



情報アクセス論 第3回

「文書の収集・変換」

情報理工学部
前田 亮

● ● ● | 第2回小テストの解説(1/2)

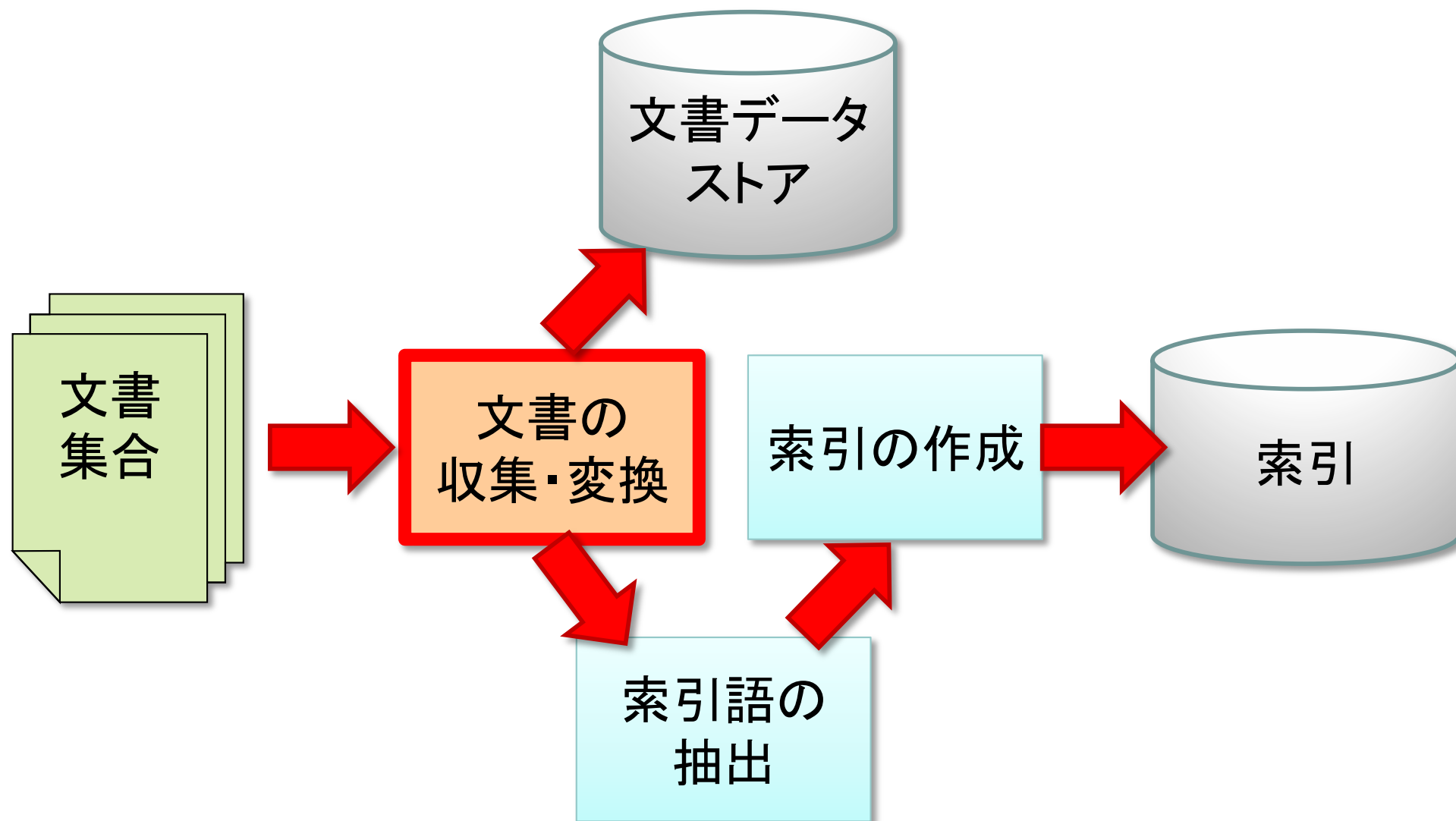
- 問1: 以下の文章のうち誤っているものを選べ
- Web検索エンジンにおける文書の収集は, Webページ間のハイパーリンクを辿って収集するクローラと呼ばれるソフトウェアを用いる.
- 個人のPC内に保存されている文書を検索するデスクトップ検索(Windows SearchやmacOSのSpotlight)では, 索引を作成しない.
- 助詞や代名詞など, 検索にあまり用いられない語を判定するために, 形態素解析における品詞の情報をを用いることができる.
- 情報検索で用いる転置索引では, ある索引語がどの文書に含まれるかという情報が格納される.

