Strings

Marco Gallotta

SACO Training Camp

March 2007

String Matching

- Text: A string of *n* characters to search in
- Pattern: A string of *m* characters to search for

Brute Force

- Search for pattern starting at each position
- Reasonable for average case, but O(*nm*) worst case

Example

- Searching for “BARBER”
  - BARBER
  - |
  - ... A ...
  - BARBER
  - |
  - ... A ...

Horspool's Algorithm

- The "A" will cause a definite failure for the next three positions, so we can safely shift by four.
  - `BARBER`
  - `      |`
  - `  ... A ...`
  - `    BARBER`
  - `      |   |`
  - `  ... A ...`

Shift Table

```
jump = [m,m,...,m]
for i = 0..m-2:
   jump[pattern[i]] = m-1-i
```

- `jump[a]` is the shift when failing to match character `a`.
- For the pattern “BARBER” the shift table is:

Horspool's Algorithm: Example

- *JIM SAW* ME IN A BARBERSHOP
- BARBER

Horspool's Algorithm: Example

- JIM *S**A**W ME* IN A BARBERSHOP
- BARBER
- B**A**RBER

Horspool's Algorithm: Example

- JIM S*AW* ***ME*** IN A BARBERSHOP
- BARBER
- BARBER
- BARB**E**R

Horspool's Algorithm: Example

- JIM SAW ME *IN A B*ARBERSHOP
- BARBER BARBER
- BARBER
- BARBER

Horspool's Algorithm: Example

- JIM SAW ME IN *A* ***B****A**R*BERSHOP
- BARBER BARBER
- BARBER BAR**B**E*R*
- BARBER

Horspool's Algorithm: Example

- JIM SAW ME IN A *BA**R**BER*SHOP
- BARBER BARBER
- BARBER BARBER
- BARBER *BA**R**BER*

Analysis

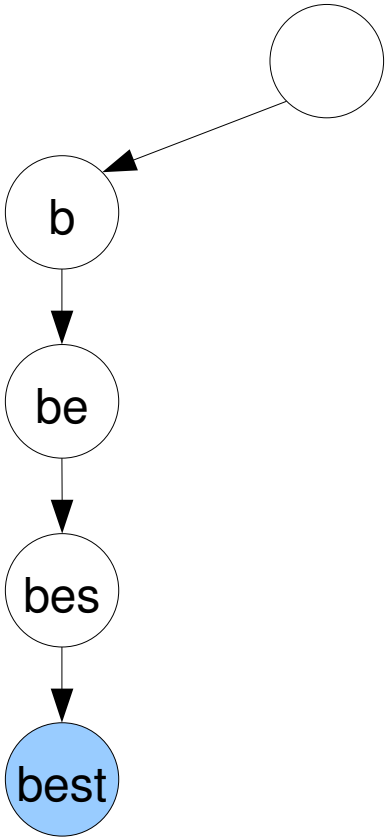
- For random text it is O(*n*), but worst case is still O(*nm*)
- Example: Pattern “baa”, text “aaaaaaaa”
- Boyer-Moore algorithm improves the worst case – O(*n*)
- Uses an additional shift table for based on the matching suffix

Trie

- Data structure for storing a set of strings
- Edges represent characters
- Nodes represent string terminations

