Indexing Factors with Gaps

M. Sohel Rahman and Costas S. Iliopoulos

Algorithm Design Group

Department of Computer Science

King's College London

www.dcs.kcl.ac.uk/adg

Definitions:Text/String

- A sequence of zero or more symbols from an alphabet Σ.
- Denoted by $\mathcal{T}[1..n] = \mathcal{T}_1\mathcal{T}_2\ldots\mathcal{T}_n$, where $\mathcal{T}_i \in \Sigma$ for $1 \leq i \leq n$.
- The length of $\mathcal{T}$ is denoted by $|\mathcal{T}| = n$.
- $\overleftarrow{\mathcal{T}}$ denotes the reverse of the string $\mathcal{T}$.
- Example: $T = ACAAGTGCA$ is a text of length 9
- So, $\overleftarrow{\mathcal{T}} = ACGTGAACA$

Definitions:Factors

$T = ACAAGTGCA$

- A string w is a factor of $\mathcal{T}$ if $\mathcal{T} = uwv$ for $u, v \in \Sigma^*$
- In this case, the string w occurs at position $|u| + 1$ in $\mathcal{T}$.
- w is denoted by $\mathcal{T}[|u| + 1..|u| + |w|]$.
- Example: $w = AAG = \mathcal{T}[3..5]$ is a factor of $\mathcal{T}$
- A k-factor is a factor of length k.
- Example: $w = AAG$ is a 3-factor of $\mathcal{T}$

Definitions: Prefix and Suffix

$T = ACAAGTGCA$

- A prefix of $\mathcal{T}$ is a factor $\mathcal{T}[1..y]$, $1 \leq y \leq n$.
- Example: ACA is a prefix of $\mathcal{T}$.
- ith prefix is the prefix ending at position i.
- Hence ACA is the 3rd prefix.
- A suffix of $\mathcal{T}$ is a factor $\mathcal{T}[x..n]$, $1 \leq x \leq n$.
- Example: $TGCA$ is a suffix of $\mathcal{T}$
- ith suffix is the suffix starting at position i.
- Hence $TGCA$ is the 6th suffix.

Definitions: Gapped Factors

$T = ACAAGTGCA$

- *gapped-factor* is a concatenation of two factors separated by a gap i.e. a block of don't care characters
- A don't care character '$*$' can match any character $a \in \Sigma$ and $* \notin \Sigma$.
- $CA***G$ is a gapped factor of $\mathcal{T}$

		1	2	3	4	5	6	7	8	9
$\mathcal{T}$	=	A	C	A	A	G	T	G	C	A
			C	A	$*$	$*$	$*$	G		

Definitions: $(k-d-k')$-Gapped Factors

- A $(k-d-k')$-gapped-factor is a gapped-factor where the length of the two sub-factors are, respectively, k and k' and the length of gaps is d.
- A $(k-d-k')$-gapped-factor is $X = X_f *^d X_\ell$, where $X_f = X[1..k]$, $X_f = X[k+d+1..|X|]$ and $*^d$ denotes the concatenation of d don't care characters.
- $X = CA *^3 G$ is a $(2-3-1)$-gapped factor of $\mathcal{T}$

Definitions: Gapped Factors Occurrence

- A $(k-d-k')$-gapped-factor X is said to occur at position i of a string Y if and only if:
  1. we have an occurrence of X_f at position i and
  2. we have an occurrence of X_ℓ at position $i+k+d$.
- The position i is said to be an occurrence of X in $\mathcal{T}$.
- We denote by Occ_X^T the set of occurrences of X in $\mathcal{T}$.

Example: Gapped Factors Occurrence

		1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathcal{T}$	=	A	G	G	A	C	C	G	G	G	T	T	G	A
X	=			G	A	C	$*$	$*$	$*$	G	T	T	G	A
				$\leftarrow - -$	k	$- - \rightarrow$				$\leftarrow - -$		k'		$- - \rightarrow$
X	=												G	A

- We have GAC at position 3
- we have $GTTGA$ at position 3 + k + d = 3 + 3 + 3 = 9
- So we have an occurrence of $GAC *^3 GTTGA$ at position 3.

