

# データ構造と アルゴリズムII

第3回  
動的計画法 2

105

## 予定 (I)

4/11: 第1回: イントロダクション: マッチング

4/18: 第2回: 動的計画法1

4/25: 第3回: 動的計画法2

5/2: 金曜日の講義日

5/9: 小テスト

5/16: 第4回: 動的計画法3

5/23: 第5回: 貪欲法

5/30: 第6回: 計算量解析

6/6: 前半まとめテスト

106

- 再来週の5/9は小テスト
- 範囲はマッチングと動的計画法

107

## 例題 連鎖行列積

108

## 問題

- 行列積  $A_1A_2\dots A_n$  の計算量を小さくしたい

109

## 問題

- 行列積  $A_1A_2\dots A_n$  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

•  $(A_1(A_2(A_3A_4)))$

•  $(A_1((A_2A_3)A_4))$

•  $((A_1A_2)(A_3A_4))$

•  $((A_1(A_2A_3))A_4)$

•  $((((A_1A_2)A_3)A_4))$

110

### 問題

- 行列積  $A_1A_2\dots A_n$  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

111

### 問題

- 行列積  $A_1A_2\dots A_n$  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

112

### 問題

- 行列積  $A_1A_2\dots A_n$  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)

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

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

113

### 問題

- 行列積  $A_1A_2\dots A_n$  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)

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

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

114

### 問題

- 行列積  $A_1A_2\dots A_n$  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

→  $A_1A_2\dots A_n$  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  $p_{i-1} \times p_i$

115

### 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

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

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

117

### 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

$(A_1(A_2(A_3A_4)))$                        $((A_1(A_2A_3))A_4)$   
 $(A_1((A_2A_3)A_4))$      $((A_1A_2)(A_3A_4))$      $((A_1A_2)A_3)A_4$   
 $\underbrace{\hspace{1.5cm}}_{P(1)} \underbrace{\hspace{1.5cm}}_{P(3)}$      $\underbrace{\hspace{1.5cm}}_{P(2)} \underbrace{\hspace{1.5cm}}_{P(2)}$      $\underbrace{\hspace{1.5cm}}_{P(3)} \underbrace{\hspace{1.5cm}}_{P(1)}$

118

### 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

$(A_1(A_2(A_3A_4)))$                        $((A_1(A_2A_3))A_4)$   
 $(A_1((A_2A_3)A_4))$      $((A_1A_2)(A_3A_4))$      $((A_1A_2)A_3)A_4$   
 $\underbrace{\hspace{1.5cm}}_{P(1) \times P(3)} + \underbrace{\hspace{1.5cm}}_{P(2) \times P(2)} + \underbrace{\hspace{1.5cm}}_{P(3) \times P(1)}$

$$\rightarrow \sum_{k=1}^{n-1} P(k)P(n-k) \quad 119$$

119

### アルゴリズムのステップ1: 最適解の構造を 特徴づける

120

### $A_i A_{i+1} \dots A_j$ の最適括弧付けが

- $A_k$  と  $A_{k+1}$  で分割すると仮定する

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

121

### $A_i A_{i+1} \dots A_j$ の最適括弧付けが

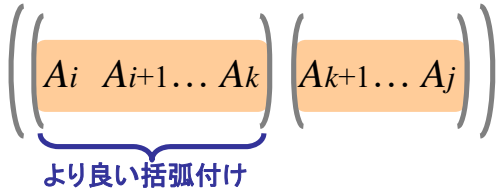
- $A_k$  と  $A_{k+1}$  で分割すると仮定する

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

部分括弧付けは  $A_i \dots A_k$  の最適括弧付けとなっている    部分括弧付けは  $A_{k+1} \dots A_j$  の最適括弧付けとなっている

### なぜならば

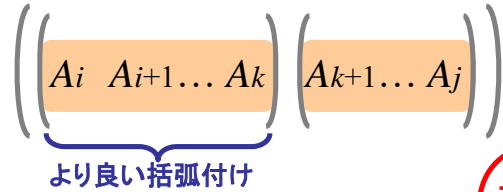
- より良い括弧付けがあればそれを使って  $A_i A_{i+1} \dots A_j$  をより良くできる