● ● ● | 第2回小テストの解説(2/2)

- 問2: 以下の文章のうち誤っているものを選べ.
- 問合せ処理のうち, 問合せに関連する語を自動的に追加する機能を問合せ拡張と呼ぶ.
- 問合せ処理のうち, 最初の検索結果を基に, より良い検索結果を得るために問合せを修正する機能を適合性フィードバックと呼ぶ.
- 検索結果のランキングのために, 文書中での索引語の出現頻度の情報を用いることができる.
- フレーズ検索のために, 文書中での索引語の出現頻度の情報を用いることができる



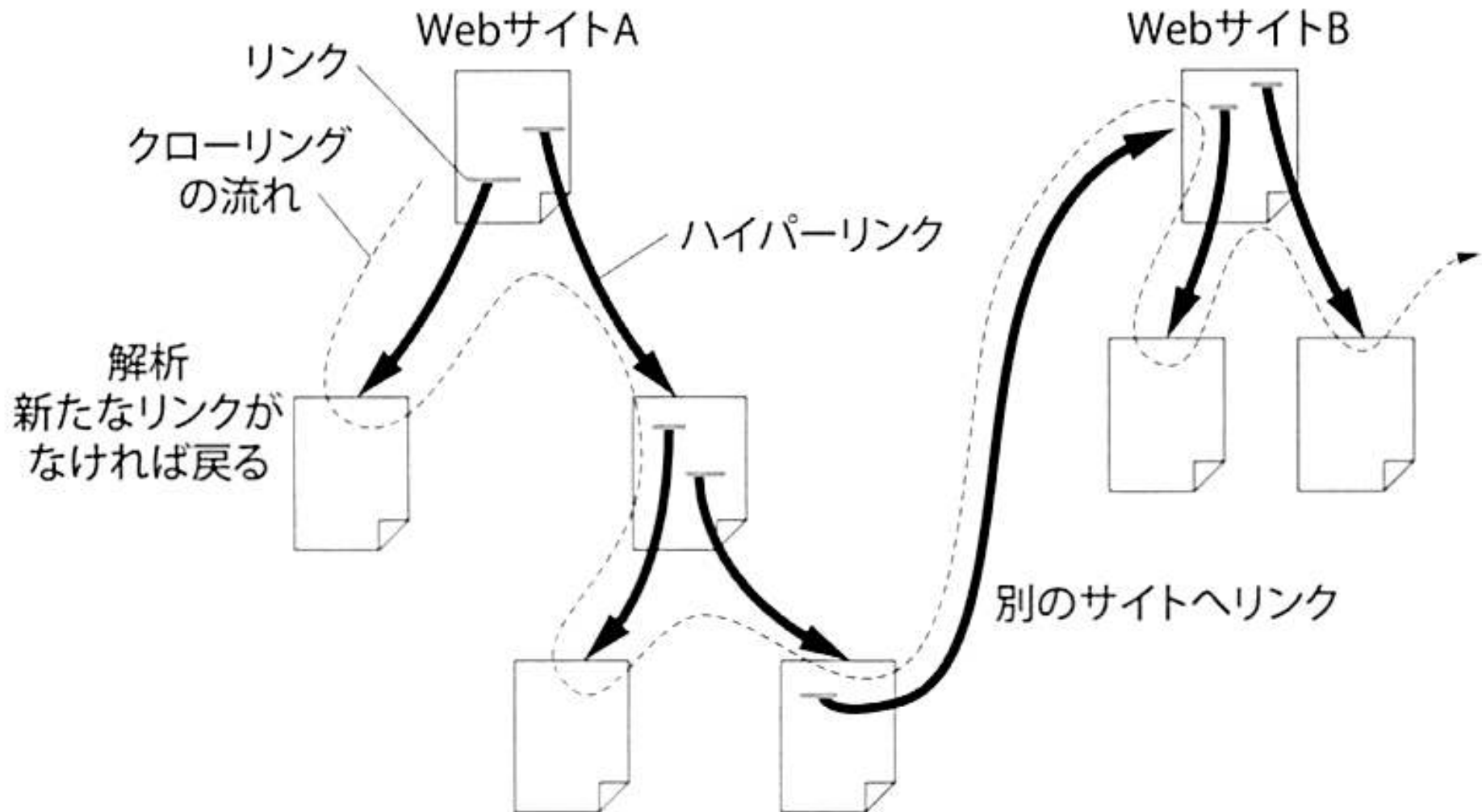
索引付けのプロセス(再掲)

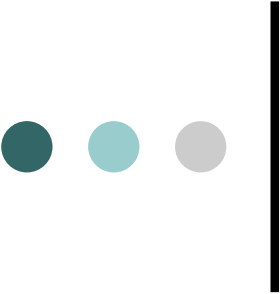


● ● ● | Webクローラ

- Web文書を自動的にリンクをたどりながら収集
 - 別名:「Webロボット」「ボット」「スパイダー」
 - 具体例: Googlebot, Bingbot, Yahoo! Slurp
- Webクローラの課題
 - Webはすでに巨大である
 - 数百億ページ以上あるといわれている
 - Webは絶えず成長している
 - 一年で約25%程度の増加
 - Webページは絶えず変化している
 - Webページの平均寿命は1～3カ月程度

WebクローラによるWeb文書 収集の流れ





単純なクローラのアルゴリズム

1. **初期URL集合**を与えて開始
2. 初期URL集合を**URLキュー**(待ち行列)に追加
3. URLキューからWebページを取得
4. 取得されたページを解析し, リンクURLを抽出
5. 抽出したURLをURLキューに追加
6. 新たなURLがなくなるまで, 上記3~5の手順を繰り返す



