Problem 62.

Nodes → / # Edges ↓	m	n	o	p	q	r	s	t	u	v	w	x	y	z	remaining count
	0	0	2	0	2	3	2	2	2	2	1	2	1	2	14
	/	0	2	0	1	2	2	2	2	2	1	1	1	2	13
	/	/	1	0	0	2	2	2	1	2	1	1	1	2	12
	/	/	0	/	0	2	1	2	1	2	1	1	1	1	11
	/	/	/	/	0	1	0	2	1	1	1	1	1	1	10
	/	/	/	/	/	1	0	1	1	1	1	1	1	1	9
	/	/	/	/	/	0	/	1	1	1	1	1	1	1	8
	/	/	/	/	/	/	/	1	0	1	1	1	0	1	7
	/	/	/	/	/	/	/	/	0	/	1	1	1	0	6
	/	/	/	/	/	/	/	/	/	/	1	1	1	0	5
	/	/	/	/	/	/	/	/	/	/	0	1	1	/	4
	/	/	/	/	/	/	/	/	/	/	/	0	0	/	3
	/	/	/	/	/	/	/	/	/	/	/	/	0	/	2
	/	/	/	/	/	/	/	/	/	/	/	/	/	0	1
	/	/	/	/	/	/	/	/	/	/	/	/	/	/	0

/ = deleted-edge (0)

So, we can see that all the edges can be systematically removed and re-organized, without any cycling occuring.

This figure will not encounter any cycling pattern.

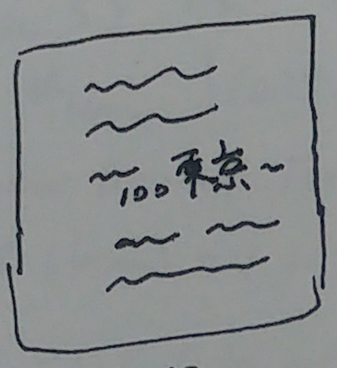
Alex Taniguchi
08-174510
Database Hw #6 11.1

Problem 71. At its core, the problem is really sortable as an intersection problem, where we want the Boolean set of documents (and positions) for "Tokyo"(東京) AND "University"(大学). We would normally start with the term known to have less matches and then search within those smaller-subset of matches for the term which had a larger subset of matches.

It is worth noting here that if the answer returned is $\{\phi\}$ (empty) then we are exempt from worries about position within the document, as having the term "東京大学"(Tokyo University) is impossible.

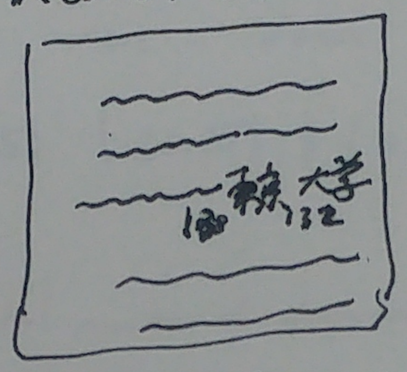
(Shall we try?)

Document 1(d_1)



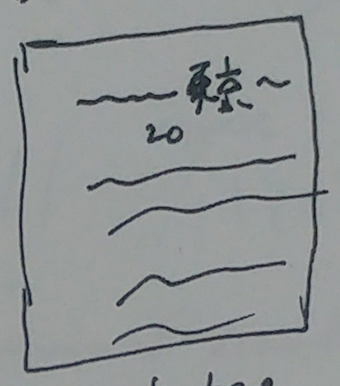
nope

Document 2(d_2)



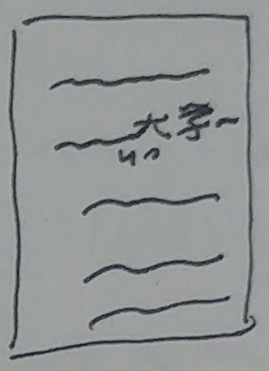
* This is interesting...