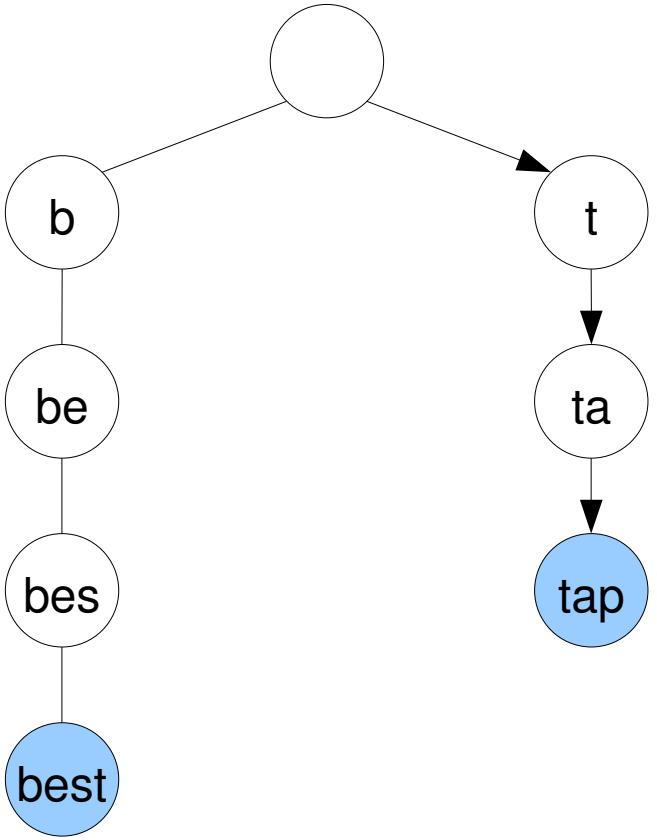
Trie

- **best**
- tap
- be
- bat
- bet
- train
- bed



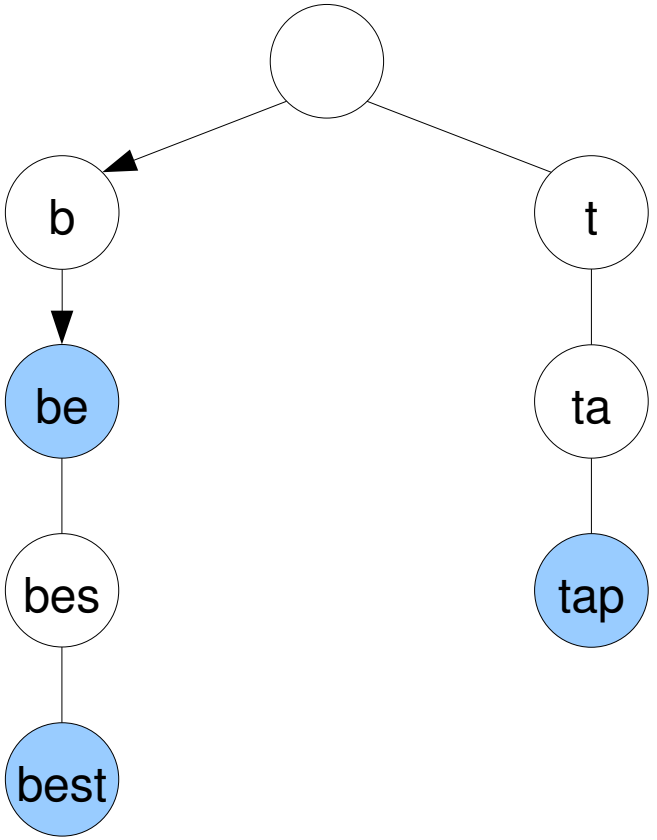
Trie

- best
- **tap**
- be
- bat
- bet
- train
- bed



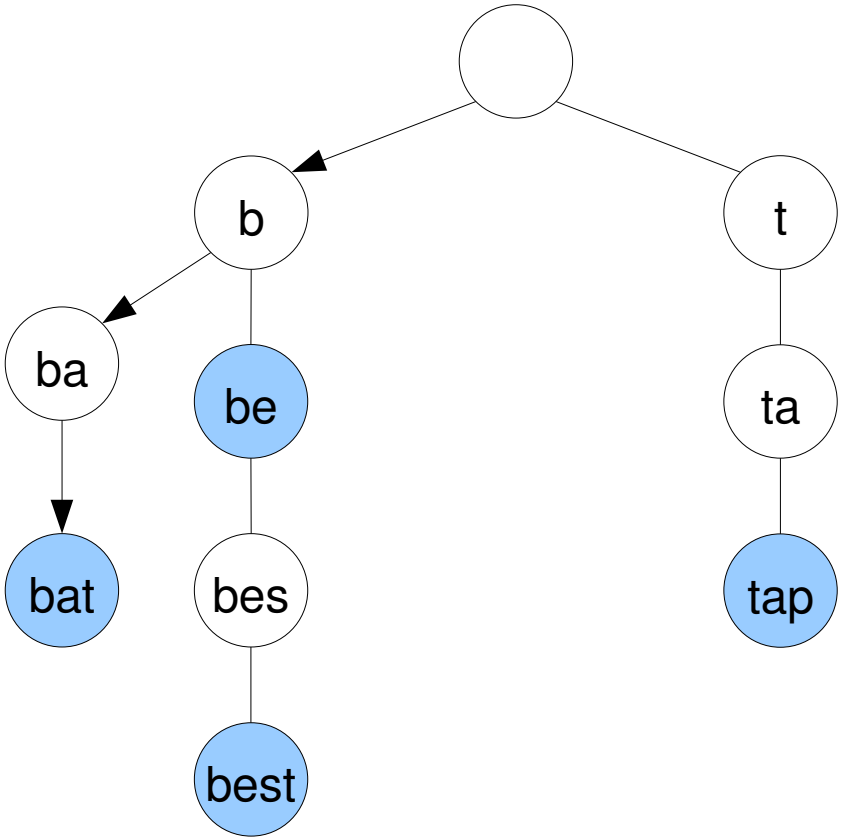