Our Goal

- Present an efficient data structure to index gapped factors

The Main Idea

Goal: Finding the Occurrence of

$X = X_f *^d X_\ell = AC**GTG$ in $\mathcal{T} = ACACACGTGTGTG$

1: Compute $Occ^T_{X_f}$ {$Occ^T_{X_f} = \{1, 3, 5\}$}

2: Compute $Occ^T_{X_\ell}$ {$Occ^T_{X_\ell} = \{7, 9, 11\}$}

3: **for** $i \in Occ^T_{X_\ell}$ **do**

4: $\quad i = i - |X_f| - d$ {$Occ^T_{X_f} = \{3, 5, 7\}$}

5: **end for**

6: Compute $Occ^T_X = Occ^T_{X_f} \bigcap Occ^T_{X_\ell}$ {$Occ^T_X = \{3, 5\}$}

7: **return** Occ^T_X {$Occ^T_X = \{3, 5\}$}

Gapped Factors Occurrences

		1	2	3	4	5	6	7	8	9	10	11	12	13
$\mathcal{T}$	=	A	C	A	C	A	C	G	T	G	T	G	T	G
X	=			A	C	$*$	$*$	G	T	G				
X	=					A	C	$*$	$*$	G	T	G		

Occurrences of $AC = \{1, 3, 5\}$
Occurrences of $GTG = \{7, 9, 11\} \dashrightarrow$ shift $\dashrightarrow \{3, 5, 7\}$
Intersection: $\{1, 3, 5\} \bigcap \{3, 5, 7\} = \{3, 5\}$

The Main Idea

- Find the occurrences of the first factor (X_f).
- Find the occurrences of the last factor (X_ℓ)).
- Perform a bit of shifting and then compute the intersection.

However, we need to do it now in the context of indexing!

GFI (Gapped Factor Index) Construction Algorithm

STEP 1:

- Build a suffix tree $ST_{\mathcal{T}}$ of $\mathcal{T}$.
- Preprocess $ST_{\mathcal{T}}$ such that each internal node stores the range of leaves it corresponds to.

Why?: We will find the occurrence of X_ℓ using $ST_{\mathcal{T}}$.

STEP 2:

- Build a suffix tree $ST_{\overleftarrow{\mathcal{T}}}$ of $\overleftarrow{\mathcal{T}}$.
- The label of each leaf is replaced by $(n+1) - actual\_label + d + 1$.
- Preprocess $ST_{\overleftarrow{\mathcal{T}}}$ such that each internal node stores the range of leaves it corresponds to.

Why?: We will find the occurrence of $\overleftarrow{X_f}$ using $ST_{\overleftarrow{\mathcal{T}}}$. Finding these occurrences will be equivalent to finding the occurrences of X_f according to the desired shift.

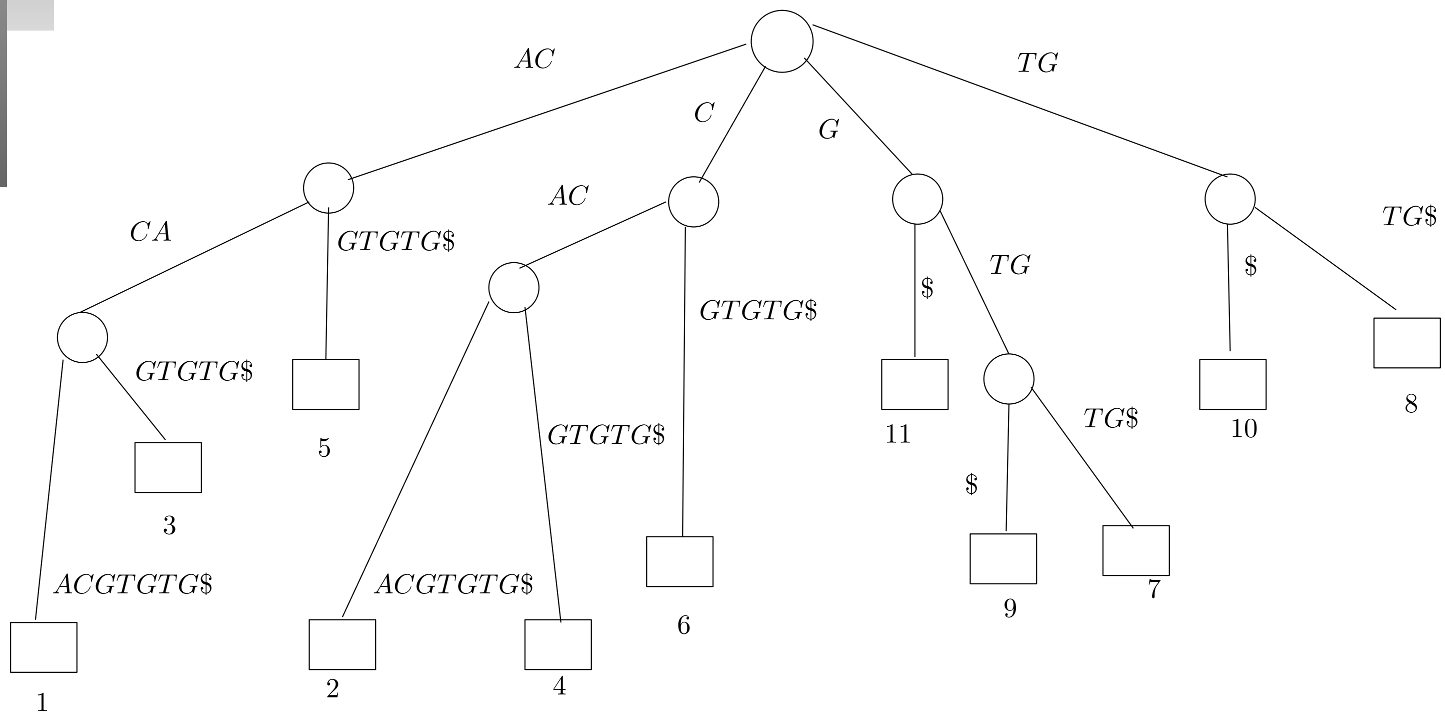
GFI (Gapped Factor Index) Construction Algorithm

