• **DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

• Main topics:
  - Review: NP hardness, NP completeness, and Approximation Algorithms.
  - Brief overview of string hashing.

The plan is to talk briefly about hashing, and the use of randomized algorithms. We will do so using the string matching problem, which is a well-studied problem in the processing of signals, text, and biological data.

The simplest form of string matching is given a pattern $P$ determine whether it exists as a substring of some text $T$. Doing this by brute force takes time $|P| \cdot (|T| - |P| + 1)$, which is quadratic in input size.

Consider the following heuristic of doing better: treat the characters as integers, and compute the sum of every contiguous chunk of length $|P|$. Once we have these sums, only check the ones whose sum match with the sum of $P$. This can be implemented in linear time: we can compute the sum of each prefix,

$$s_i = \sum_{j \leq i} x_i,$$

then we have

$$\sum_{l \leq j \leq r} = s_r - s_{l-1},$$

so we can actually query for the value of any subinterval in $O(1)$ time.

This smaller quantity is known as a **hash**, and is applicable to many problems. In this case, it’s still possible for two very different strings to have the same hash, e.g.

```
aaaaaab, baaaaa.
```

In fact, any hashing scheme that simply sums the characters is not sensitive to the ordering of the characters. This is hugely problematic for string hashing because the order of the characters is quite important.
This issue is addressed by the Rabin-Karp scheme. Instead of taking the sum, we multiply element $j$ by $z^i$ for some $z$ that we'll pick randomly, aka.

$$s_i = \sum_{j \leq i} z^j \times x_j.$$  

Then the substring in $[l, r]$ has sum

$$s_r - s_{l-1} = \sum_{l \leq j \leq r} x_j z^l = z^{l-1} \left( \sum_{l \leq j \leq r} x_j z^{l-j+1} \right).$$

Note that the term in the bracket is precisely what we’d get if we hash the string $x_l, x_{l+1} \ldots x_r$ by itself. Therefore we can compute the hash of the pattern, multiply it by $z^{l-1}$, and check for equality.

Furthermore, all of these computations can be done modulo some prime number, and the main result of the Rabin-Karp theorem says that as long as this prime of size around $n^{O(1)}$, then with high probability the hash computations are correct. Its proof relies on theorems about the number of roots of polynomials.