Document 3 (d_3)



not here

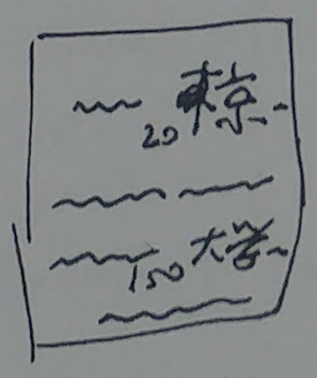
Document 4 (d_4)



nope

Now, my code above cannot handle the difference between 'Specimen Document (d_5)' and Document 2 (d_2) above.

Specimen Document (d_5)



Even with 130-words between 'Tokyo' and 'University', the overlap is enough to differ my boolean algorithm from above. How do I ensure that I have '東京' and '大学' next to each other?

The easiest (yet O(n)-time most expensive) solution is to make a positional index proximity-search algorithm with an extra-variable 'k'. While adjacency in English is '1 word' so k=1 in that case, adjacency in Japanese kanji by position is dependent on the length of the kanji. A very expensive algorithm could automate reading the length of your term (here, k=2) for you.

In the case of a posting index structure of max-length 'N', then individual search terms are solved in individual logarithmic time - O(log N), and the boolean search is $N \wedge N$ scans so it would be O(N log N). The position-adjacency check adds another (in generalized form) - "O(log N)" relative search, so the total time for my algorithm above is an admittedly dismal O((N+1) log N).

Problem 72. My three sentences will be from the classic poem 'Hickory Dickory Dock':

1 → The mouse ran up the clock. // mouse ran up clock

2 → The clock struck one. // clock struck one

3 → The mouse ran down! // mouse ran down

TF (Term frequency)

Normalise the TF (#s in BLUE)

(Doc 1) the mouse ran up clock // mouse ran up clock

#term 2($0.\bar{3}$) | ($0.1\bar{6}$) | ($0.1\bar{6}$) | ($0.1\bar{6}$) | ($0.1\bar{6}$) // 1 | (0.25) | (0.25) | (0.25) | (0.25)

(Doc 2) the ~~clock mouse~~ struck one // clock struck one

#term | (0.25) | (0.25) | (0.25) | (0.25) // | ($0.\bar{3}$) | ($0.\bar{3}$) | ($0.\bar{3}$)

(Doc 3) the mouse ran down // mouse ran down

#term | (0.25) | (0.25) | (0.25) | (0.25) // | ($0.\bar{3}$) | ($0.\bar{3}$) | ($0.\bar{3}$)

IDF (Inverse Document Frequency)

IDF (term) = $1 + \log_e$ (Total number of documents / Number of documents including the term within)

IDF (the) = $1 + \log_e(3/3) = 1.0000$

IDF (mouse) = $1 + \log_e(3/2) \approx 1.4055$

IDF (clock) = $1 + \log_e(3/1) \approx 2.0986$

IDF (ran) = $1 + \log_e(3/2) \approx 1.4055$

IDF (struck) = $1 + \log_e(3/1) \approx 2.0986$

IDF (up) = $1 + \log_e(3/1) \approx 2.0986$

IDF (one) = $1 + \log_e(3/1) \approx 2.0986$

IDF (down) = $1 + \log_e(3/1) \approx 2.0986$

TF * IDF (Term-Frequency * Inverse Document Frequency Score)