STEP 3:

- Build a data structure to facilitate the intersection

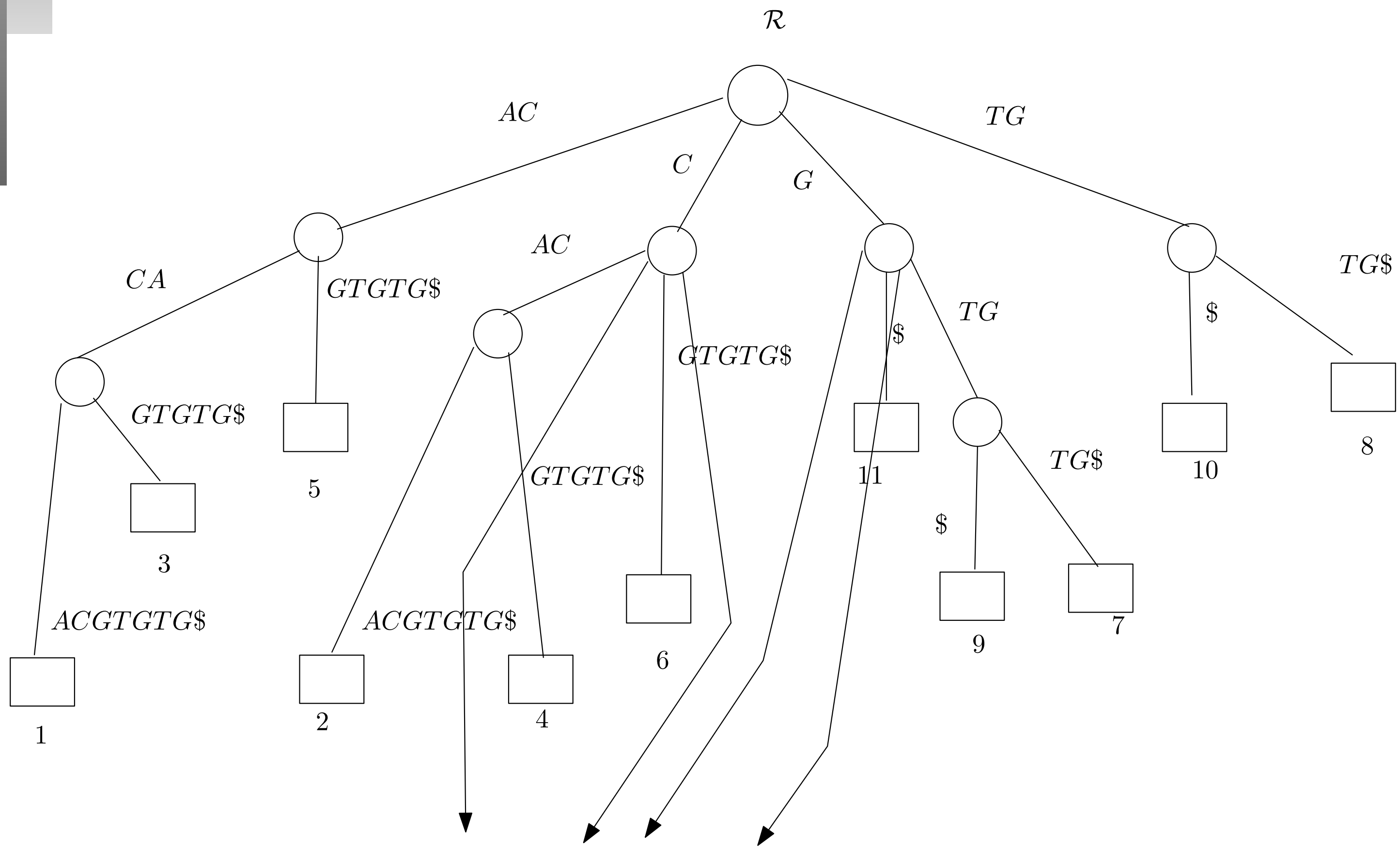
First Step

$$\mathcal{T} = ACACACGTGTG$$



First Step