Webクロールリングの注意点

- Webクローラでは、リクエストに対する応答の待ち時間に多くの時間が費やされる
- この問題に対処するため、スレッドを用いて同時に多くのページを取得
- 特定のサイトにリクエストが集中すると、負荷をかける可能性がある
- この問題を避けるため、「礼儀正しい」アクセスを行う
 - 同一のWebサーバに対するアクセスには一定の間隔を空ける

クローリングの制御

- 「robots.txt」という名前のファイルをWebサーバのルートに置くことで、どのファイルの取得を許可するかをサーバ管理者が制御できる

User-agent: *

すべての種類のクローラを対象

Disallow: /local/

Allow: /local/public/

/local/以下はアクセスを拒否、
ただし/local/public/以下は許可

User-agent: MyCrawler

Disallow:

MyCrawlerに対してはすべての
コンテンツへのアクセスを許可



ディレクトリの走査

- 個人PCやファイルサーバのような、ローカルのファイルシステム上にある文書
 - Web文書のようにリンクをたどって収集する必要がない
- 検索システムでは、ファイルシステムから対象の文書を収集する機能を備えている



文書の加工

- 索引付けを行うには、まず対象の文書からテキストを抽出する必要
 - Web文書では、HTMLのタグを取り除く
 - Office文書（Word, PowerPointなど）やPDFなどでは、テキスト部分を抽出するためのソフトウェアが必要
- テキストの文字コードが統一されていない場合、文字コードの変換が必要



文字コード

- コンピュータ上で、文書は**文字コード**を用いて記述される
- 最も単純な情報検索は、問合せと文書中に現れる語との**文字コードの一致**
 - 問合せと文書の文字コードが同じでなければ検索できない
- 環境によって使用する文字コードが異なる
- Webのような**多言語の文書が混在**する環境では、使われる文字コードも多様である
 - 文字コードを知らなければシステムは作れない



用語の定義

- **文字集合 (character set)**
 - テキスト情報を表現するのに用いる要素(文字)の集合
- **符号化文字集合 (coded character set)**
 - 各文字に符号値が割り振られた文字集合
- **文字符号化方式 (character encoding scheme)**
 - (一つあるいはそれ以上の)文字集合からデータ(文書)を表現するビット列へのマッピング
- **文字エンコーディング (character encoding)**
 - (一つあるいはそれ以上の)符号化文字集合と文字符号化方式の組み合わせ
 - 「文字エンコーディング」「エンコード」などと呼ばれるものは、通常これを指す



文字コードの概念

符号を割り当てた文字の集合
(アルファベット, 漢字など)

複数の文字集合に符号
を割り当てる方式

符号化文字集合と文字
符号化方式の組合せ

符号化文字集合

A B
ASCII
C

高 崎
JIS X 0213

漢
JIS X 0208
字

文字符号化方式

シフトJIS

EUC

ISO 2022

文字エンコーディング

シフトJIS

EUC-JP

ISO-2022-JP



符号化文字集合

ASCII (1967)

- American Standard Code for Information Interchange
- 「アスキー」と発音
- アルファベット+数字+記号を7ビットで表現
- 34の制御文字
 - 主にテレタイプや紙テープ用
- 94の図形文字(形のある文字)
- 現在に至るコンピュータの文字コードの基盤

	00	01	02	03	04	05	06	07
00			SP	0	@	P	`	p
01			!	1	A	Q	a	q
02			"	2	B	R	b	r
03			#	3	C	S	c	s
04			\$	4	D	T	d	t
05			%	5	E	U	e	u
06	制御文字		&	6	F	V	f	v
07			'	7	G	W	g	w
08			(8	H	X	h	x
09)	9	I	Y	i	y
10			*	:	J	Z	j	z
11			+	;	K	[k	{
12			,	<	L	\	l	
13			-	=	M]	m	}
14			.	>	N	^	n	~
15			/	?	O	_	o	DEL



ISO 8859

- ASCIIを基にした国際標準規格であるISO 646に、ヨーロッパ言語のアクセント付き文字を追加
- ISO 8859-1～16の16種類が存在する
 - キリル文字
 - アラビア文字
 - ヘブライ文字
 - タイ文字
 - etc.

ISO 8859-1 (西ヨーロッパ言語用)

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00			SP	0	@	P	`	p			NBS P	°	À	Đ	à	đ
01			!	1	A	Q	a	q			ı	±	Á	Ñ	á	ñ
02			”	2	B	R	b	r			¢	²	Â	Ò	â	ò
03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
04			\$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			¥	µ	Å	Ö	å	ö
06			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
07			‘	7	G	W	g	w			§	·	Ç	×	ç	÷
08			(8	H	X	h	x			¨	,	È	Ø	è	ø
09)	9	I	Y	i	y			©	¹	É	Ù	é	ù
10			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
11			+	;	K	[k	{			«	»	Ë	Û	ë	û
12			,	<	L	\	l				¬	¼	Ì	Ü	ì	ü
13			-	=	M]	m	}			SHY	½	Í	Ý	í	ý
14			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
15			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

