CSE 417: Algorithms and Computational Complexity

Pattern Matching
The Value of Preprocessing

Autumn 2002
Paul Beame

Pattern Matching

- Given
  - a string, **s**, of **n** characters
  - a pattern, **p**, of **m** characters
  - usually **m<<n**
- Find
  - all occurrences of the pattern **p** in the string **s**
- Obvious algorithm:
  - try to see if **p** matches at each of the positions in **s**, stopping at a failed match

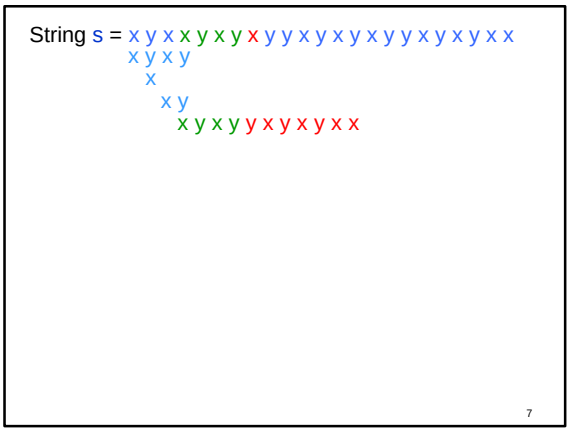
String s = x y x x y x y x y y x y x y y x y x y x x

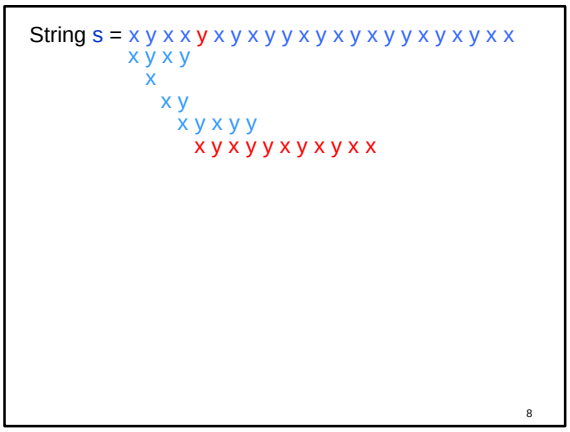
Pattern p = x y x y y x y x y x x

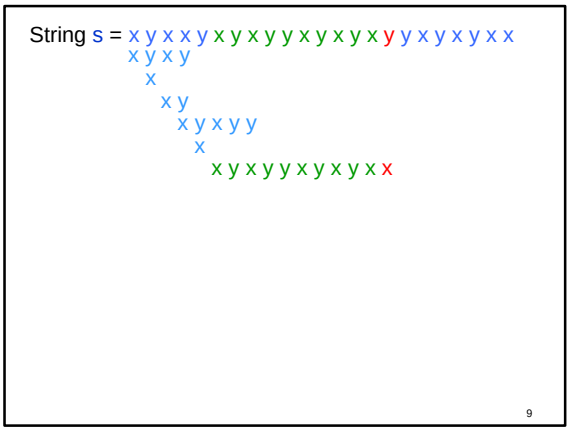
String s = x y x x y x y x y y x y x y y x y x y x x
x y x y y x y x y x x

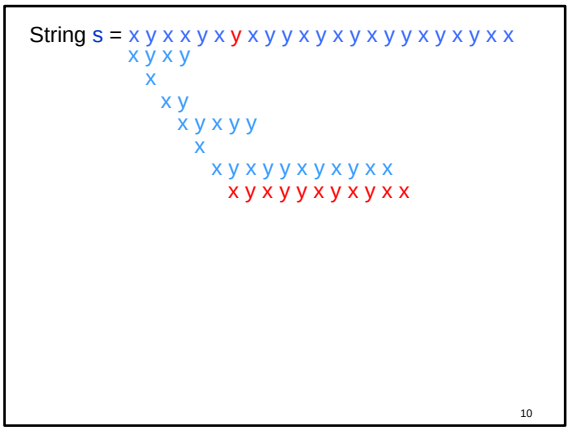
String s = x y x x y x y x y y x y x y y x y x y x x
x y x y
x y x y y x y x y x x

