



アルゴリズムとデータ構造III

8回目: 12月06日

グラフ
(動的計画法, DPマッチング)

授業資料 <http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html>



授業評価アンケート 個別質問

■ 9. 創意・工夫

- この授業に関して、教員の創意・工夫が感じられた。

■ 10. コミュニケーション

- この授業において、教員は学生の理解度・反応をみて授業をしていた。

■ 時間割番号：263216

■ 科目名：アルゴリズムとデータ構造III

■ 教員：鈴木良弥



中間試験

- 中間試験日
 - 12月13日(木)
- 範囲
 - スタック
 - 構文解析
 - CKYアルゴリズム
 - チャート法(特徴)
 - 動的計画法
 - ダイクストラ法
 - DPマッチング

補講アンケート

補講: 12/17(月) 5時限 B2-11

12/10

12/17

12/11

12/18

12/12

12/19

12/13

12/20

12/14

12/21

	月	火	水	木	金
1	×		言語論		
2		デジタル回路		×	アーキテクチャII
3	△	×	△	×	基礎解析学
4	構成法演習	×	情報数学	×	
5 16:30~ 18:00	12/17 B2-11				



ハードウェア実験II受講者へ (詳しくはCNSで確認)

- 12月06日(木) 会社見学
 - 見学場所:ファナック株式会社(忍野村)
 - 12:20 新守衛所前に集合
 - 12:30 大学発(観光バス)
 - 14:00~16:00 会社見学
 - 17:30 大学着(の予定)
 - ファナック
 - FAとロボット
 - <http://www.fanuc.co.jp/>

授業の予定(中間試験まで)

10/11	スタック(後置記法で書かれた式の計算)
10/18	文脈自由文法
10/25	構文解析 CKY法
11/01	構文解析 CKY法, チャート法
11/08	構文解析 CKY法, チャート法
11/15	構文解析 チャート法
11/29	グラフ(動的計画法, ダイクストラ法, DPマッチング)
12/06	グラフ(DPマッチング, ビームサーチ, A*アルゴリズム)
12/13	中間試験

授業の予定(中間試験以降)

12/17	全文検索アルゴリズム (simple search, KMP, BM)
12/20	全文検索アルゴリズム (Aho-Corasick)
01/10	テキスト圧縮 暗号 (例: モールス信号, 黄金虫, 踊る人形, ハフマン符号, Zipfの法則), zip
01/17	音声圧縮 ADPCM, MP3
01/24?	音声圧縮 (CELP), 画像圧縮 (JPEG)
01/31?	期末試験



本日のメニュー

- 動的計画法
- DPマッチング
 - アルゴリズム
 - 動作
- 中間試験の範囲の説明

動的計画法

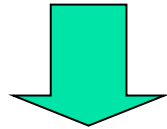
(Dynamic Programming)

- 解くのに時間のかかる問題を、複数の部分問題に分割することで効率的に解くアルゴリズム



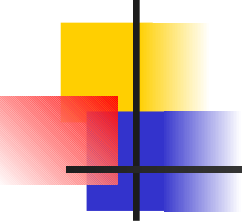
ダイクストラ法

- 動的計画法を最短経路問題に適用



- 最適経路中の部分経路もまた最適経路になっている

ダイクストラ法 アルゴリズム

- 
1. 初期化: スタートノードの値(最小コスト候補)を0, 他のノードの値を無限大に設定
 2. 未確定ノードが無くなるまで以下のループを繰り返す.
 1. 確定中ノードのうち, 最小の値を持つノードを見つけ, 確定ノードとする.
 2. 確定ノードからのエッジに対して「確定ノードまでのコスト + エッジのコスト」を計算し, そのノードの現在値よりも小さければ更新.

ダイクストラ法のアルゴリズム

■ begin

■ for each $x \in V$ do begin

- $cost[x] := w[s, x];$
- $parent[x] := 's';$

costとparentの初期化

■ end

■ $U := V - \{s};$ U (未確定ノード)の初期化

■ while $U \neq \emptyset$ do

■ begin

- U中のmで, $cost[m]$ が最小となる頂点mを選ぶ;

- $U := U - \{m};$

- mから隣接する頂点の集合を D_m とする;

- for each $x \in D_m \cap U$ do

- If $cost[m] + w[m, x] < cost[x]$
- then begin
 - $Cost[x] := cost[m] + w[m, x];$
 - $Parent[x] := m$
- end

集積コストが最も小さいノードmを選んで, $cost[m]$ と $parent[m]$ を確定

頂点mから隣接するノードすべての集合 D_m を求める。
 D_m の要素で且つ未確定ノードである各xについてmを経由してxに至る最短経路のコストを計算し, 現在の $cost[x]$ と比較し, 小さければ更新する

■ end

■ end



ダイクストラ法の特徴

- 最短経路の見つけ方
 - ゴールノードから「どこから来たのか」調べ, さかのぼる.
- マイナスのコストを持つエッジは扱えない.
- 特定のノードからの最短距離およびその経路が全てのノードに対して求まる.