123

### なぜならば

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



$A_i A_{i+1} \dots A_j$  が最適括弧付けであることと



124

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

||

**部分問題最適性**

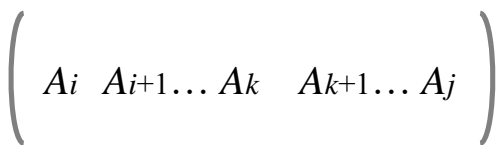
125

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

126

### 最適解の値

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

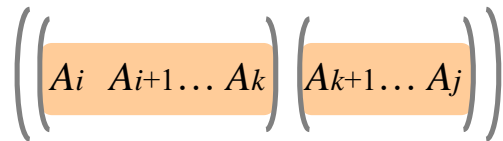


127

### 最適解の値

$m[i, j] =$   $A_i \dots A_j$  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



128

## 最適解の値

$m[i, j] =$   $A_i \dots A_j$  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

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

$$m[i, j] = m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \quad 129$$

## 未知の値 $k$ を求める

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

全ての  $k$  を調べて最良のものを選択する

130

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

131

再帰だと**指数時間**  
が必要

132

## 部分問題の数が少ない

- $1 \leq i \leq k \leq n$  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

133

そこで**ボトムアップ**に

134

**MATRIX-CHAIN-ORDER( $p$ )**

```

 $n \leftarrow \text{length}[p]-1$ 
for  $i \leftarrow 1$  to  $n$ 
  do  $m[i,i] \leftarrow 0$ 
for  $l \leftarrow 2$  to  $n$ 
  do for  $i \leftarrow 1$  to  $n-l+1$ 
    do  $j \leftarrow i+l-1$ 
       $m[i,j] \leftarrow \infty$ 
      for  $k \leftarrow i$  to  $j-1$ 
        do  $q \leftarrow m[i,k] + m[k+1,j] + p_{i-1}p_kp_j$ 
          if  $q < m[i,j]$ 
            then  $m[i,j] \leftarrow q$ 
               $s[i,j] \leftarrow k$ 

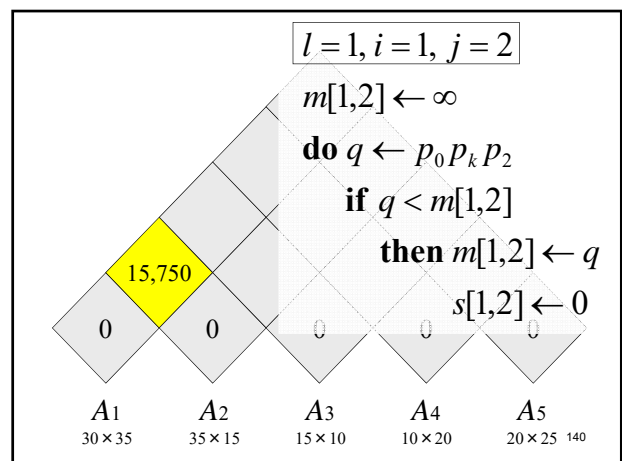
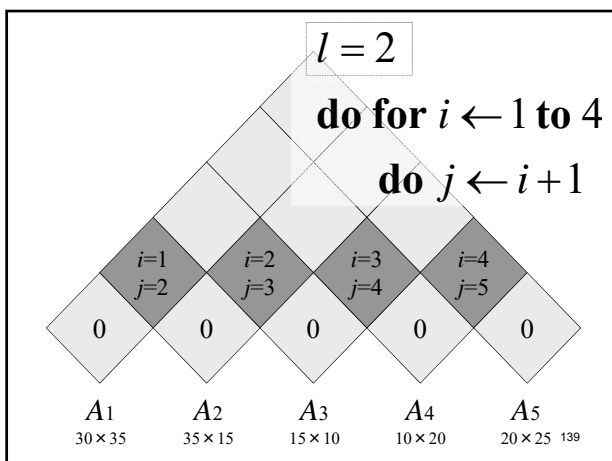
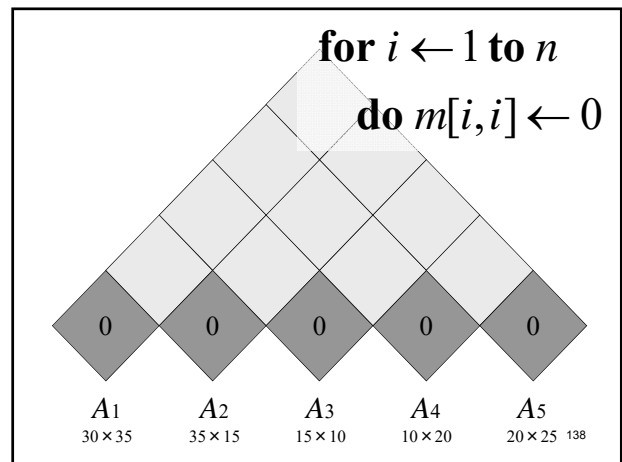
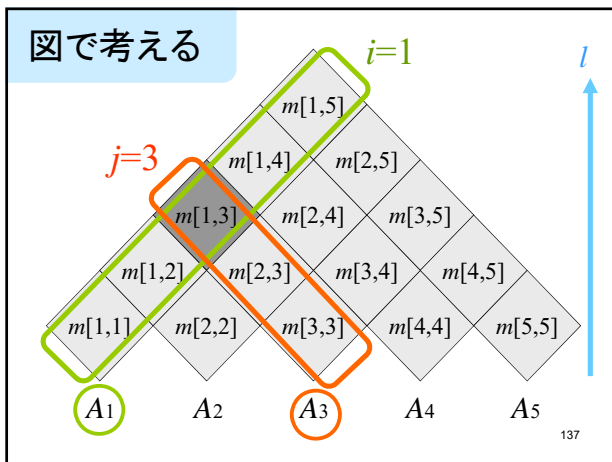
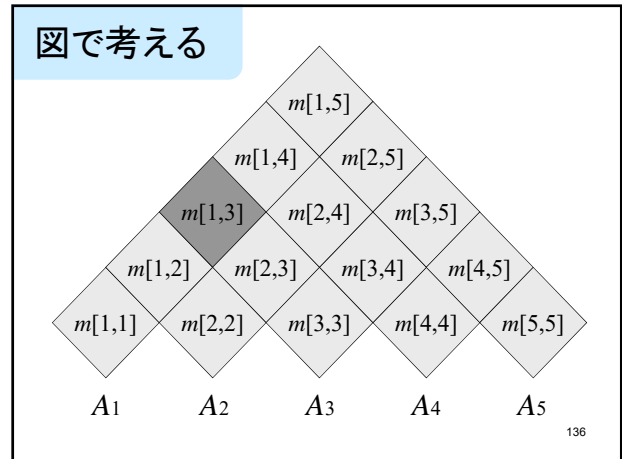
```