JIS X 0201 (1969)

- ISO 646の日本版
- ASCIIの「\」が「¥」に,
「~」が「 」に置き換えられている
- 0xA1~0xDFにカタカナ
(いわゆる半角カナ)と記
号を追加

	ASCII							JIS X 0201								
	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00				0	@	P	`	p				一	タ	ミ		
01			!	1	A	Q	a	q				。	ア	チ	ム	
02			”	2	B	R	b	r				「	イ	ツ	メ	
03			#	3	C	S	c	s				」	ウ	テ	モ	
04			\$	4	D	T	d	t				、	エ	ト	ヤ	
05			%	5	E	U	e	u				・	オ	ナ	ユ	
06			&	6	F	V	f	v				ヲ	カ	ニ	ヨ	
07			‘	7	G	W	g	w				ァ	キ	ヌ	ラ	
08			(8	H	X	h	x				ィ	ク	ネ	リ	
09)	9	I	Y	i	y				ウ	ケ	ノ	ル	
10			*	:	J	Z	j	z				ェ	コ	ハ	レ	
11			+	;	K	[k	{				ォ	サ	ヒ	ロ	
12			,	<	L	¥	l					ャ	シ	フ	ワ	
13			-	=	M]	m	}				ュ	ス	ヘ	ン	
14			.	>	N	^	n	—				ョ	セ	ホ	ゝ	
15			/	?	O	_	o					ッ	ソ	マ	。°	

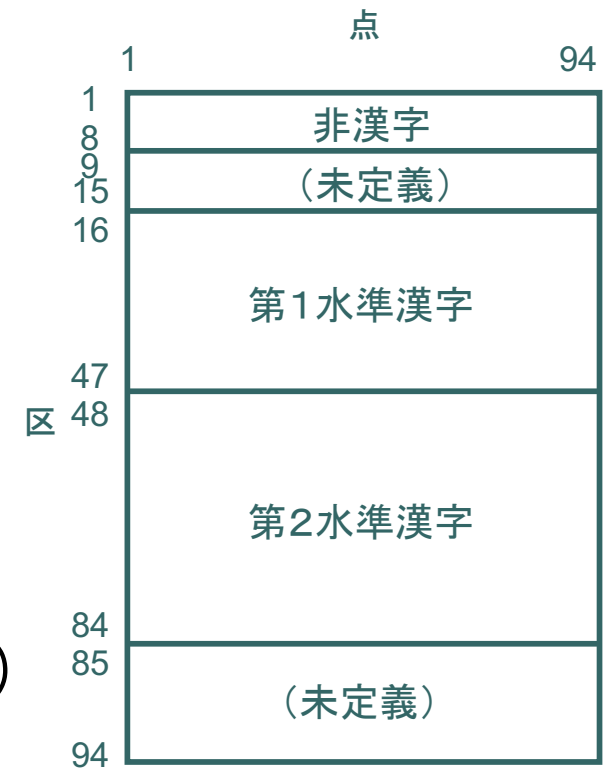
JIS X 0208 (1978)

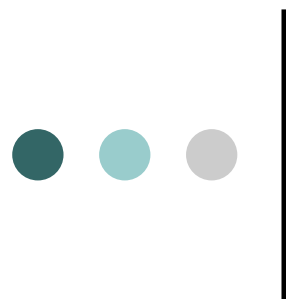
- 世界初の複数バイト符号化文字集合
 - 2バイトで1文字を表す
- 6,879文字を規定 (94区 × 94点 = 8,836)

- 問題点

- JIS X 0201のアルファベット, 記号, カタカナとは別のコードポイント(いわゆる全角文字)を割当て
 - ・ 同じ文字なのに, 2種類の符号・形が存在 (例: 「A」「A」)
 - ・ 悪名高いシフトJISを生む要因
- 1983年の改定時に, 一部字形の変更と入れ換え
 - ・ 互換性が失われ, 文字化けの原因

- 最新版はJIS X 0208 (1997)





JIS X 0213

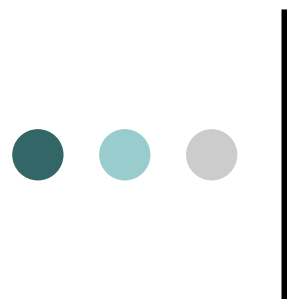
- JIS X 0208の第一・第二水準漢字に、第三・第四水準漢字(拡張漢字)を加えた文字集合
 - 4,354文字が追加され、計11,233文字
- JIS X 0208の区・点に加えて、面を追加
 - 1面は第一～第三水準漢字、2面は第四水準漢字
- Windows Vista以降で標準採用
- 一部の文字(168字)の字体を変更
 - 互換性の問題が発生

「葛城市」？「葛城市」？

- 奈良県葛城市は、「葛」を正式な字としている
- しかし、この文字の正字は「葛」である
- 葛城市が誕生する2003年当時普及していたJIS X 0208では「葛」が採用されていた
 - 住民の利便性を考え、市名の正式な字に採用
- 2004年に制定されたJIS X 0213では、正字の「葛」に変更された
 - 新しいPCでは、正しい市名が表示できない！