$$\mathcal{T} = ACACACGTGTG$$

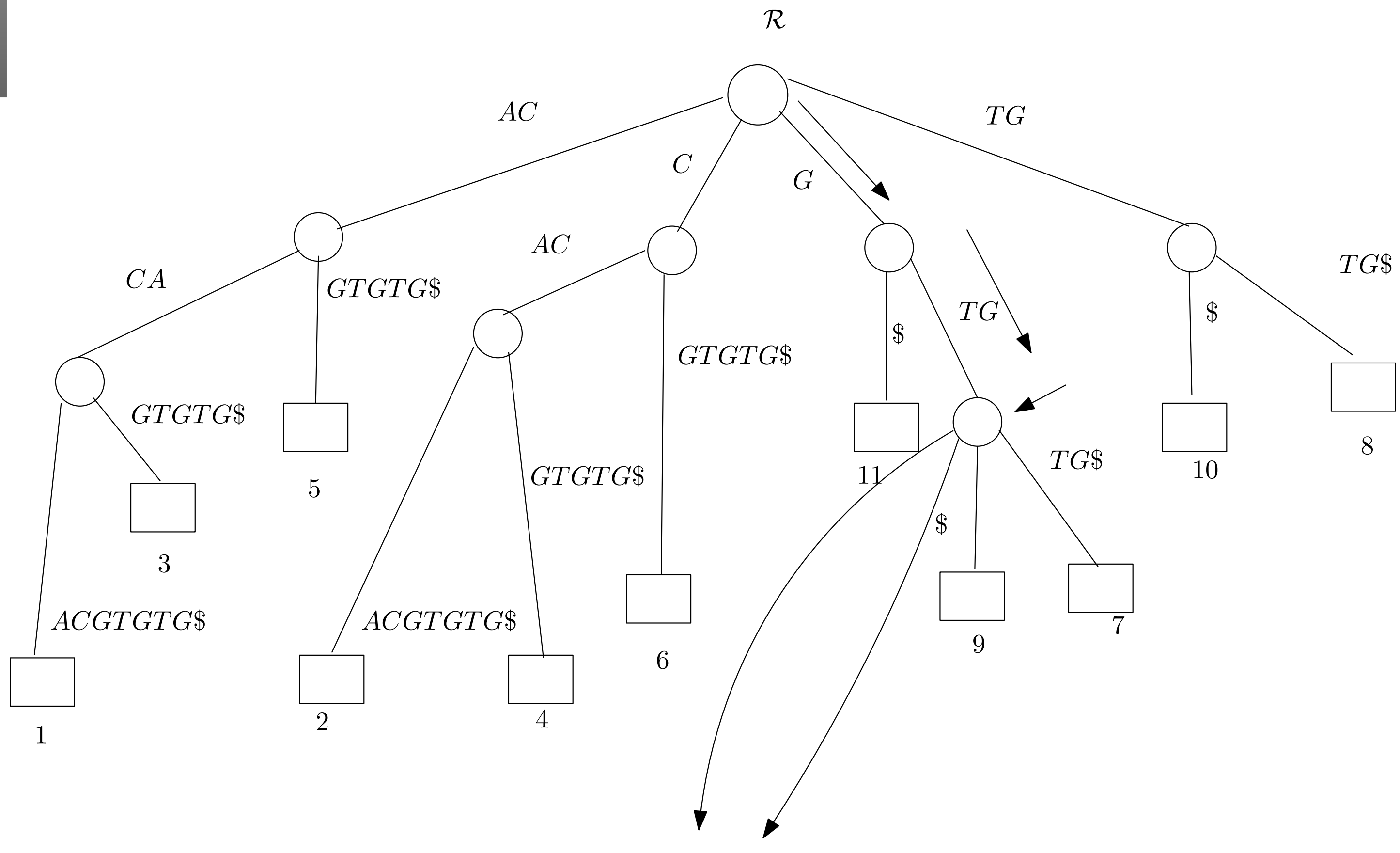


$$\mathcal{L} = 1 \to 3 \to 5 \to 2 \to 4 \to 6 \to 11 \to 9 \to 7 \to 10 \to 8$$