DPマッチング

(例: 文字列の照合)

- 2つの文字列がどのくらい似ているかを調べる.
 - takeda は nakadaiとどのくらい似ているか
- 音声認識にも使える
 - 音声を文字列に変換した後, 登録単語と比較
 - (現在主流の)HMM(Hidden Markov Model)に拡張可能
- DNAの比較にも使える
 - A(アデニン), G(グアニン), C(シトシン), T(チミン)の並び方の比較
 - ACTGAGCATTとCTGGACTACGの比較

DPマッチング(例:文字列の照合) 1/7

takeda と nakadai

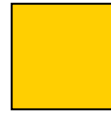
不一致コスト表

文字が一致 → 0
文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

DPマッチング (例: 文字列の照合) 2/7

■ takeda と nakadai



の値を求める

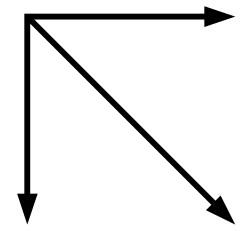
	r	a	k	a	c	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
c	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

1文字ずらしたけれど文字が不一致: $1+3=4$ を加算

移動のペナルティ

横だけ1字ずらす

1



1

縦だけ1字ずらす

0

同時に1文字移動

不一致のペナルティ

文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7						
k	11						
e	15						
d	19						
a	23						

DPマッチング (例: 文字列の照合) 3/7

takeda と nakadai

の値を求める

	r	a	k	a	c	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
c	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ

横だけ1字ずらす

1

1

1

0

同時に1文字移動

不一致のペナルティ

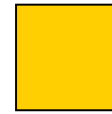
文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11						
e	15						
d	19						
a	23						

DPマッチング (例: 文字列の照合) 4/7

- takeda と nakadai



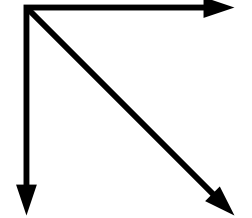
の値を求める

	r	a	k	a	c	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
c	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ

横だけ1字ずらす

1



1

0

縦だけ1字ずらす

同時に1文字移動

不一致のペナルティ

文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15						
d	19						
a	23						

DPマッチング (例: 文字列の照合) 5/7

■ takeda と nakadai

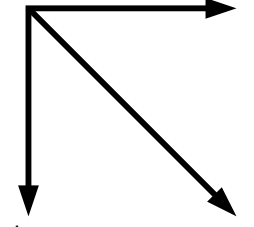
の値を求める

	r	a	k	a	c	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
c	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ

横だけ1字ずらす

1



1

0

縦だけ1字ずらす

同時に1文字移動

不一致のペナルティ

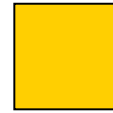
文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19						
a	23						

DPマッチング (例: 文字列の照合) 6/7

- takeda と nakadai



の値を求める

	r	a	k	a	c	a	i
t	∞	∞	∞	∞	∞	∞	∞
a	∞	∞	∞	∞	∞	∞	∞
k	∞	∞	∞	∞	∞	∞	∞
e	∞	∞	∞	∞	∞	∞	∞
c	∞	∞	∞	∞	∞	∞	∞
a	∞	∞	∞	∞	∞	∞	∞

移動のペナルティ

横だけ1字ずらす

1

1

0

同時に1文字
移動

不一致のペナルティ

文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19	15	11	10	6	10	14
a	23						



DPマッチング (例: 文字列の照合) 7/7

- takeda と nakadai



の値を求める

	r	a	k	a	c	a	i
t	∞	∞	∞	∞	∞	∞	∞
a	∞	∞	∞	∞	∞	∞	∞
k	∞	∞	∞	∞	∞	∞	∞
e	∞	∞	∞	∞	∞	∞	∞
c	∞	∞	∞	∞	∞	∞	∞
a	∞	∞	∞	∞	∞	∞	∞

移動のペナルティ

横だけ1字ずらす

1

1

0

同時に1文字

移動

不一致のペナルティ

文字が一致 → 0

文字が不一致 → 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19	15	11	10	6	10	14
a	23	16	18	11	10	6	10





DPマッチングの応用

- DPマッチングの探索空間を制限し、探索時間を削減する方法
 - ビームサーチ(最適解は保証されない)
 - A*アルゴリズム(最適解は保証される)
- HMM(隠れマルコフモデル)とビタビアルゴリズム
 - 音声認識手法の主流