文字エンコーディング



シフトJIS

- 最も普及している日本語の符号系
 - Windows, Mac OS, 一部のUNIX環境など
- 1982年にアスキー・マイクロソフト・三菱電機などが独自に制定(国際規格に非準拠)
 - その後JIS X 0208-1997の付属書で正式に規定
- JIS X 0201の未定義領域にJIS X 0208(漢字)を移動(シフト)している
 - 当時すでに普及していたJIS X 0201との互換性を図るため



シフトJIS

○ 1バイト目は右の図の通り

- 1バイト目が81~9FもしくはE0~FFであれば2バイト文字, それ以外は1バイト文字
- 2バイト文字の2バイト目は40~FCを使用

2バイト目の符号領域

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00			SP	0	@	P	`	p				一	タ	ミ		
01			!	1	A	Q	a	q				。ア	チ	ム		
02			"	2	B	R	b	r				「イ	ツ	メ		
03			#	3	C	S	c	s				」ウ	テ	モ		
04			\$	4	D	T	d	t				、エ	ト	ヤ		
05			%	5	E	U	e	u				・オ	ナ	ユ		
06			&	6	F	V	f	v				ヲカ	ニ	ヨ		
07			'	7	G	W	g	w				アキ	ヌ	ラ		
08			(8	H	X	h	x				イク	ネ	リ		
09)	9	I	Y	i	y				ウケ	ノ	ル		
10			*	:	J	Z	j	z				エコ	ハ	レ		
11			+	;	K	[k	{				オサ	ヒ	ロ		
12			,	<	L	¥	l					ヤシ	フ	ワ		
13			-	=	M]	m	}				ユス	ヘ	ン		
14			.	>	N	^	n	_				ヨセ	ホ	ゝ		
15			/	?	O		o	DEL				ッソ	マ	。		

制御文字

JIS X 0208 の 1 ~ 62 区

JIS X 0208 の 63 ~ 94 区

機種依存文字



シフトJISの問題点

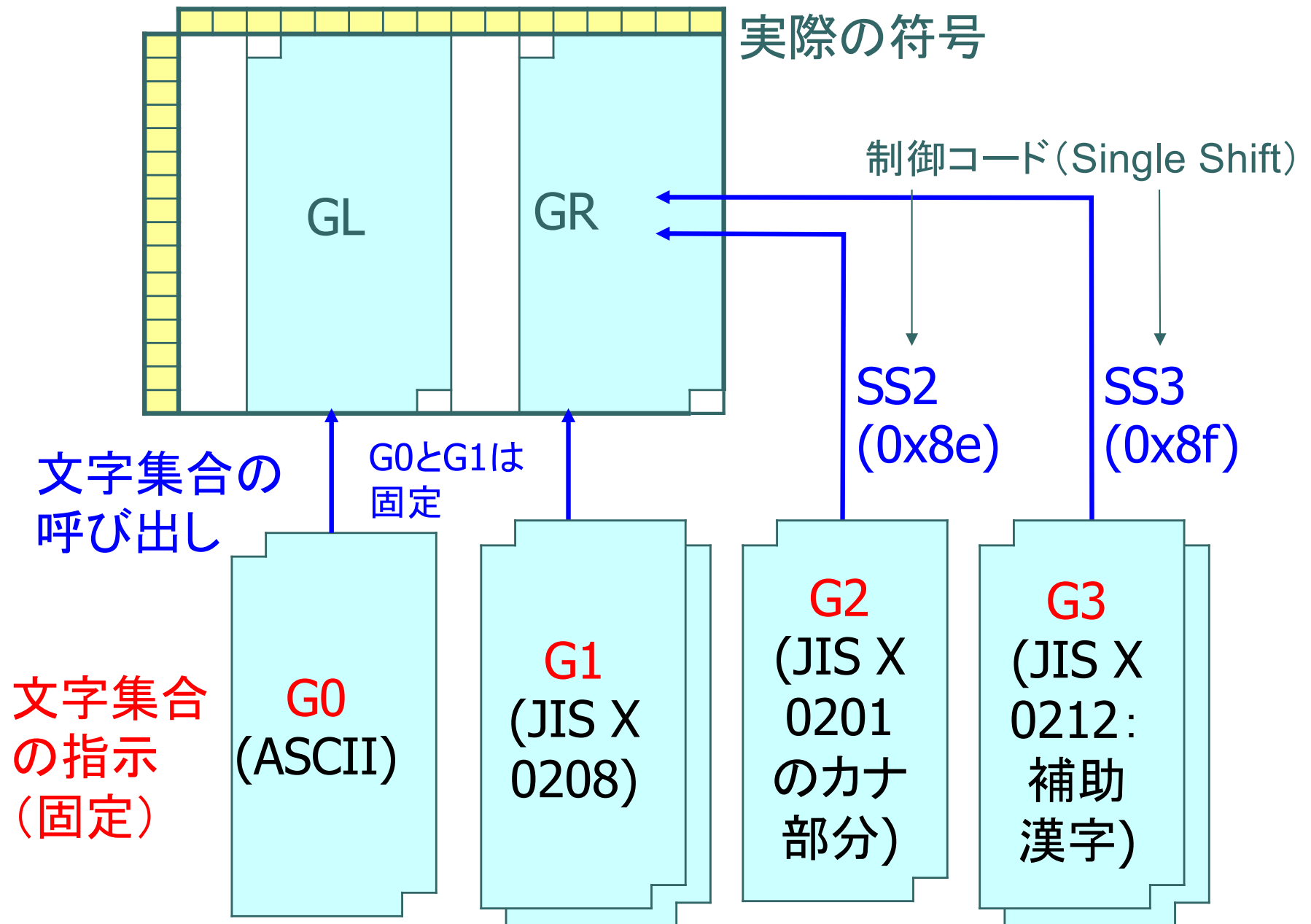
- 2バイト目の一部がASCIIのコード領域(40～7F)と重なる
 - 2バイト目の5C(\)の文字は, コンピュータで特殊な文字として扱われ, 文字化けの原因に
 - 「—(ダッシュ)」「ソ」「十(漢数字の10)」「表」...
- 補助漢字が使用できない
 - JIS X 0213以前に作られた, 人名・地名などの漢字を含む文字集合

