Design and Analysis of Algorithms

CS3230

Week 6

Fingerprinting & Streaming

Warut Suksompong

Announcements

- Midterm during next lecture slot (March 3). Logistics uploaded to LumiNUS. Please read carefully!
- Video tutorial on this lecture's content to be uploaded on Monday of recess week (Feb 21)
- Week 7 Tutorial: Tutors may go over any material they like or turn their sessions into consultation sessions. No attendance taken.
- Week 8 Tutorial cancelled
- Week 9 onwards: Back to normal!

More announcements

- Practice midterm to be uploaded by Monday of recess week (Feb 21)
- Review session to go over solutions for practice midterm next Friday (Feb 25), 2-4pm, on Zoom
- Extra practice problems to be posted under Supplementary Materials
- Programming Assignment 1 (optional) posted soon, due on Sunday, March 6
- Anonymous mid-semester survey under “Survey” on LumiNUS, open until next Friday (Feb 25)

Today’s Agenda: Hashing II

- String pattern matching

- Frequency estimation in streaming model

Hash Table Resizing

- Last lecture, we discussed hash tables with $M = O(N)$, where N is number of inserted items and M is size of hash table.

- But in the dynamic setting, N is typically not known ahead of time. How do you set M?

- **Solution**: **Rehashing**. When N is too large, choose a new hash function of larger size and re-hash all elements.
  - The re-hashing step is costly, but it happens infrequently. We'll come back to this idea in the next lecture on ***amortized analysis***.

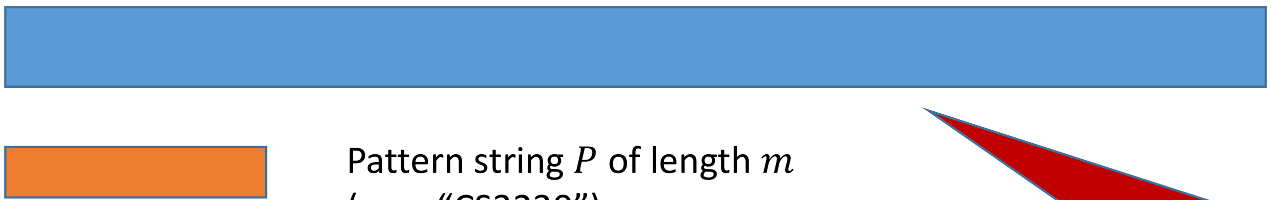
Design and Analysis of Algorithms

CS3230

Fingerprinting

Warut Suksompong

String Pattern Matching



Text String T of length n
(e.g., your emails)

Pattern string P of length m
(e.g., “CS3230”)

Strings assumed to be binary in lecture, but ideas generalize!

Does the pattern string P occur as a substring of the text string T?

Applications: Search engines, plagiarism detection, DNA sequencing, …

Naïve matching

```
NAIVE-STRING-MATCHER(T, P)
n = T.length
m = P.length
for s = 0 to n − m
    if P[1..m] == T[s + 1..s + m]
        print "Pattern occurs with shift" s
```

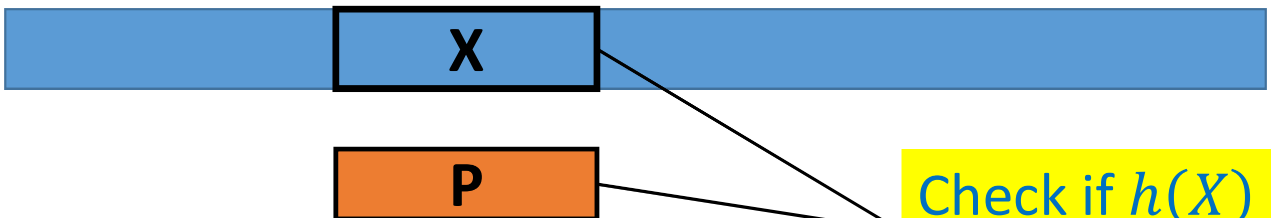
- Time complexity is $\Theta((n-m+1)m)$. If $m = n/3$, then this is $\Theta(n^2)$.
- Can we do better if m is $\Omega(n)$? What if we use randomization?

Karp-Rabin algorithm: Two great ideas

1. **Faster string equality**

2. **Rolling hash**

Faster String Equality



Check if $h(X) == h(P)$ for a hash function h.

- Will design h so that it outputs a small number on which arithmetic is doable in constant time.

- Runtime for equality check: $|\mathrm{hash}_P| + |\mathrm{hash}_X| + O(1)$

Cost of hashing P

- Total runtime for pattern matching:

$$|\mathrm{hash}_P| + (n - m + 1)(|\mathrm{hash}_X| + O(1))$$

Rolling Hash

- Will design hash function h so that we can update $h(T[1 \dots m])$ to $h(T[2 \;\dots m+1])$ in constant time. More generally, from $h(T[s+1 \dots s+m])$ to $h(T[s+2 \dots s+m+1])$ for all s.