TF-IDF (the, doc 1) = $0.3\bar{3}$

TF-IDF (the, doc 2) = $0.1\bar{6}$

TF-IDF (the, doc 3) = $0.1\bar{6}$

TF-IDF (mouse, doc 1) $\approx$ 0.[redacted]2343

TF-IDF (mouse, doc 3) ≈ 0.3514

TF-IDF (clock, doc 2) ≈ 0.5247

TF-IDF (ran, doc 1) $\approx$ 0.[redacted]2343

TF-IDF (ran, doc 3) ≈ 0.3514

TF-IDF (struck, doc 2) ≈ 0.5247

TF-IDF (up, doc 1) ≈ 0.3498

TF-IDF (one, doc 2) ≈ 0.5247

TF-IDF (down, doc 3) ≈ 0.5247

TF-IDF (clock, doc 1) ≈ 0.3498

Reduced TF-IDF

mouse, $d_1 \approx 0.3514$

ran, $d_1 \approx 0.3514$

up, $d_1 \approx 0.5247$

clock, $d_1 \approx 0.5247$

clock, $d_2 \approx 0.6995$

struck, $d_2 \approx 0.6995$

one, $d_2 \approx 0.6995$

mouse, $d_3 \approx 0.4685$

ran, $d_3 \approx 0.4685$

down, $d_3 \approx 0.6995$

→ Next steps:

- Calculate TF-normalised
- Calculate TF-removed stop words - normalised
- Calculate TF-IDF - regular
- Calculate TF-IDF - reduced (remove stop words)

72 (continued). It is worth noting that we now have our numbers necessary for creating the vectors, that is:

$d_1 = \langle 0.33, 0.16, 0.16, 0.16, 0.16 \rangle$

$d_2 = \langle 0.25, 0.25, 0.25, 0.25, 0 \rangle$

$d_3 = \langle 0.25, 0.25, 0.25, 0.25, 0 \rangle$ for TF - regular case - ①

$d_1 = \langle 0.25, 0.25, 0.25, 0.25 \rangle$

$d_2 = \langle 0.33, 0.33, 0.33, 0 \rangle$ ← filler #s should be 0 (!)

$d_3 = \langle 0.33, 0.33, 0.33, 0 \rangle$ for TF - stop words removed case - ②

$d_1 = \langle 0.3333, 0.2343, 0.2343, 0.3498, 0.3498 \rangle$

$d_2 = \langle 0.1667, 0.5247, 0.5247, 0.5247, 0 \rangle$

$d_3 = \langle 0.1667, 0.3514, 0.3514, 0.5247, 0 \rangle$ for TF-IDF - regular case - ③

$d_1 = \langle 0.3514, 0.3514, 0.5247, 0.5247 \rangle$

$d_2 = \langle 0.6995, 0.6995, 0.6995, 0 \rangle$

$d_3 = \langle 0.4685, 0.4685, 0.6995, 0 \rangle$ for TF-IDF - stop words removed case - ④

in the $\cos(d_1, d_2, d_3) = \dfrac{d_1 \cdot d_2 \cdot d_3}{\|d_1\| \|d_2\| \|d_3\|}$

case ① $\cos(d_1, d_2, d_3) = \dfrac{0.050625}{(0.4597)(0.5)(0.5)} \approx \dfrac{0.050625}{0.114918\ldots} \approx 0.4405$ → 44.05% cos similarity

case ② $\cos(d_1, d_2, d_3) = \dfrac{0.081675}{(0.5)(0.57157\ldots)^2} = 0.5$ → 50.00% cos similarity

case ③ $\cos(d_1, d_2, d_3) = \dfrac{0.191965777707}{(0.6824)(0.9239)(0.7417\ldots)} \approx 0.4105$ → 41.05% cos similarity

case ④ $\cos(d_4, d_2, d_3) = \dfrac{0.477176616275}{(0.8931)(1.2116)(0.9635)} \approx 0.4577$ → 45.77% cos similarity

There are a couple of interesting points, esp. where case ① (TF - regular) is a near-match to the TF-IDF score when reduced by stop words. Considering that a targeted TF (removed stop words) has highest similarity, the IDF was not kind in having repeated words over short documents. As this is a children's poem with repetition and short structure, the 50%-ish similarity between all three lines is not a surprise. Indeed, with more time it would be fun to calculate similarities for other children's rhymes.

08-174510 pp. 4