

University of Illinois at Urbana-Champaign  
Dept. of Electrical and Computer Engineering

## ECE 220: Computer Systems & Programming

Review: Letter Frequency Planning

## Time to Write Another Program

Let's say that we want to do the following:

- given an **ASCII** string (a sequence of characters terminated by a **NUL**, **ASCII x00**),
- count the occurrences of each letter (regardless of case), and
- count the number of non-alphabetic characters.

## Let's Develop a Flow Chart

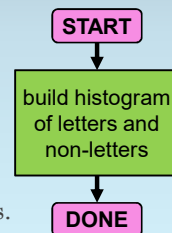
Ready?

My work here is done.

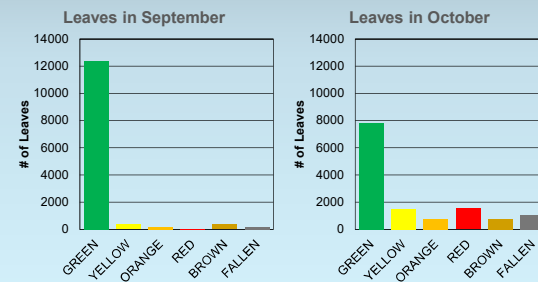
Now you can apply **systematic decomposition**.

What's a histogram?

A function on a set of categories.



## These Are Examples of Histograms



## We Need to Count Each Kind of Letter

So we want a set of counts for a string:

- How many A's (either case)?
- How many B's?
- ...
- How many Z's?
- How many non-alphabetic characters?

**How would you perform this task?**

## Let's Do an Example

“Try this string as an example.”

**How many A's? 3**

**How many B's? 0**

**How many C's? 0**

**How many D's? 0**

**How many E's? 2**

## Algorithm 1: Look Through String Once for Each Letter

Maybe something like this?

```

for each letter (and once for non-letters)
  count = 0
  for each character in the string
    if character matches letter (either case)
      count = count + 1
  store count for the letter in histogram
  
```

## Another Example: a Book

Second example: the Patt and Patel textbook.

**How many A's? 61,341**

**How many B's? 10,821**

**How many C's? Do you really think**

**How many D's? I counted these?**

**How many E's?**

**Would you approach the problem differently with a longer string?**

## Algorithm 2: Look through String Once

For a longer string, maybe we just want to look through it once?

**initialize 27-bin histogram to all 0s**  
**for each character in the string**  
**increment the appropriate histogram bin**

But figuring out which bin to increment may be complicated.

## Algorithm 3: Build a Bigger Histogram

What if we build a bigger histogram first:

**initialize 128-bin histogram to all 0s**  
**for each character in the string**  
**increment bin for that character**  
**for each letter**  
**add the two corresponding bins**  
**sum all non-letter bins**

Now finding the bin is easy, but we need extra memory and computation.

## Which Algorithm is Best?

### Which approach is better?

What is the metric?

- Number of instructions executed?
- Number of clock cycles (time) required?
- Amount of memory needed?

Does our answer depend on the length of the string?

What if the string is sorted alphabetically?

## Let's Pick Algorithm 2

The answer depends on the context and the application of our program.

We're going to go with Algorithm 2:

**initialize 27-bin histogram to all 0s**  
**for each character in the string**  
**increment the appropriate histogram bin**

Why? Implementing the complex decision in the middle will be interesting.