ローリングハッシュと Suffix Array

北海道大学 情報知識ネットワーク研究室
情報理工学専攻 修士 1年
栗田 和宏

2015/06

部分文字列検索

- テキストT：検索する文字列
  テキスト長をnとする．

- パターンP：見つけたい文字列
  パターン長をmとする．

- 自明な方法：O(nm)

ナイーブな部分文字列検索

- テキスト：abracadabra
  パターン：abr

abracadabra
abr

ナイーブな部分文字列検索

- テキスト：abracadabra
  パターン：abr

abracadabra
abr

ナイーブな部分文字列検索

- テキスト：abracadabra
  パターン：abr

abracadabra
abr

ナイーブな部分文字列検索

- テキスト：abracadabra
  パターン：abr

abracadabra
abr

ナイーブな部分文字列検索

- テキスト：abracadabra
  パターン：abr

abracadabra
abr

この方法では部分文字列検索に
O(nm)時間かかる

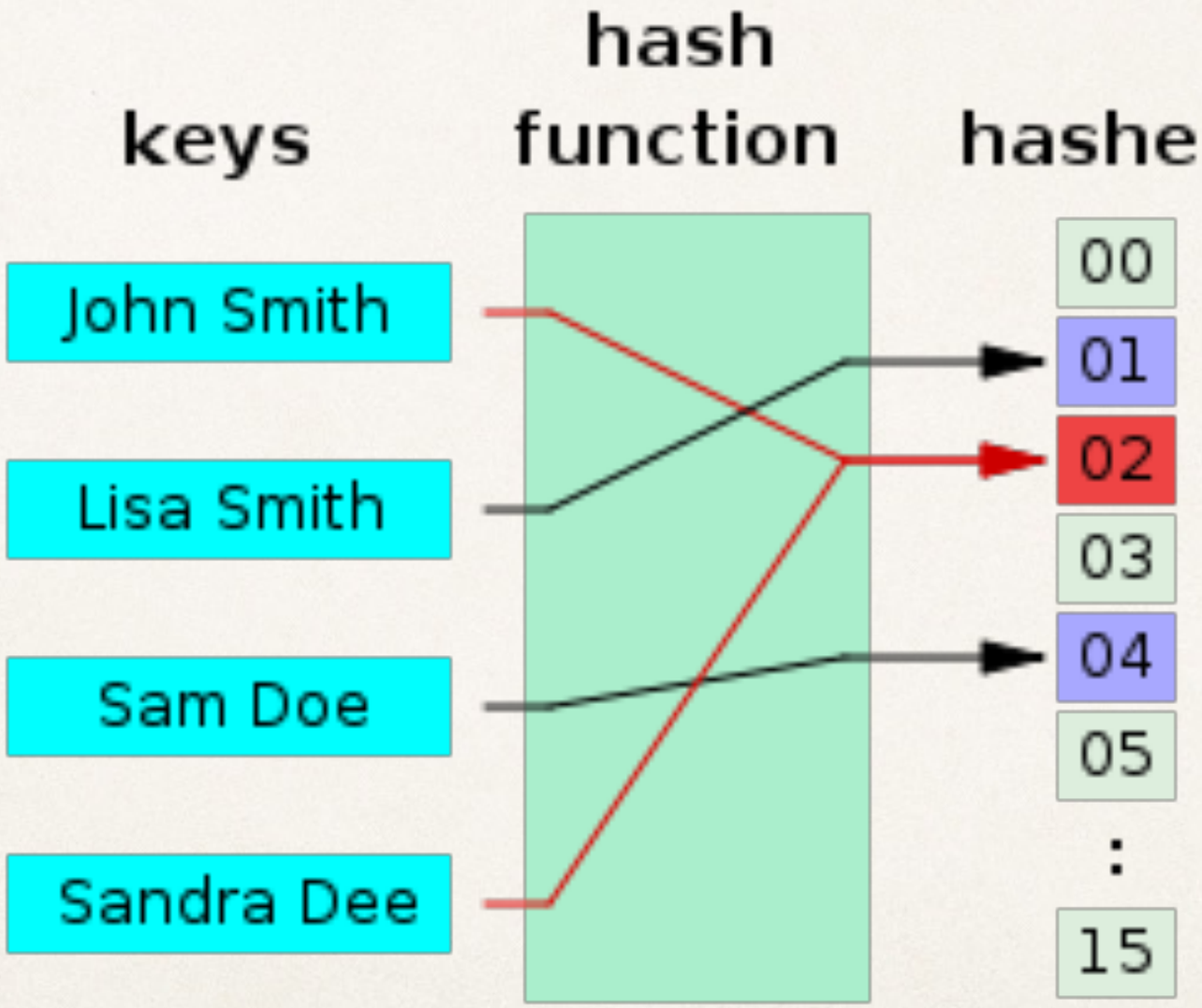
高速な文字列検索

- ローリングハッシュを使おう！
- 文字列を値にするハッシュ関数を用いて値が一致した時に文字列が一致したとする．

高速な文字列検索

- ローリングハッシュを使おう！
- 文字列を値にするハッシュ関数を用いて値が一致した時に文字列が一致したとする．
- その前にハッシュとは何か知っていますか？

ハッシュとは？



https://ja.wikipedia.org/wiki/%E3%83%8F%E3%83%83%E3%82%B7%E3%83%A5%E9%96%A2%E6%95%B0

ローリングハッシュ

- 文字1文字を値とし，その値と基数の積の総和をハッシュ値とするハッシュ関数を使う．

ローリングハッシュ

- 文字1文字を値とし，その値と基数の積の総和をハッシュ値とするハッシュ関数を使う．

$$abr = 1*10^2 + 2*10^1 + 5*10^0$$
$$= 125$$

a = 1, b = 2, r = 5, 基数：10

ハッシュ値の計算

- 愚直にハッシュ値を毎回計算するとハッシュ値の計算にO(m)時間かかり，結局O(nm)時間かかってしまう．

ハッシュ値の計算

- 愚直にハッシュ値を毎回計算するとハッシュ値の計算にO(m)時間かかり，結局O(nm)時間かかってしまう．