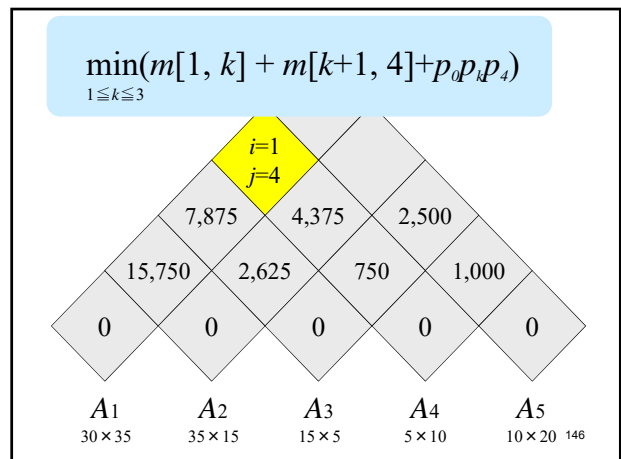
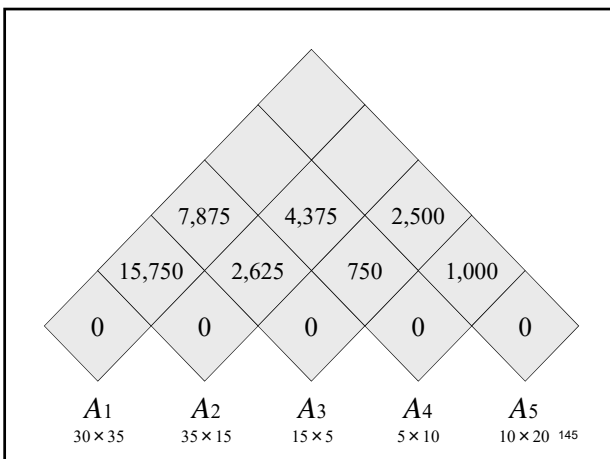
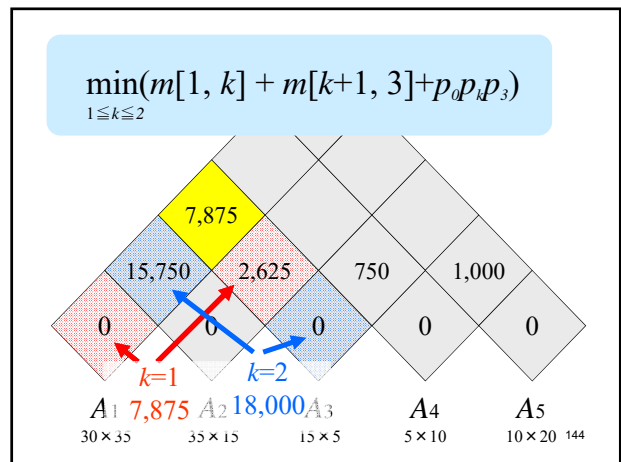
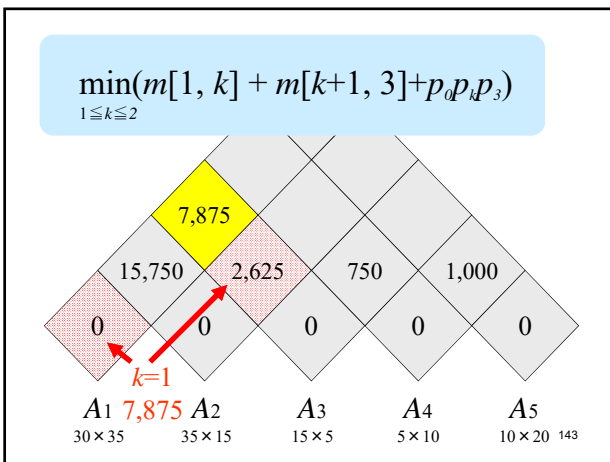
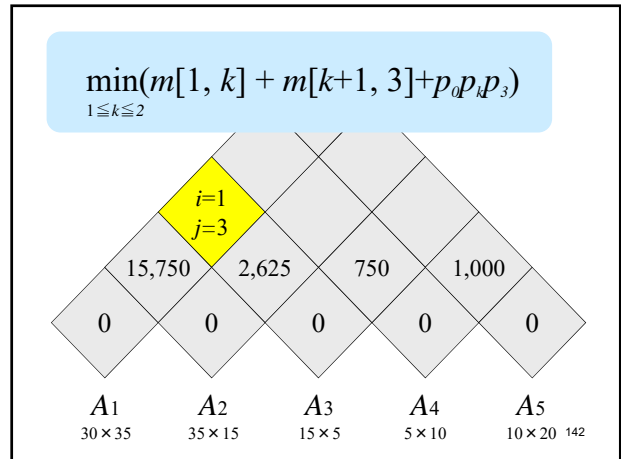
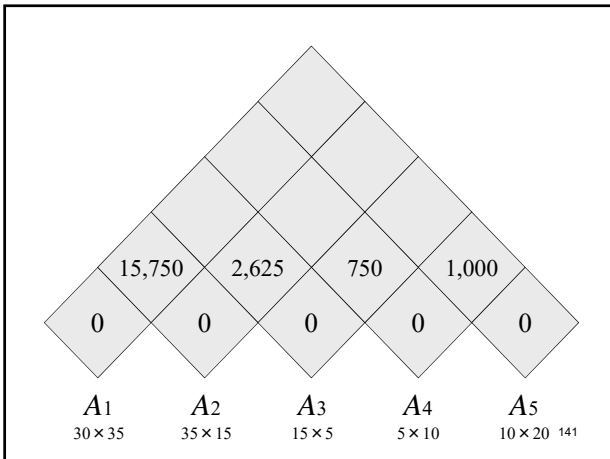
初期値を設定

