Lecture 2 : Models of Computation, Document Distance

□ Algorithm

al-khwārizmi : Father of algebra

◦ What's an algorithm

- Computational procedure for solving a problem

Input → alg. → output

program ⟷ algorithm

$\boxed{\text{programming language}}$ ⟷ $\boxed{\text{pseudo code, structed English}}$

Computer ⟷ Model of Computation

◦ Model of Computation

- what operation an algorithm is allowed
- Cost (time, ·····) of each operation

• Models

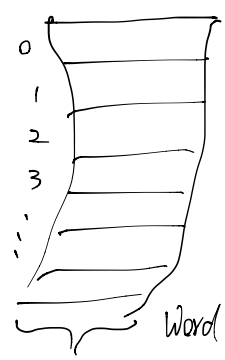
① Random Access Machine (RAM)

- Random access memory

  modeled by big array

- In const. time an algorithm can

  read in / load a const. number of



words from memory, do a const. num of operations on them and store (write out) them.

- Const. registers.
- A Word is w bits, $w \geq \lg$ (size of memory)

② Pointer Machine

- Dynamicly allocated objs.
- obj has const. num of fields
- field = word (e.g. int) or pointer to obj. or null (None)

□ Python Model

① "list" = array

$L[i] = L[i] + 5$ takes const. time

② obj. with const. num of attributes ("pointer machine model")

$X = X.\text{next}$ takes const time

$L.\text{append}(X)$ python does table doubling (lecture 9)

Can be done in Const. time

$L = L1 + L2$ Take linear time

$\equiv$ $L = [\ ]$

for x in $L1$: $\underbrace{L.\text{append}(x)}_{\text{const. time}}$ } $\Theta(|L1|)$

for x in $L2$: $\underbrace{L.\text{append}(x)}_{\text{const. time}}$ } $\Theta(|L2|)$

$\Theta(1 + L1 + L2)$

x in L : Take linear time

len(L): Const time, Lists in python are implemented with the length of the list build in. $\longrightarrow$ Instantaneous

L.sort() take $\Theta(n \log n)$ time

- time to compare items
- python uses algorithm "Comparision Sort" (lecture 3)

dict : D[key] = val.

- Takes const. time with high probability (randomized algorithm)
- Hash table (lecture 8 – 10)

long : x, y has $|x|, |y|$ words long

- $x+y$ takes $\Theta(|x| + |y|)$
- $x \cdot y$ takes $\Theta((|x| + |y|)^{\log_2 3})$, while straight forward method takes $\Theta(|x| \cdot |y|)$ time

heapq : lecture 4

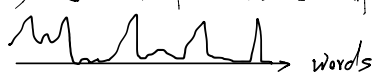
□ Document Distance Problem

Given two documents D_1, D_2, Compute the distance between them

- Distance: Are they identical

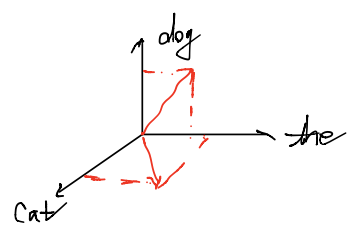
- document: sequence of words
- word: string of alphanumeric chars
- idea: look at shared words, use that to define distance
- Think of document as a vector

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



D_1 = "the cat"

D_2 = "the dog"



Document distance $\longleftrightarrow$ Vector distance

$$d(D_1, D_2) = \arccos\left(\frac{D_1 \cdot D_2}{|D_1| \cdot |D_2|}\right)$$

Algorithm:

1. Split document into words
2. Compute word frequencies
3. dot product

Implement in python:

② : iter thru doc.

$$\text{count}[\text{word}] \mathrel{+}= 1 \qquad \Theta\left(\sum |\text{word}| = |\text{doc}|\right)$$

$$\text{count}[\text{word}] = \Theta(|\text{word}|)$$

① Scan ~~thru~~ and look for alpha numerics