Exact Matching Part III: Ukkonen’s Algorithm

See Gusfield, Chapter 5

Visualizations from http://brenden.github.io/ukkonen-animation/
Goals for Today

• Understand how suffix links are used in Ukkonen's algorithm
• Reason about properties of the algorithm, and correctness
Suffix Tree and Implicit Suffix Tree for abxa$
Ukkonen’s Algorithm Review

Build an Implicit Suffix Tree for $S[1..m]$ in $O(m)$ Time

Notation: Let $I_i$ be the implicit suffix tree for prefix $S[1,...,i]$

Construct $I_1$  // Phase 1
For $i$ from 1 to $m-1$  // Phase $i+1$: Build $I_{i+1}$ from $I_i$
  For $j$ from 1 to $i+1$  // Extension $j$ of Phase $i+1$
    Ensure that a path labeled $S[j..i+1]$ is in the tree
Ukkonen’s Algorithm: Extension j of Phase i+1

Ensure that a path labeled $S[j, i+1]$ is in the tree

1. Find the end of path $S[j..i]$

2. Apply the appropriate suffix extension rule
Rule 1: Path $S[j..i]$ ends at a leaf. Add $S[i+1]$ to the end of the label on that leaf edge

Rule 2: At least one path continues from the end of $S[j..i]$, but no such path starts with $S[i+1]$
   - If $S[j..i]$ ends inside an edge, create a node at that endpoint. Let $w$ be the node at the end of the path $S[j..i]$
   - Create a new leaf node numbered “$j$”
   - Create an edge labeled $S[i+1]$ from $w$ to leaf $j$

Rule 3: Some path from the end of $S[j..i]$ starts with character $S[i+1]$. Do nothing
Ukkonen’s Algorithm Examples

Exercise: Follow the steps of Ukkonen’s algorithm to construct suffix trees for the following strings

• aab
• abababababab
• aababab
Ukkonen’s Algorithm Examples

**Exercise:** Follow the steps of Ukkonen’s algorithm to construct suffix trees for the following strings

- aab
- abababababab
- aababab

*Implicit tree \( I_7 \) for aababab.*

*Numbers inside nodes indicate order of creation.*

*Leaf numbers are in boxes.*
Claim: The tree constructed at the end of phase $i+1$ is the implicit tree $I_{i+1}$. That is:

- Each internal node, other than the root, has at least two children
- Each edge is labeled with a nonempty substring of $S[1..i+1]$
- No two edges out of a node have edge labels beginning with the same character
- Every suffix of $S[1..i+1]$ labels a path of the tree
- The concatenation of edge labels on the path from the root to leaf "j" is $S[j..i+1]$
Ukkonen’s Algorithm: Suffix Links

Since the naïve implementation takes $\Theta(m^3)$ time, we augment the algorithm to create and use suffix links

A suffix link is a pointer from $v$ to $v'$, where

- $v$ is an internal node with label $x\alpha$
- $v'$ is an internal node with label $\alpha$

where $x$ is a character and $\alpha$ is a string
Ukkonen’s Algorithm: Suffix Links

Since the naïve implementation takes $\Theta(m^3)$ time, we augment the algorithm to create and use suffix links.

A suffix link is a pointer from $v$ to $v'$, where
- $v$ is an internal node with label $x\alpha$
- $v'$ is an internal node with label $\alpha$
where $x$ is a character and $\alpha$ is a string.

In addition to maintaining suffix links, the algorithm also maintains a link to leaf node 1.
Ukkonen’s Algorithm: Extension j of Phase i+1

Ensure that a path labeled $S[j, i+1]$ is in the tree

// Revised algorithm that creates and uses suffix links

1. Find the end of path $S[j..i]$

2. Apply the appropriate suffix extension rule
Ukkonen’s Algorithm: Extension j of Phase i+1

Ensure that a path labeled S[j,i+1] is in the tree

// Revised algorithm that creates and uses suffix links

1. Find the end of path S[j..i] (using the link to leaf node 1, or suffix links, plus the end of path S[j-1..i])

2. Apply the appropriate suffix extension rule
Ukkonen’s Algorithm: Extension $j$ of Phase $i+1$

Ensure that a path labeled $S[j,i+1]$ is in the tree

// Revised algorithm that creates and uses suffix links

1. Find the end of path $S[j..i]$ (using the link to leaf node 1, or suffix links, plus the end of path $S[j-1..i]$)

2. Apply the appropriate suffix extension rule. If a new internal node (with label $S[j..i]$) is created in this step, keep a link to it
Ukkonen’s Algorithm: Extension j of Phase i+1

Ensure that a path labeled $S[j,i+1]$ is in the tree

1. Find the end of path $S[j..i]$ (using the link to leaf node 1, or suffix links, plus the end of path $S[j-1..i]$)

2. Apply the appropriate suffix extension rule. If a new internal node (with label $S[j..i]$) is created in this step, keep a temporary link to it

3. If a new internal node $w$ (with label $S[j-1..i]$) was added to the tree on extension $j-1$, add the suffix link from $w$ (and delete the temporary link to $w$)
Ukkonen’s Algorithm: Extension j of Phase i+1

1. Find the end of path $S[j..i]$

   - If $j=1$: use the 1-link (path $S[1..i]$ ends at leaf 1)
   - If $j > 1$:
     