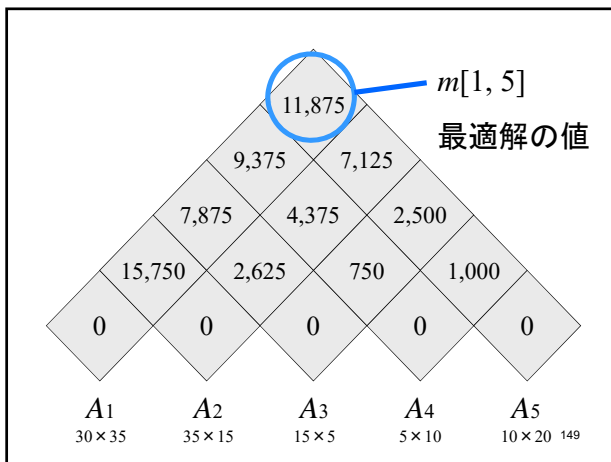
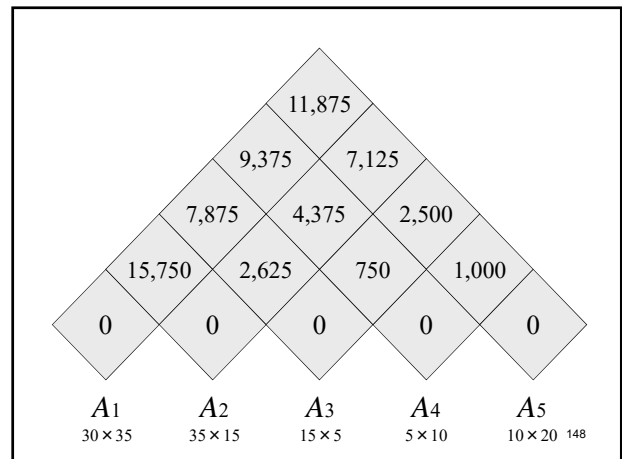
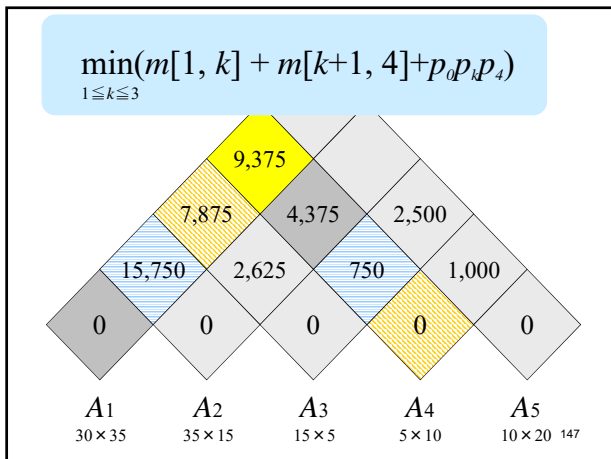
連鎖長が  $l$  のものでループ