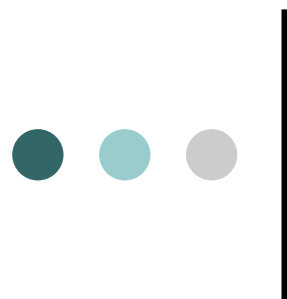


EUC-JP

- 主にUNIX環境で用いられる符号化方法
 - EUCはExtended Unix Codeの略
- ISO 2022に準拠
 - ISO 2022: 複数の符号化文字集合を切り換えて用いる方法を規定
- 他にEUC-CN(中国語), EUC-KR(韓国語)がある

EUC-JPの仕組み

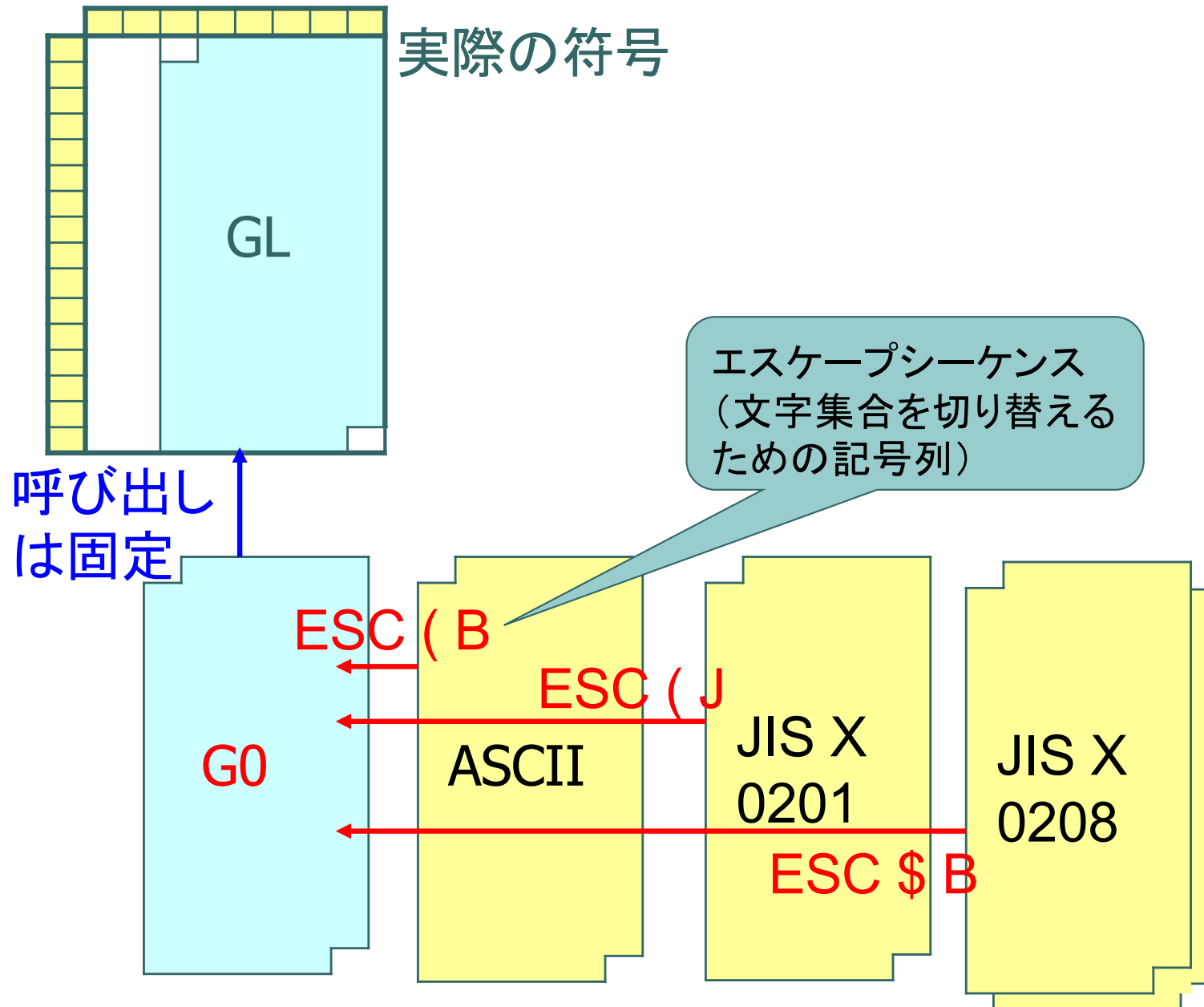


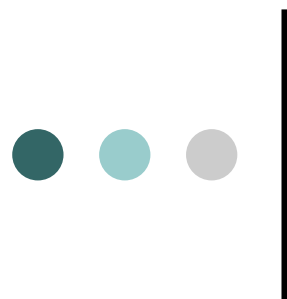


ISO-2022-JP

- 日本語の電子メールで使われている符号系
 - 俗に「**JISコード**」と呼ばれている
- 7ビット符号系
 - 最上位ビットを落とす伝送路のため
- 基本的にISO 2022のサブセット
 - 一部非準拠の部分あり
- いわゆる半角カナ(JIS X 0201)は使えない

ISO-2022-JPの仕組み





Unicode

- 多言語を一つの文字集合で統一的に扱うことを目的に、主に米国のIT企業が提唱
 - 既存の文字集合の規格をすべて収録
- 問題点
 - 日・中・韓の漢字は、似た文字は同一とみなす（**アジア軽視**）
 - 原規格で区別されているものは、互換性のためすべて区別（**欧米重視**）
 - ダッシュ(ー)・ハイフン(-)が15種類も存在

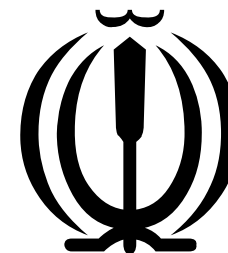
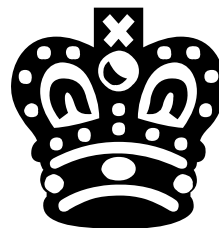
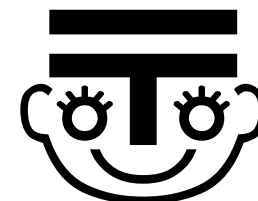
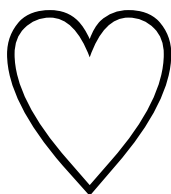
CJK統合漢字の例

Row/Cell Hex code	C G-Hanzi-T	J Kanji	K Hanja
154/168 9AA8	骨 0-3947 0-2539	骨 1-586C 1-5676	骨 0-397C 0-2592
137/210 89D2	角 0-3D47 0-2939	角 1-4B45 1-4337	角 0-3351 0-1949
082/003 5203	刃 0-4850 0-4048	刃 1-4443 1-3635	刃 0-3F4F 0-3147



Unicodeの絵文字の例①

