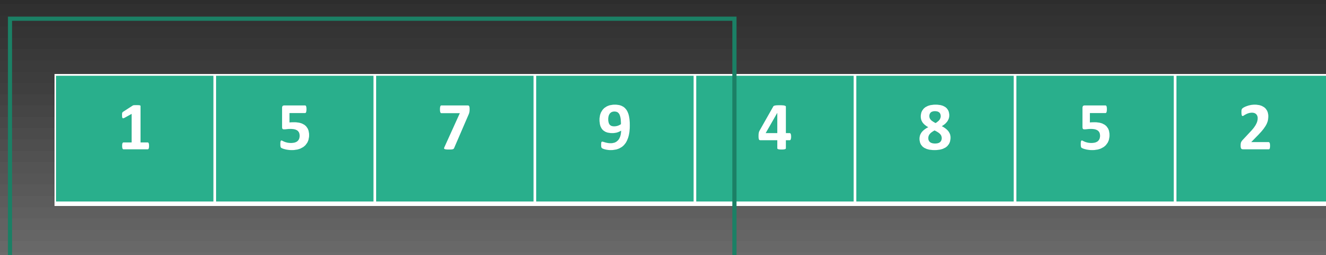
CS 4800: Algorithms & Data

Lecture 24

April 17, 2018

String matching

- Given a text T and a pattern P
- Find in the text T all occurrences of P



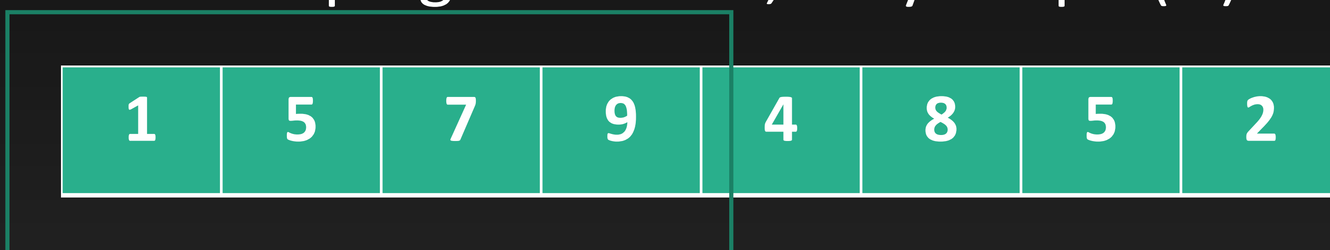
Streaming characters

- $1579 \rightarrow 5794$
- Delete first digit a, multiply by 10, add last digit b
- $N' = 10\left(N - 10^{|P|-1}a\right) + b$
- Slide window from left to right, in every step
  - Form N’ from current N
  - Compare N’ with pattern P
- Time: O(T)
- N might be too large to fit in an int

1	5	7	9	4	8	5	2

Rabin-Karp/rolling hash

- Pick a prime p
- h(N) = N mod p
- Instead of keeping track of N, only keep h(N)



$$h(N') = \left(10\left(N - 10^{|P|-1}a\right) + b\right) mod\ p$$

$$= \left(10\left((N\ mod\ p) - \left(10^{|P|-1}\ mod\ p\right)a\right) + b\right)\ mod\ p$$

Fixed prime p doesn’t work

- p = 131
- Pattern 1448 matches “1579”

1	**5**	**7**	**9**	**4**	**8**	**5**	**2**

Use random prime!

Use random prime

- $\pi(n)$: #primes smaller than or equal to n
- Fact: $\pi(n) \geq \frac{7}{8} \cdot \frac{n}{\ln n}$
- Consider at any location where the text does not match the pattern
- We compare 2 numbers smaller than $10^{|P|}$
- Their difference is smaller than $10^{|P|}$
- What is the probability random prime p divides the difference < $10^{|P|}$?

Collision probability

- At most $\log(10^{|P|})$ different primes divide the difference
- If we try a random prime p up to z then the probability of collision is at most $\frac{\log(10^{|P|})}{\pi(z)}$
- The probability we make error anywhere is at most $\frac{|\mathrm{T}| \cdot \log(10^{|P|})}{\pi(z)}$
- Exercise: how large is z to make failure prob. < 1/100?
- Can also pick k primes (with replacement)
- Exercise: what is failure prob. with k primes?

How to find random prime?

- Pick random number p in {2,3,…,z}
- Check if p is a prime
- $\pi(z) \geq \frac{7}{8} \cdot \frac{z}{\ln z}$
- Probability p is prime is at least $\frac{7}{8 \ln z}$
- Exercise: what is expected number of trials before we find a prime?

