Outline

String Matching Problem

- Given a text T and a pattern P, find all occurrences of P within T
- Notations:
  - n and m: lengths of P and T
  - Σ: set of alphabets (of constant size)
  - P_i: ith letter of P (1-indexed)
  - a, b, c: single letters in Σ
  - x, y, z: strings

Example

- $T =$ AGCATGCTGCAGTCATGCTTAGGCTA
- $P =$ GCT
- P appears three times in T

- A naive method takes $O(mn)$ time
  - Initiate string comparison at every starting point
  - Each comparison takes $O(m)$ time

- We can do much better!

Outline

Hash Function

- A function that takes a string and outputs a number
- A good hash function has few collisions
  - *i.e.*, If $x \neq y$, $H(x) \neq H(y)$ with high probability
- An easy and powerful hash function is a polynomial mod some prime p
  - Consider each letter as a number (ASCII value is fine)
  - $H(x_1 \dots x_k) = x_1 a^{k-1} + x_2 a^{k-2} + \cdots + x_{k-1} a + x_k \pmod{p}$
  - How do we find $H(x_2 \dots x_{k+1})$ from $H(x_1 \dots x_k)$?

Hash Table

- Main idea: preprocess T to speedup queries
  - Hash every substring of length k
  - k is a small constant
- For each query P, hash the first k letters of P to retrieve all the occurrences of it within T
- Don't forget to check collisions!

Hash Table

- Pros:
  - Easy to implement
  - Significant speedup in practice

- Cons:
  - Doesn't help the asymptotic efficiency
    - Can still take $\Theta(nm)$ time if hashing is terrible or data is difficult
  - A lot of memory consumption

Outline