- 従来の文字コードに存在する文字



Unicodeの絵文字の例②

○ 携帯電話・スマートフォンの絵文字



UnicodeのNFC/NFD問題 (1/2)

- Unicodeでは、複数の文字を合成して一つの文字として表現できる
 - 「は」+「◌゜」→「は」, 「e」+「´」→「é», etc.
- 検索・整列などのための正規化(文字の合成・分解)方法が複数ある
 - **NFC** (Normalization Form Canonical Composition)
 - 合成済み文字で表現: 「は」→「は」(U+3071)
 - **NFD** (Normalization Form Canonical Decomposition)
 - 分解された文字列で表現:
「は」→「は」(U+306F) + 「◌゜」(U+309A)



UnicodeのNFC/NFD問題 (2/2)

- テキストの正規化の方法がNFCとNFDで異なると検索できない場合がある
 - 「**ブログ**」
 - NFCで正規化: 30D6 30ED 30B0
 - NFDで正規化: 30D5 3099 30ED 30AF 3099
- ファイルシステム, アプリケーションなどで正規化方法が異なると問題になる
 - ファイルシステムでは, Windows/LinuxはNFC, macOSのHFS+はNFD, APFSはNFC
 - Windowsで作成したPDFをmacOSでコピーペーストすると, 濁点・半濁点が分解される場合がある



UTF-8 (1)

- Unicodeの文字エンコーディングの一つ
- ASCIIと互換性を持つ可変長エンコーディング
 - ASCII部分は1バイト(ASCII互換)
 - ISO 8859部分は2バイト
 - 漢字などは3バイト
 - 最大6バイト
- 可変長だが、文字の境界を明確に判定できる
(2バイト目以降は必ず上位2ビットが「10」)



UTF-8 (2)

- 各文字の1バイト目の上位ビットで、文字に用いられるバイト数が決まる
 - 1バイト文字: 最上位ビットが「0」
 - 2バイト文字: 上位3ビットが「110」
 - 3バイト文字: 上位4ビットが「1110」
 - 4バイト文字: 上位5ビットが「11110」
 - 5バイト文字: 上位6ビットが「111110」
 - 6バイト文字: 上位7ビットが「1111110」