Trie

- best
- tap
- **be**
- bat
- bet
- train
- bed



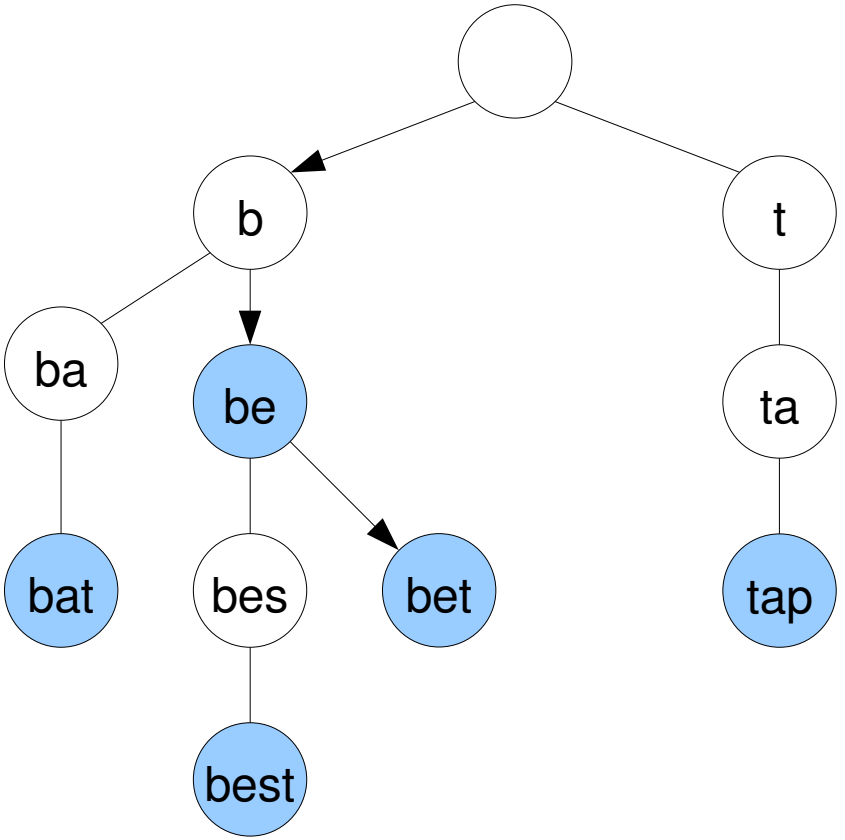
Trie

- best
- tap
- be
- **bat**
- bet
- train
- bed



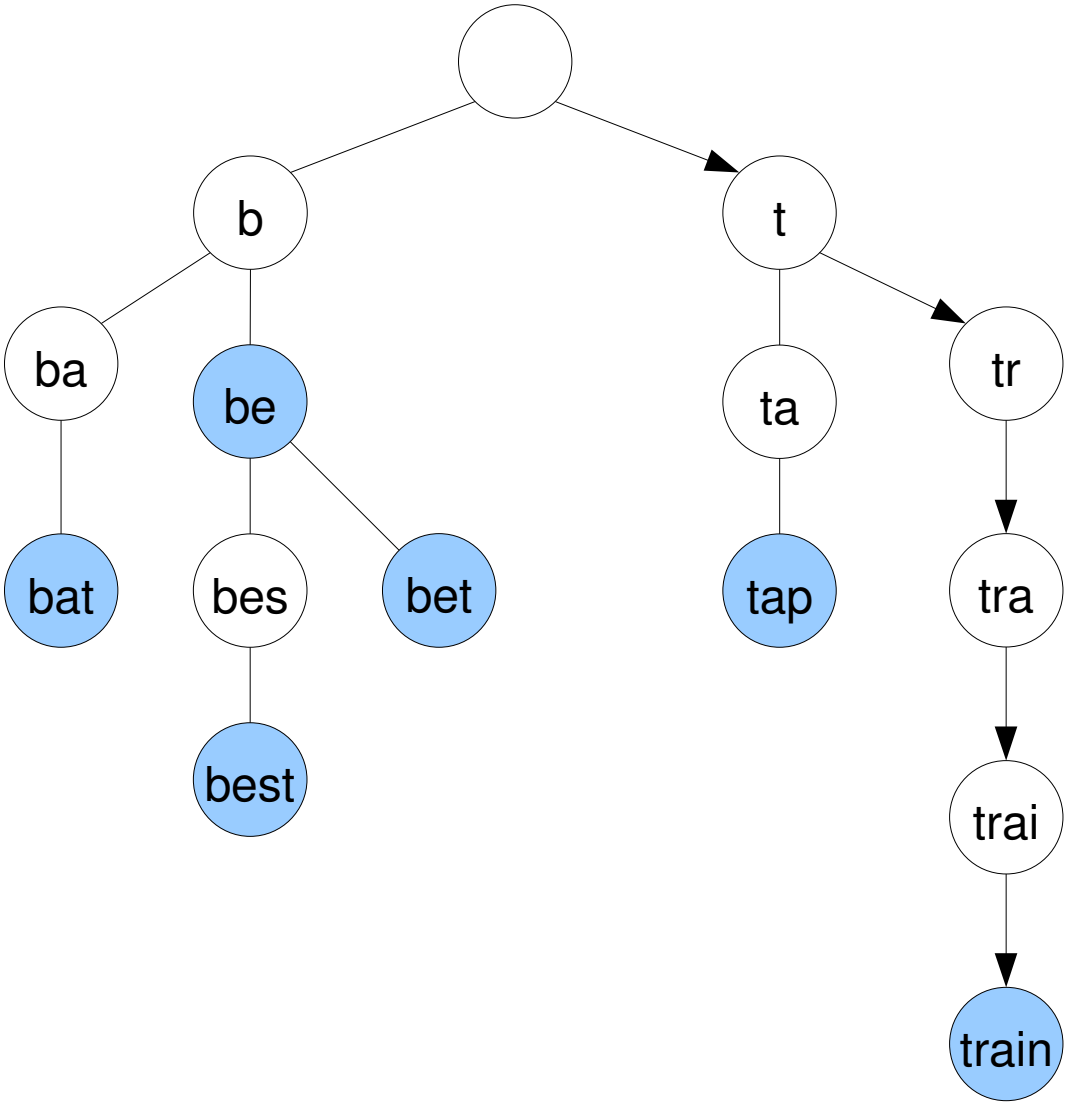
Trie