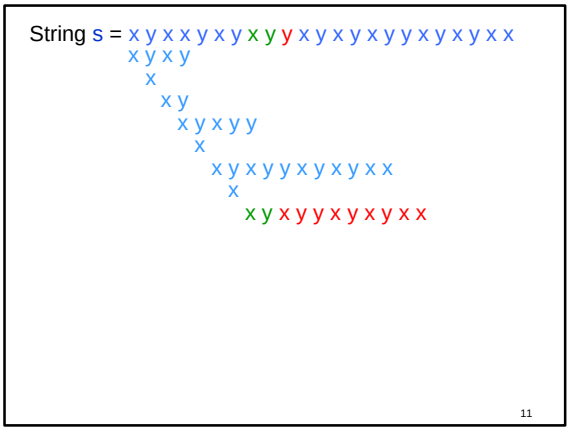
String s = x y x x y x y x y y x y x y y x y x y x x
x y x y
x
x y x y y x y x y x x

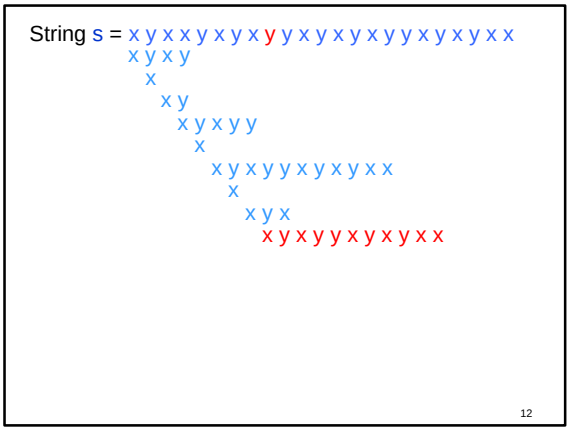

String s = x y x x y x y y x y x y x y y x y x x x
x y x y
x
x y
x y x y y x y x y x x
7


String s = x y x x y x y x y y x y x y x y y x y x y x x
x y x y
x
x y
x y x y y
x y x y y x y x y x x
8


String s = x y x x y x y x y y x y x y x y y x y x x
x y x y
x
x y
x y x y y
x
x y x y y x y x y x x
9


String s = x y x x y x y x y y x y x y x y y x y x x
x y x y
x
x y
x y x y y
x
x y x y y x y x y x x
x y x y y x y x y x x
10


String s = x y x x y x y x y y x y x y x y y x y x x
x y x y
x
x y
x y x y y
x
x y x y y x y x y x x
x
x y x y y x y x y x x
11


String s = x y x x y x y x y y x y x y x y y x y x x
x y x y
x
x y
x y x y y
x
x y x y y x y x y x x
x
x y x
x y x y y x y x y x x
12

String s = x y x x y x y x y y x y x y x y y x y x y x x

x y x y
x
x y
x y x y y
x
x y x y y y x y x y x x
x
x y x
x
x y x y y y x y x y x x

String s = x y x x y x y x y y x y x y x y y x y x y x x

x y x y
x
x y
x y x y y
x
x y x y y y x y x y x x
x
x y x
x
x
x y x y y y x y x y x x

String s = x y x x y x y x y y x y x y x y y x y x y x x

x y x y
x
x y
x y x y y
x
x y x y y y x y x y x x
x
x y x
x
x
x y x y y
x y y x y x y x y x x

String s = x y x x y x y x y y x y x y x y y x y x y x x

x y x y
x
x y
x y x y y
x
x y x y y y x y x y x x
x
x y x
x
x
x y x y y
x y x y y y x y x y x x

Worst-case time
O(mn)

String s = x y x x y x y x y y x y x y x y y x y x y x x

x y x y
x
x y
x y x y y
x
x y x y y y x y x y x x
x
x y x
x
x
x y x y y
x
x y x y y x y x y x x