UTF-8による符号化の例

元の文字列

京都

文字

京

都

符号位置(16進数)

4EAC

90FD

符号位置(2進数)

0100 1110 1010 1100

1001 0000 1111 1101

UTF-8(2進数)

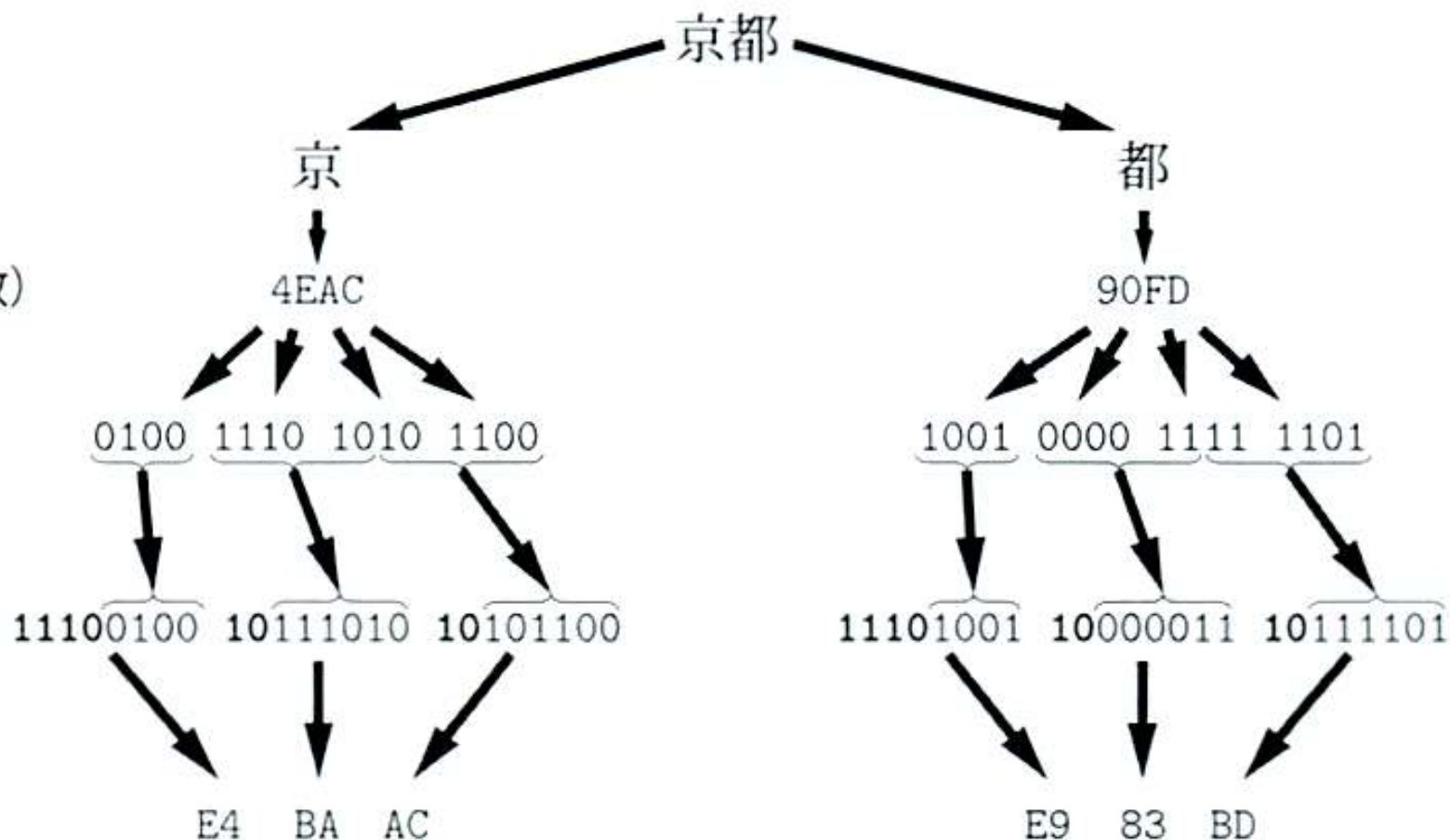
11100100 10111010 10101100

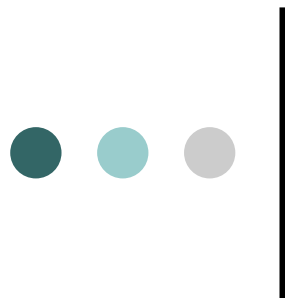
11101001 10000011 10111101

UTF-8(16進数)

E4 BA AC

E9 83 BD





まとめ

- 索引付けの処理の第1段階である文書の収集・加工の手法について説明した
- Web文書を収集する手法として, Webクローラについて説明
- 文字コードに関わる概念と実例を説明
- 文字エンコーディングの正しい理解は, 情報検索システムに限らず日本語を扱う情報システムにおいて非常に重要