- best
- tap
- be
- bat
- **bet**
- train
- bed



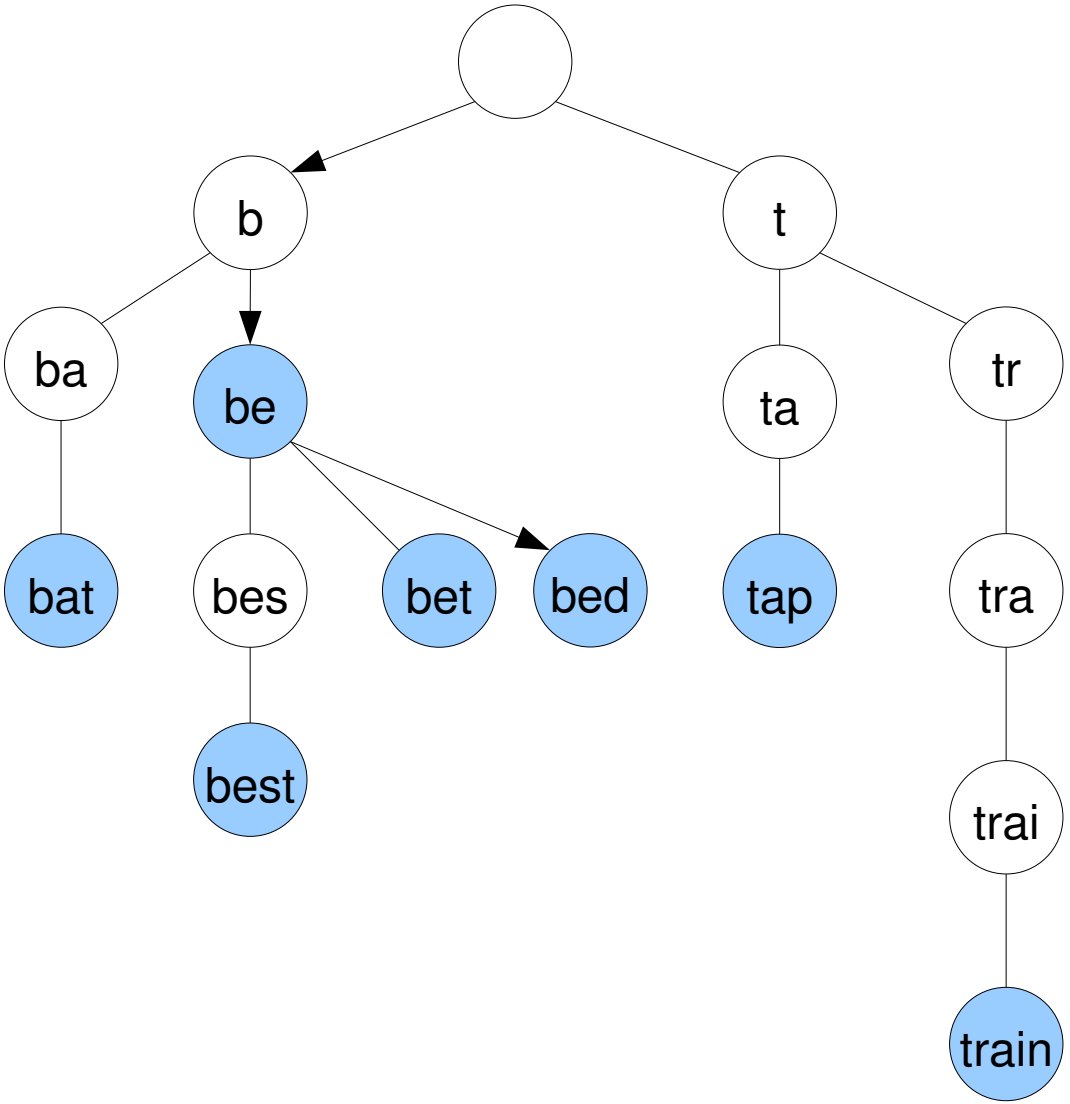
Trie

- best
- tap
- be
- bat
- bet
- **train**
- bed



Trie

- best
- tap
- be
- bat
- bet
- train
- **bed**



Trie: Analysis

- Insertions and lookups are O(*n*) for a string of length *n*
- Longer strings and sparse graphs take up lots of memory
- Patricia tries solve this problem by grouping consecutive non-terminating edges

Questions

?