Hashing for large scale data processing

Document similarity

- Collection of documents (e.g. web crawl)
- Want to identify near duplicates
- How to identify exact duplicates?
  - Hashing
- Near duplicates could have very different hash values

Set similarity

- Two sets A and B of 64 bit numbers
- $sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- {1,3,5}, {3,7}
  - $sim(A, B) = 1/4$

Compute set similarity

- $sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$
- Midterm 1:
- Sort elements in A & B e.g. {1,3,5} and {3,4,5} give
  - 1,3,3,4,5,5
- # of pairs of consecutive elements that are equal
  - $|A \cap B|$
- $|A \cup B| = |A| + |B| - |A \cap B|$
- Time: O(n log n)

Fast approximation

- Random permutation π of 64 bit numbers
  - Random shuffling of all 64 bit numbers
  - 10, 7, 4, 5, …
  - $\pi(4) =$ position of 4 in the permutation
- Set S of numbers
- $\pi(S)$: position of numbers in S in the permutation
- When does $\min\left(\pi(A)\right) = \min(\pi(B))$?

Fast approximation

- When does $\min\big(\pi(A)\big) = \min(\pi(B))$?
- When there exists x such that
$$\pi(x) = \min\big(\pi(A)\big) = \min(\pi(B))$$
- $x \in A \cap B$ and after shuffling, it is the first among all numbers in $A \cup B$
- After random shuffling, all numbers have equal chance of being first
- {1, 3, 5} and {3, 7},
$$\Pr\Big[\min\big(\pi(\{1,3,5\})\big) = \min\big(\pi(\{3,7\})\big)\Big] = ?$$
- $\Pr\Big[\min\big(\pi(A)\big) = \min\big(\pi(B)\big)\Big] = \frac{|A \cap B|}{|A \cup B|}$

Fast approximation

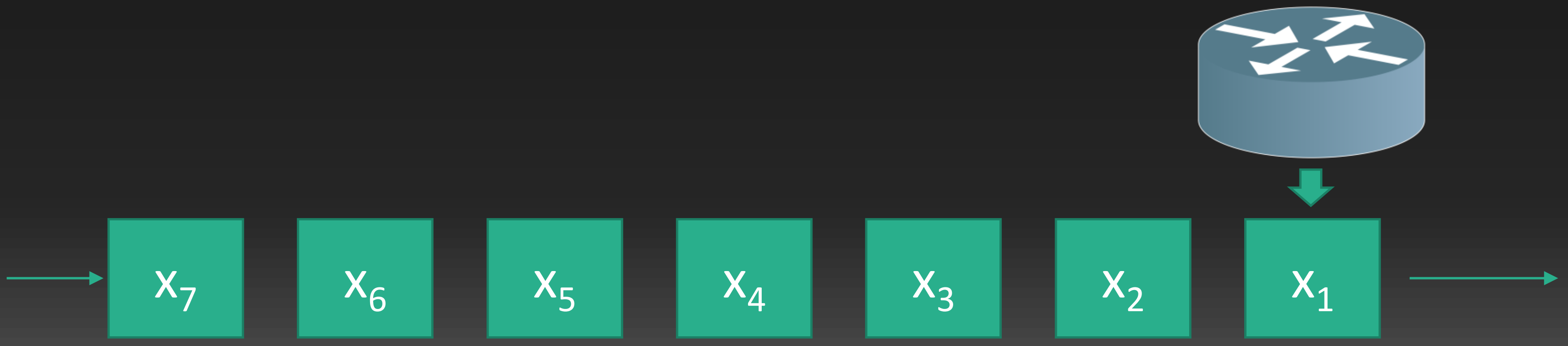
- Instead of 1, use 100 random permutations
- For each set A, compute $\min(\pi_i(A))$ for i=1,…,100
- To estimate sim(A,B)
  - Compare the min for each permutation
  - Count the number of times the minima agree
  - Divide by 100

Document similarity to set similarity

- Hash every 4 consecutive words (“shingle”) into a 64 bit number
- Each document D gives a set S_D of numbers
- Similarity of 2 documents A and B reduces to similarity of 2 sets S_A and S_B

Computation with limited storage

- Input data too large to fit in memory
- Compute information without storing whole input!
- Input arrives in a stream x_1, x_2,…
- Process one record at a time



Estimating distinct elements

- A stream of objects
- Goal: estimate the number of distinct objects
- Example: 1 2 3 3 2 1 1 3
- Applications
  - Router estimating number of communicating machines
  - Estimating number of distinct queries in query log

Hashing solution

- U: universe of all objects
- Hash function $h : U \rightarrow [0,1]$
- Algorithm
  - Apply h to every object x in the stream
  - Store the minimum value h(x) over all x seen so far
- Let y be the minimum hash value h(x) over all x in stream
- Observation:
  - y only depends on the collection of distinct values
  - Duplicate x's do not affect y

Analyzing y

- k: the number of distinct values in stream
- $E[y] = \frac{1}{k+1}$
- Thus, can estimate the number of distinct values by 1/y-1
- Can improve accuracy by having many hash functions and taking the average/median