Why the First Step?

$$\mathcal{T} = ACACACGTGTG, X = AC**GTG$$

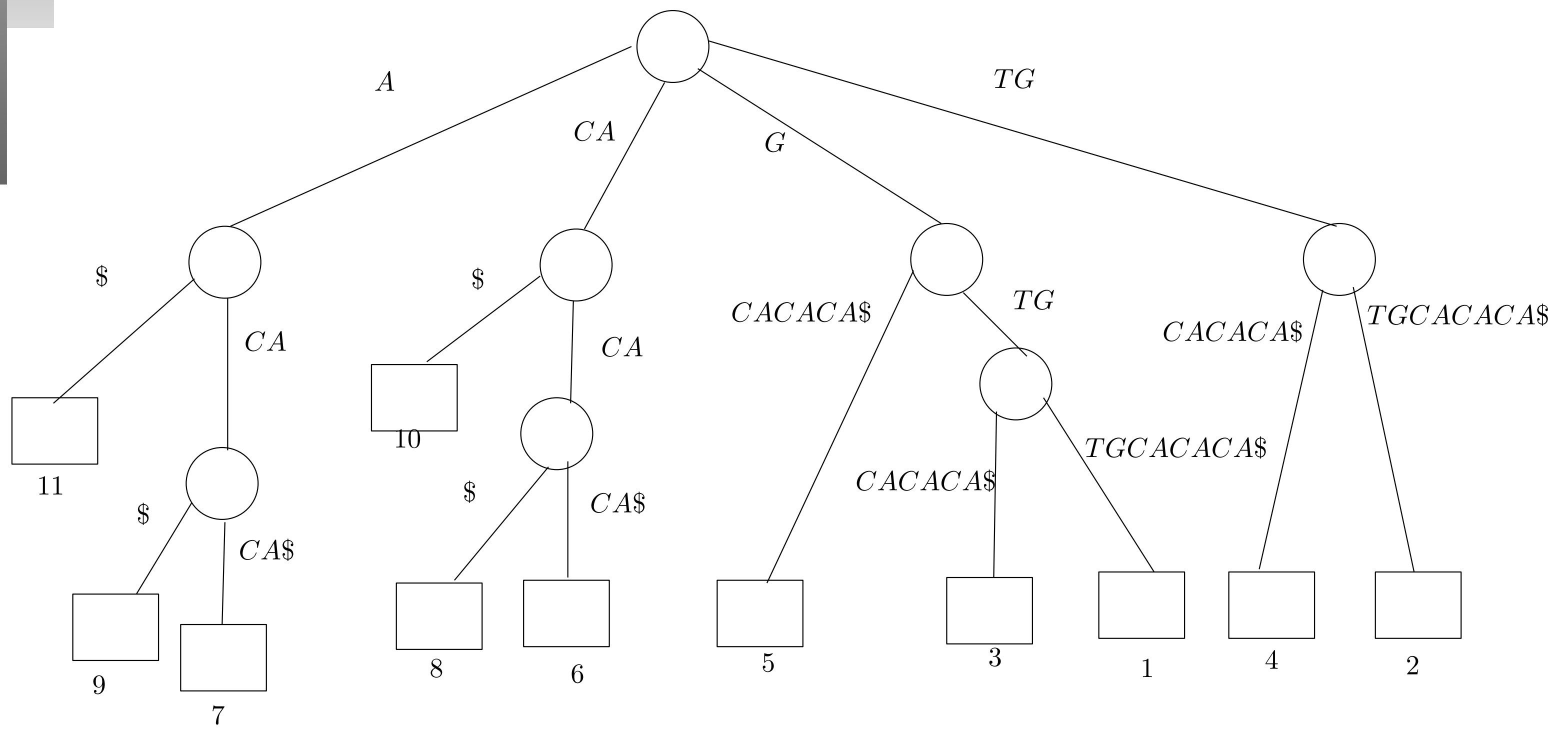
Used to find the occurrences of $X_\ell = GTG$.