     // In extension $j-1$, we located the end of path $S[j-1..i]$
     
     – Find the first node $v$ at or above the end of path $S[j-1..i]$. Let $\gamma$ denote the string from $v$ to the end of $S[j-1..i]$
     
     – If $v$ is the root, follow the (unique) path $\gamma$
     
     – If $v$ is not the root, traverse the suffix link from $v$, say to $v'$ and follow the (unique) path $\gamma$ from $v'$
2. Apply the appropriate suffix extension rule

- **Rule 1**: Path $S[j..i]$ ends at a leaf. Add $S[i+1]$ to the end of the label on that leaf edge

- **Rule 2**: At least one path continues from the end of $S[j..i]$, but no such path starts with $S[i+1]$
  - If $S[j..i]$ ends inside an edge $(u1,u2)$, create a node $w$ at that endpoint (with edges $(u1,w), (w,u2)$)
  - Create a new leaf node numbered “$j$”
  - Create an edge labeled $S[i+1]$ from $w$ to leaf $j$

- **Rule 3**: Some path from the end of $S[j..i]$ starts with character $S[i+1]$. Do nothing
Ukkonen’s Algorithm: Extension j of Phase i+1

3. If a new internal node $w$ was added to the tree on extension $j-1$, add the suffix link from $w$

- Let the label of $w$ be $x\alpha$ ($=S[j-1..i]$)
- Let $w'$ be the node with label $\alpha$ ($w'$ is created in extension $j$ of the phase if not already in the tree)
- Add the link $(w,w')$
Ukkonen’s Algorithm Examples

Let’s see how suffix links are created and used, by doing some more examples

• ababb
Ukkonen’s Algorithm Examples

Let’s see how suffix links are created and used, by doing some more examples

• abbbba
Ukkonen’s Algorithm Examples

Let’s see how suffix links are created and used, by doing some more examples

• abbba

Note that the suffix link points to an ancestor (Here and in future illustrations, we do not include suffix links to the root)
Ukkonen’s Algorithm Examples

Let’s see how suffix links are created and used, by doing some more examples

• aabababb

Implicit tree $I_7$ for aababab.
(In this and future trees, suffix links from internal nodes to the root are not shown.)

Simulate Phase 8 of the algorithm to obtain the tree $I_8$
Ukkonen’s Algorithm Examples

• aababababababaa
Ukkonen’s Algorithm Examples

- aabababababaabababababab
  (try this one yourself on the visualization website)
Correctness: Suffix Link Properties

• Later we’ll use properties of suffix links to reason about the efficiency of Ukkonen’s algorithm
• We’ll describe and prove these properties now
Correctness: Suffix Link Properties

Claim 1: At the end of phase \(i+1\), every internal node of \(I_i\) (except for the root) has a suffix link.

We’ll use the following claim to prove Claim 1.

Claim 2: Let \(1 \leq j \leq i \leq m\). If \(S[j..i]c\) is in the tree at the start of extension \(j\) of phase \(i+1\), then so is \(S[j+1..i]c\).
Correctness: Suffix Link Properties

Claim 1: At the end of phase $i+1$, every internal node of $I_i$ (except for the root) has a suffix link.

Proof: By induction on $i$.

Base case $i=0$: In this case the statement is trivially true since $I_1$ has no suffix links.
Inductive step: i > 0. Suppose a new internal node w with label S[j..i] is created at extension j of phase i+1, by extension Rule 2. Note that no new internal node is created in extension i+1, so j ≤ i.

Let w break the edge (u1, u2). Let c be the first character of (w, u2); since Rule 2 is applied, c is not S[i+1].

By Claim 2, S[j+1..i]c is a path in the tree. Therefore, in extension j+1, if S[j+1..i] does not label an internal node or the root, Rule 2 will be applied again and another internal node w' will be added with label S[j+1..i]. And so the suffix link (w, w') can be added.
Correctness: Suffix Link Properties

Claim 2: Let $1 \leq j < i \leq m$. If $S[j..i]c$ is in the tree at the start of extension $j$ of phase $i+1$, then so is $S[j+1..i]c$.

Proof: Case 1: $S[j..i]c$ was added to the tree in phase $i'$ for some $i' < i+1$. Then $S[j+1..i]c$ was also added to the tree in phase $i'$.

Case 2: $S[j..i]c$ was added to the tree in phase $i+1$. Given its length, it must have been added in extension $j-1$. So it must be that $S[j..i]c = S[j-1..i]$ In this case, $S[j-1] = S[j] = ... S[i] = c$. And so $S[j+1..i]c = S[j..i]$, which we know is already in the tree.
Correctness: Suffix Link Properties


Diagram showing the tree structure with nodes and edges.
Correctness: Suffix Link Properties

Claim 3: Let \((w,w')\) be any suffix link traversed during Ukkonen’s algorithm. At that moment, the node-depth of \(w\) is at most one greater than the node depth of \(w'\).

Proof: Consider when \((w,w')\) is traversed.

• First, the suffix link from any ancestor \(u\) of \(w\) goes to an ancestor \(u'\) of \(w'\); this follows since the label of \(u\) is a prefix of \(w\), and also the label of \(u'\) is a prefix of \(w'\).

• Second, links from distinct ancestors of \(w\) must go to distinct ancestors of \(w'\), because edge labels have length at least 1.

• Thus there is a 1-1 correspondence between the internal nodes on the path to \(w\) except for the root, and those on the path to \(w'\). The claim follows.