String Algorithms
Segment Trees
Dijkstra
Floyd-Warshall

Floris Kint

String algorithms

Longest common substring (DP)

Trie = prefix tree

String matching

Longest common substring

cache[i][j] =

cache[i-1][j-1] + 1, if a[i]==b[j]

0, if a[i] != b[j]

DP, only need one level of history

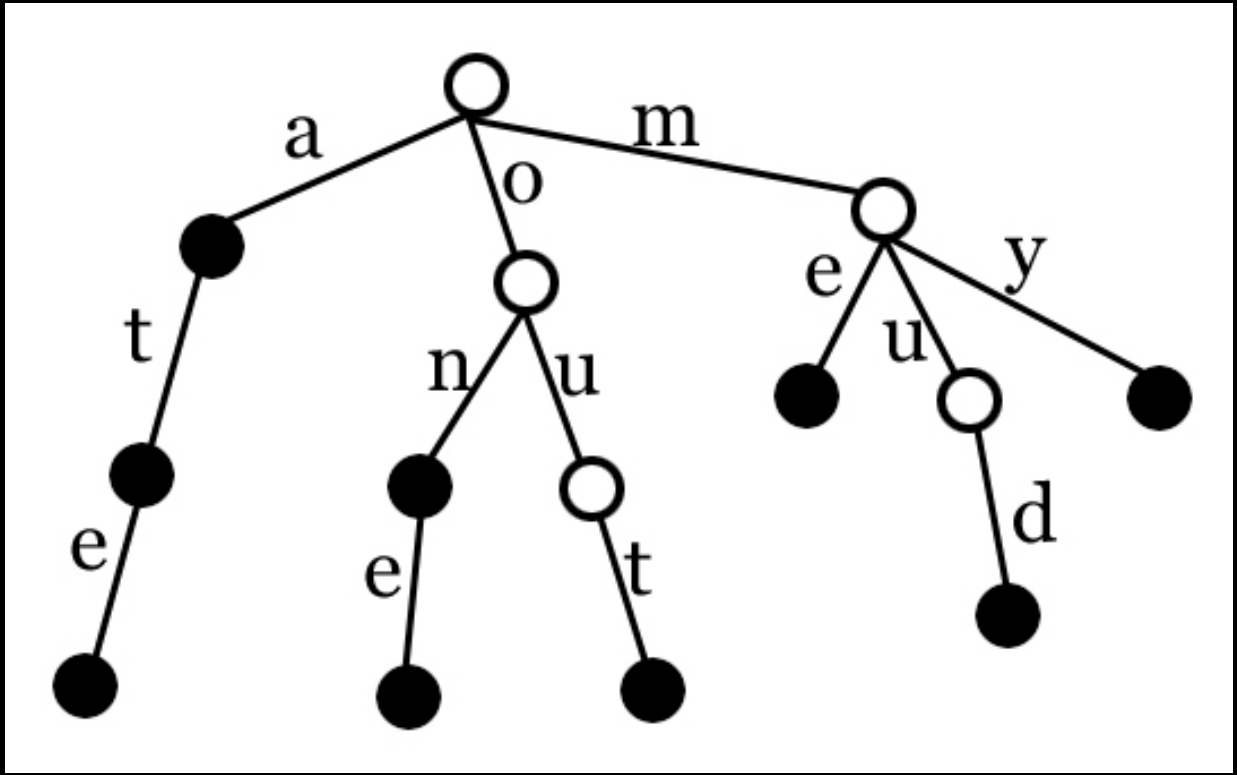
-> O(N) space complexity

Prefix tree (aka Trie)

Store set of strings by their prefixes.

Applications: auto-complete

Prefix tree (aka Trie)



Exercises

- UVa 11590
- FHC 2015 Round 1 - Autocomplete

String matching

Is x (length m) a substring of S (length n)?

- Naive approach: O(m*n)

String matching: naive approach

```
int matching(string text, string pattern){
  for(int i = 0; i <= text.size() - pattern.size(); ++i){
    bool found = true;
    for(int j = 0; j < pattern.size(); ++j){
      if(text[i+j]!=pattern[j]){
        found = false;
        break;
      }
    }
    if(found){
      return i;
    }
  }
  return -1;
}
```

String matching

Better: Knuth-Morris-Pratt -> O(M+N)

Two steps

1. construct ‘back table’ for pattern
2. iterate over text quickly using back table

KMP - Preprocessing

```
vector<int> preprocess(string pattern){
  vector<int> back_table(pattern.size()+1, 0);
  int j = -1;
  back_table[0] = -1;
  for(int i = 0; i < pattern.size(); ++i){
    while(j >= 0 && pattern[i] != pattern[j]){
      j = back_table[j];
    }
    ++j;
    back_table[i+1] = j;
  }
  return back_table;
}
```

KMP - processing

```
int matching(string text, string pattern){
  vector<int> back_table = preprocess(pattern);
  int j = 0;
  for(int i = 0; i < text.size(); ++i){
    while(j >= 0 && text[i] != pattern[j]){
      j = back_table[j];
    }
    j++;
    if(j == pattern.size()){
      return i+1-j;
    }
  }
  return -1;
}
```

KMP - Performance

Time: O(M + N)

Space: O(M)

KMP - Exercises

- Codeforces R299_1 B
- UVa 11475

Segment trees

See previous trainings

Many queries for range sum (or similar), while elements in the data array are frequently updated.

Exercise: UVa 11297

All-Pairs Shortest Paths

- N times Dijkstra
- Floyd-Warshall (Proof by induction)

```
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if
```

Exercises

Dijkstra: UVa 11338

Floyd-Warshall: UVa 821