrow{p_1} w \xrightarrow{p_2} v$$

- もし w以外の点 x を共有すると
  - $u \to x \to w \to x \to v$
  - $x \to w \to x$ をショートカットすれば，より最短な経路になる
  - 矛盾！

173

部分問題重複性の性質

- 再帰の仮定で同じ問題が何度も出てくる
  - 動的計画法では部分問題の解を表に記録することで対応する
- 分割統治法
  - 再帰の仮定で新しい問題が生まれる

174

連鎖行列積を再帰で解いたら…

- $T(n)$:= 再帰で解いた場合の計算時間

$$\begin{cases} T(1) \geq 1 \\ T(n) \geq 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1) \quad (n > 1 \text{のとき}) \end{cases}$$

$$\| $$

$$2\sum_{i=1}^{n-1} T(i) + n$$

175

連鎖行列積を再帰で解いたら…(2)

- 置き換え法で $T(n)=\Omega(2^n)$ を証明する

$$\begin{aligned} T(n) &\geq 2\sum_{i=1}^{n-1} 2^{i-1} + n \\ &= 2\sum_{i=0}^{n-2} 2^i + n \\ &= 2(2^{n-1} - 1) + n \\ &= (2^n - 2) + n \\ &\geq 2^{n-1} \end{aligned}$$

つまり…
少なくとも 2^nの計算量！

176

メモ化 (memoize)

- 再帰の過程で計算結果を覚えておく
- 重複する部分問題を$O(1)$で思い出せると仮定

	動的計画法	再帰+メモ化	再帰のみ
連鎖行列積	$O(n^3)$	$O(n^3)$	2^n

177

動的計画法 v.s. 再帰＋メモ化

- 全ての部分問題を解く必要がある場合
  - → 動的計画法の方が優れている
  - – 再帰のオーバーヘッドが少ない
  - – 時間や表の領域を節約するアルゴリズムもある
- 一部の部分問題のみを解けばよい場合
  - – 再帰＋メモ化は不要な問題を解かなくてよい

178