$m[i,j]$  を求める

135







### アルゴリズムの評価

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

150

### アルゴリズムのステップ4: 最適解を構成する

151

- $s[i, j]$  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 ")"
    
```

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

### 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

154

### 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

155

### 部分構造最適性を発見する手順

- 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。
- 最適解を導く**選択が与えられている**と仮定する。
- 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。
- 部分問題の解が**最適解**であることを決定する。

156

### 例題で考える

- 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。
- 最適解を導く**選択が与えられている**と仮定する。
- 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。

ステーション $S_{i,j}$ を最速で終了する順路が

- $S_{i,j}$ を経由していた場合

$A_i A_{i+1} \dots A_j$ の最適括弧付けが

- $A_k$ と $A_{k+1}$ で分割すると仮定する

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

157

### 例題で考える

- 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)

ステーション $S_{i,j}$ を最速で終了する順路が

- $S_{i,j}$ を経由していた場合
- $S_{i,j}$ は最速で終了する順路を導ってきた

$A_i A_{i+1} \dots A_j$ の最適括弧付けが

- $A_k$ と $A_{k+1}$ で分割すると仮定する

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

部分括弧付けは $A_i \dots A_k$ の最適括弧付けと $A_{k+1} \dots A_j$ の最適括弧付けとなっている。最適括弧付けとなっている

158

### 例題で考える

- 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)

なぜならば

• もっと速く $S_{1,j}$ に到達する道があるなら、この経路を使って $S_{1,j}$ にもっと速く着ける

**矛盾**

なぜならば

• より良い括弧付けがあればそれを使って $A_1 \dots A_j$ をより良くできる

$(A_1 A_{i+1} \dots A_k) (A_{k+1} \dots A_j)$

より良い括弧付け

$A_1 A_2 \dots A_j$ が最適括弧付けであること **矛盾**

### 部分構造最適性を発見する手順

- 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。
- 最適解を導く**選択が与えられている**と仮定する。
- 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
- 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？

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

#### 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

161

### 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

- $A_1 A_2 \dots A_j$ 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  $A_1 \dots A_k$ と $A_{k+1} \dots A_j$ を生じる

スタートの移動  
かかないとダメ!

部分問題の空間に含まれない

162

### 特徴量

- 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
- 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 ( $A_i \dots A_k$ と $A_{k+1} \dots A_j$ )	$j - i$ ( $k=i, i+1, \dots, j-1$ )

163

### 計算時間

- 部分問題の総数
  - 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 $n$	$O(n^3)$

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
- $S_0 = \{0\}$ ,
- $S_1 = \{0, 3\}$ ,
- $S_2 = \{0, 3, 4, 7\}$ ,
- $S_3 = \{0, 3, 4, 6, 7, 9, 10, 13\}$ ,
- $S_4 = \{0, 3, 4, 6, 7, 9, 10, 11, 13, 14, 16, 17, 20\}$ ,
- $S_5 = \{0, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 25\}$

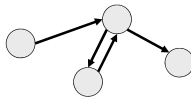
3, 4, 6, 7, 5

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

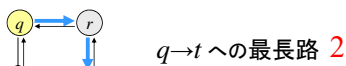
- 部分構造最適性を持つ
- $u \xrightarrow{p_1} v$  を  $u \xrightarrow{p_1} w \xrightarrow{p_2} v$  に分解する
  - $p_1$  は  $u$  から  $w$  への最短路
  - もしそうでないならば, 最短路を  $p_1'$  とすれば  $p_1'$  と  $p_2$  を組み合わせるとより短い経路を作成できる

168

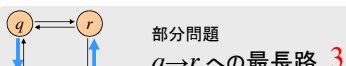
### 重みなし最長単純路問題

- 部分構造最適性を持たない

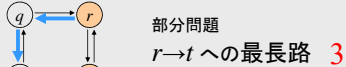
(反例)



$q \rightarrow t$  への最長路 2



部分問題  $q \rightarrow r$  への最長路 3



部分問題  $r \rightarrow t$  への最長路 3

169

### 重みなし最長単純路問題

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

170

### 違いは「独立」

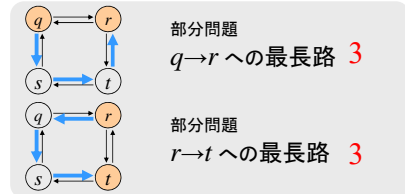
- 最短単純路：独立
- 最長単純路：独立でない

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
  - $u \rightarrow x \rightarrow w \rightarrow x \rightarrow v$
  - $x \rightarrow w \rightarrow x$  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

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

.....

$$\parallel$$

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

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

### 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  $l[j]$  に記録した
  - $l[j]$  がなくても、わずかな計算  $O(n)$  で再構成できる
    - $f_1[j] = f_1[j-1] + a_1$  ならば  $S_{1j}$  を経由
    - $f_1[j] = f_2[j-1] + t_{j-1} + a_1$  ならば  $S_{2j}$  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  $\Theta(j-i)$  が必要
  - $s[i, j]$  があれば  $O(1)$  で計算できる

177

### 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	$O(n^3)$	$O(n^3)$	$2^n$

178

### 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

179