- Total runtime for pattern matching:

$$\left|\text{hash}_{T[1\dots m]}\right| + |\text{hash}_P| + (n-m+1)\big(O(1)+O(1)\big) = O(m+n)$$

Division Hash

- Choose p to be a random prime number in the range $\{1, \ldots, K\}$.
  - Fact: # primes in range $\{1, \ldots, K\}$ is $> K/\ln K$.

- Define, for any integer x:

$$h_p(x) = x \bmod p$$

- If p is small and x is b-bits long in binary, hashing takes $O(b)$ time.

- Will show that hash family $\{h_p\}$ is “approximately” universal.

Probability of Collision

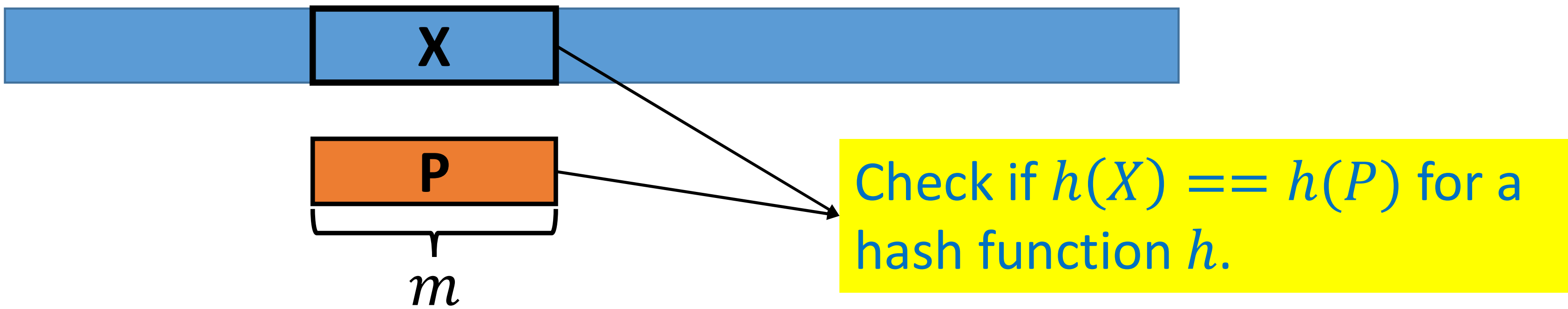
Claim: If $0 \leq x < y < 2^b$, then:

$$\Pr_p[h_p(x) = h_p(y)] < \frac{b \ln K}{K}.$$

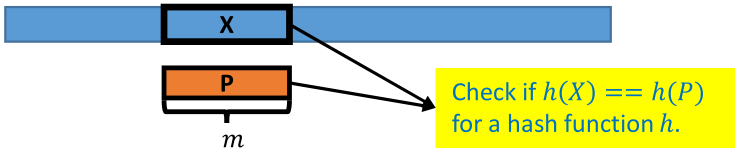
Proof:

- $h_p(x) = h_p(y)$ when $y - x = 0 \pmod{p}$. Let $z = y - x$.
- Note $z < 2^b$, so z can have at most b distinct prime factors. (Why?)
- p divides z if p is one of these $\leq b$ prime factors.
- Prob of this happening is $< b/\left(\frac{K}{\ln K}\right)$.

Equality Check Analysis: Example



- Suppose m is 1 million. Naïve equality check would take $\Omega(m)$ time.
- Set K to be 100 million
- If $X \neq P$, $\Pr_p[h(X) = h(P)] < 10^6 \cdot \frac{19}{10^8} < \frac{1}{5}$.
- Note that $K < 2^{32}$, so $h(X)$ and $h(P)$ can be stored in one machine word. They can be compared in constant time!

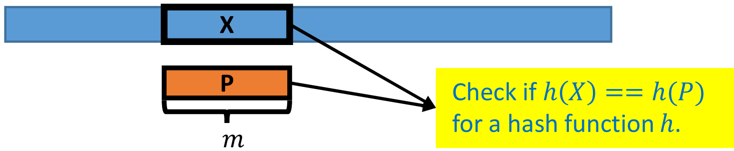


Equality Check Analysis: False Positives

Claim: Recall T and P are n and m bits long, respectively. Let $K = 200mn\ln(200mn)$. Then, probability of getting a false positive is < 1%.

Proof:

- Let E_i be the event that $T[i+1 \ldots i+m]$ and P are unequal but have the same hash.
- $\Pr[E_i] < m\dfrac{\ln(200mn\ln(200\,mn))}{200mn\ln(200\,mn)} < \dfrac{1}{200n}\left(1 + \dfrac{\ln(\ln(200\,mn))}{\ln(200\,mn)}\right) < \dfrac{1}{100n}$.
- $\Pr[\cup_i E_i] \leq \sum_i \Pr[E_i] < \frac{1}{100}$. (First inequality uses "union bound")



Equality Check Analysis: Runtime

- If $K = 200mn\ln(200mn)$, then any prime $\leq K$ can be stored as a bit string of length $\lg K = O(\lg n)$.

- In the word-RAM model, can assume that $O(\log n)$ bits fit in a constant number of machine words.
  - E.g., all of the internet is about 5 billion GB, so $n \approx 2^{70}$. But $\lg n$ fits in three 32-bit words.