Lots of wasted work

Suppose we knew the pattern well

Since we know earlier agreement of the string with the pattern, these can't possibly match

Preprocess the Pattern

- After each character in the pattern figure out ahead of time what the next useful work would be if it failed to match there
  - i.e. how much can one shift over the pattern for the next match

String s = x y x x y x y x y y x y x y x y x y x x x y x y

x y x y y

x y x y y x y x y x x

x y x y y x y x y x x

19

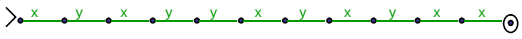
Preprocessing the pattern

- At each mismatch
  - Look at the last part that matched plus extra mismatched character
  - Try to fit pattern as far to the left in this as possible
    - i.e. look for the longest prefix of the pattern that matches the end of the sequence so far.

20

Preprocessing the pattern

Pattern p=x y x y y x y x y x x

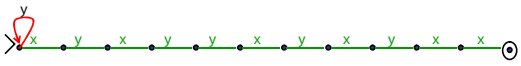


Each dot represents how far in the pattern things are matched

21

Preprocessing the pattern

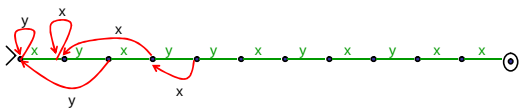
Pattern p=x y x y y x y x y x x



22

Preprocessing the pattern

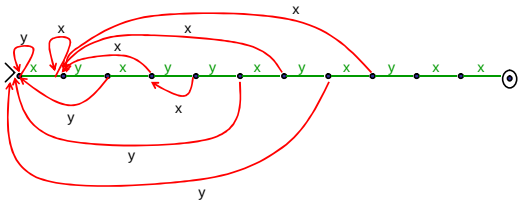
Pattern p=x y x y y x y x y x x



23

Preprocessing the pattern

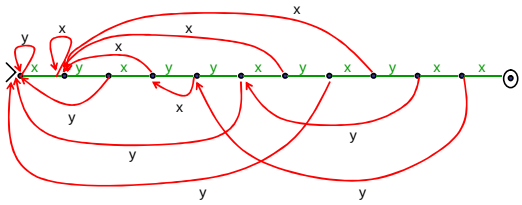
Pattern p=x y x y y x y x y x x



24

Preprocessing the pattern

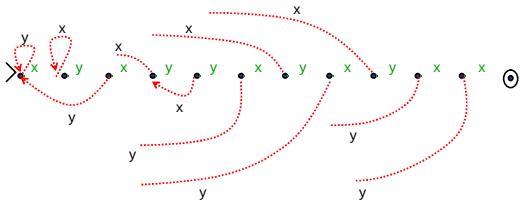
Pattern p=x y x y y x y x y x x



25

Running on the string

String s = x y x x y x y x y y x y x y x y x y x x



26

Knuth-Morris-Pratt Algorithm

- Once the preprocessing is done there are only n steps on any string of size n
  - just follow your nose
- Obvious algorithm for doing preprocessing is $O(m^2)$ steps
  - still usually good since m<<n
- Knuth-Morris-Pratt Algorithm can do the pre-processing in O(m) steps
  - Total O(m+n) time

27

Finite State Machines

- The diagram we built is a special case of a **finite automaton**
  - start state
  - goal or accepting state(s)
  - an arc out of each state labeled by each of the possible characters
- Finite automata take strings of characters as input and decide what to do with them based on the paths they follow

28

Finite State Machines

- Many communication protocols, cache-coherency protocols, VLSI circuits, user-interfaces, even adventure games are designed by making finite state machines first.
  - The "strings" that are the input to the machines can be
    - a sequence of actions of the user
    - the bits that arrive on particular ports of the chip
    - a series of values on a bus

29

Finite State Machines

- Can search for arbitrary combinations of patterns not just a single pattern
  - Given two finite automata can build a single new one that accepts strings that either of the original ones accepted
- Typical text searches are based on finite automata designs
  - Perl builds this in as a first-class component of the programming language.

30