MODELS OF COMPUTATION, DOCUMENT DISTANCE

What's an algorithm?

- computational procedure for solving a problem

input -> alg -> output

program	⟷	algorithm
programming language	⟷	pseudocode / structured English
computer (Random Access Memory)	⟷	model of computation (Random Access Machine)

Model of computation

specifies

- what operation an algorithm is allowed
- cost (time) of each operation

(1) Random Access Machine (RAM)

- Random Access Memory
  Modeled by big array
- in $O(1)$ time an algorithm can
  - load $O(1)$ words
  - do $O(1)$ computations
  - store $O(1)$ words
  - $O(1)$ registers

WORD : w bits

$w \geq \lg(\text{size of memory})$

(2) Pointer Machine

- dynamically allocated objects
- object has $O(1)$ fields
- field = word (e.g. int)
  or pointer to object

Python model

(1) "list" = array

$L[i] = L[j] + 5$ → $O(1)$ time

② object with $O(1)$ attributes

$x = x.next \rightarrow O(1)$ time

③ L.append(x)

↳ table doubling (LS)

↳ $O(1)$ time

④ $L = L1 + L2 \rightarrow$

≡ $L = []$

for x in L1: $O(1)$ $O(|L1|)$

L.append(x)

for x in L2: $O(1)$ $O(|L2|)$

L.append(x)

$\Rightarrow O(|L1| + |L2|)$

⑤ len(L) → constant

⑥ x in L → constant

⑦ L.sort() → $O(|L| \lg |L|)$

dictionary

⑧ dict: D[key] = val. → $O(1)$

⑨ long:

$x + y \rightarrow O(|x| + |y|)$ lg3

$x * y \to O(|x|+|y|)^2$

can be $O(|x|*|y|)$ — better

Document distance problem:

$d(D_1, D_2)$

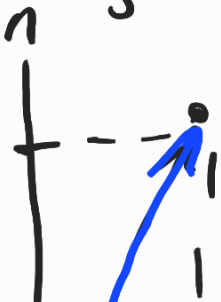
- document = sequence of words
- word = string of alphanumeric chars
- idea = shared words
- think of document as a vector

$D[w]$ = #occurences of w in D

D_1 = "the cat"

D_2 = "the dog"

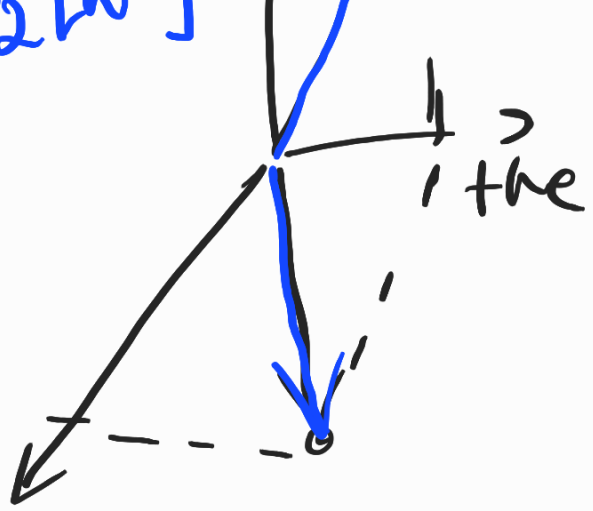
$d'(D_1, D_2) = D_1 \cdot D_2 =$



$= \sum_{k} D_1[w_j] \cdot D_2[w_j]$

$d'(D_1, D_2) = \frac{D_1 \cdot D_2}{|D_1| \cdot |D_2|}$

$d'(D_1, D_2) = \arccos$



Algorithm:

1. split doc. into words
2. compute word frequencies
3. compute dot product

iterat throw the words in doc:
and put in dictionary

$O(|doc|)$

freq[word] += 1

$O(|word|)$

OTHER OPTION

SORT and then count