- なので，ローリングしながらハッシュ値を計算することで毎回のハッシュ値計算を0(1)時間にする．

ハッシュ値の計算

abra = 125

abracadabra

125

ハッシュ値の計算

abra = 125

abracadabra

251

ハッシュ値の計算

abra = 125

abracadabra

513

ハッシュ値の計算

abra = 125

abracadabra

513

1つ前のハッシュ値に基数をかけ，新しい文字に対応する値を足し，先頭文字に対応する値を引く．

接尾辞配列（Suffix array）

- 文字列のある場所から末尾までの文字列を**接尾辞**という．
- 1つの文字列のすべての接尾辞を辞書順にソートしたものを接尾辞配列という．
- 自明に$O(n^2 \log n)$で作れる．

接尾辞配列の例

- abracadabraの接尾辞配列

接尾辞配列で何ができるか？

- 文字列の検索（さっきと同じ問題）がO(m log n)で解ける．場合によってはローリングハッシュよりも高速に解ける．
- LCP Array（Longest Common Prefix Array）との組み合わせで繰り返し出現する部分文字列を見つけることができる．（他にもいろいろできるっぽい）

記号の説明

- 接尾辞配列の$O(n(\log n)^2)$時間構築アルゴリズムを紹介する.
- S[i, k]：文字列Sのi文字目からk文字の部分文字列
- rank_k(i)：S[i, k]がすべてのk文字の部分文字列をソートした時に何番目に小さいか

接尾辞配列の構築

- 基本的なアイディアはダブリングである.
- $rank_k$(i)と$rank_{2k}$(i)のペアをソートすることでi文字目からi + 2k文字目まではソートされる
- rankのソートをlog n回行うことでO(n $(\log n)^2$)で構築可能

接尾辞配列の構築

sa[i]	S[sa[i], 2]	S[sa[i], 4]	$rank_2$(sa[i])	$rank_2$(sa[i] + 2)
11	emp	emp	0	-1
10	a	a	1	-1
0	ab	abra	2	8
7	ab	abra	2	8
3	ac	acad	3	4
5	ad	adab	4	2
1	br	bra	5	1

最新の接尾辞配列の構築法

- SA-ISという手法で線形時間構築が可能！
  （俺には理解できませんでした．）