- Can check equality of hashes in constant time!

Rolling Hash: Main Observation

- Let X and X' be the binary numbers corresponding to $T[1 \ldots m]$ and $T[2 \ldots m+1]$.

$$X = \sum_{i=1}^{m} T[i] \cdot 2^{m-i} \qquad\qquad X' = \sum_{i=1}^{m} T[i+1] \cdot 2^{m-i}$$

- E.g., if $T = [a_1, a_2, a_3]$ with $m = 2$, then $X = 2a_1 + a_2$ and $X' = 2a_2 + a_3$. You can “roll” from X to X' by $X' = 2X - 4a_1 + a_3$.

Rolling Hash: Main Observation

- Let X and X' be the binary numbers corresponding to $T[1 \ldots m]$ and $T[2 \ldots m+1]$.

$$X = \sum_{i=1}^{m} T[i] \cdot 2^{m-i} \qquad\qquad X' = \sum_{i=1}^{m} T[i+1] \cdot 2^{m-i}$$

- $X' = 2X - 2^m T[1] + T[m+1]$.

- Can roll X into X'. What about $h_p(X)$ into $h_p(X')$?

Rolling Hash: Main Observation

- $X' = 2X - 2^m T[1] + T[m+1]$.

- Division hash is **linear**:

$$h_p(X') = 2h_p(X) - T[1] \cdot h_p(2^m) + T[m+1] \pmod{p}$$

- Given $h_p(X)$ and $h_p(2^m)$, can get $h_p(X')$ in constant time!

Ready to hash and roll!

- Monte Carlo algorithm with error probability $< 1\%$ and runtime $O(m+n)$.
- Recap:

1. Pick random prime p in range $\{1, \ldots, \lceil 200mn \ln(200\, mn) \rceil\}$
2. Compute $h_p(P)$, $h_p(2^m)$ and $h_p(T[1 \ldots m])$
3. Check if $h_p(P) == h_p(T[1 \ldots m])$
4. For each $i = 1 \ldots n - m$
   i. Update $h_p(T[i \ldots i+m-1])$ to $h_p(T[i+1 \ldots i+m])$, using $T[i], T[i+m]$, and pre-computed $h_p(2^m)$
   ii. Check if $h_p(T[i+1 \ldots i+m])$ equals $h_p(P)$

- Idea generalizes to more complicated pattern matching problems.

Design and Analysis of Algorithms

National University of Singapore

CS3230

Streaming

Warut Suksompong

Drinking from a fire hydrant

Hard Drive

RAM

Network

Local storage

Science experiments

Goal: Maintain small data structure to *approximately* answer queries about data stream

Streaming Model

- Sequence of insertions or deletions of items from a large universe $\mathcal{U}$.
  - Think of the universe as $\{1, \ldots, U\}$ for a large number U.

- E.g., `add(3), add(1), add(7), add(3), add(7), delete(3), add(1), delete(3), delete(7)`

- At the end of the stream, **frequency** f_i of an item i is its net count.
  - In above example, $f_3 = 0, f_1 = 2, f_7 = 1$.
  - We assume that $f_i \geq 0$ for all i.

- Let M denote the sum of all frequencies (i.e., total net count) at the end of the stream.
  - Above, $M = 0 + 2 + 1 = 3$.

Frequency Estimation

For a query $i \in [U]$, give a "good" estimate of f_i at end of the stream.

- Call an approximation $\hat{f}_i$ to be **ϵ-approximate** if:

$$f_i - \epsilon M \leq \hat{f}_i \leq f_i + \epsilon M$$

Naïve Solutions

- One simple solution is to create a direct access table T, where you increment (decrement) $T[i]$ if i is added (deleted).
    - Space: $\Omega(U)$. Terrible!

- Another simple solution is to store the items with nonzero frequency as a sorted list.
    - Space: $\Omega(M)$. Update times not $O(1)$.

Use a Hash Table!

- Choose a hash function h: $\mathcal{U} \rightarrow \{1, \dots, k\}$
- Initiate an empty table A of size k
- If j is added, increment $A[h(j)]$
- If j is deleted, decrement $A[h(j)]$

- At end of stream, to estimate f_i, return $A[h(i)]$

Analysis

- Note:

$$A[h(i)] = \sum_{j:h(j)=h(i)} f_j$$

- No matter the choice of h, observe $A[h(i)] \geq f_i$. Never an **underestimate**.

- If h is drawn from a universal family, $\mathbb{E}[A[h(i)] - f_i] \leq M/k$.
  - See analysis in lecture notes or in presentation.

- Output is in expectation an ϵ-approximation if $k = 1/\epsilon$. Space is $O\left(\frac{1}{\epsilon}\right)$ counters plus cost of storing hash function, $O(\log U \cdot \log(k))$.

Count-Min Sketch

- Instead of a bound on the expected error, a stronger guarantee would ensure low error with high probability.

- This can be done by constructing multiple hash tables. Estimate of f_i obtained by taking minimum of each table's overestimate.

- Analysis beyond of the scope of this module. Take CS5330 ☺

Acknowledgement

- The slides are modified from
  - Prof. Arnab Bhattacharyya