$\mathcal{L} = 1 \to 3 \to 5 \to 2 \to 4 \to 6 \to 11 \to 9 \to 7 \to 10 \to 8$

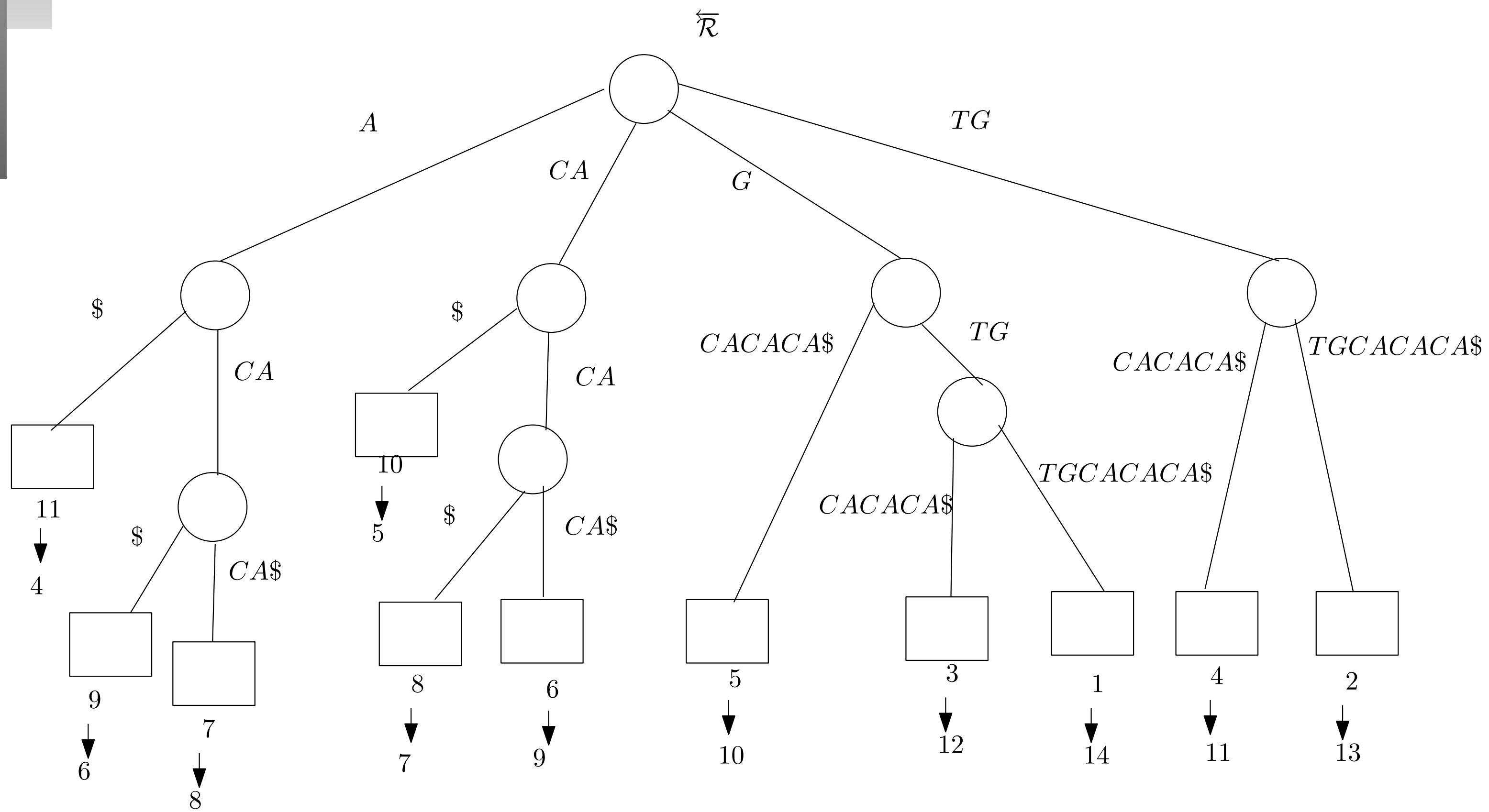
Second Step

$$\overleftarrow{\mathcal{T}} = GTGTGCACACA$$



Second Step

$$\overleftarrow{\mathcal{T}} = GTGTGCACACA$$

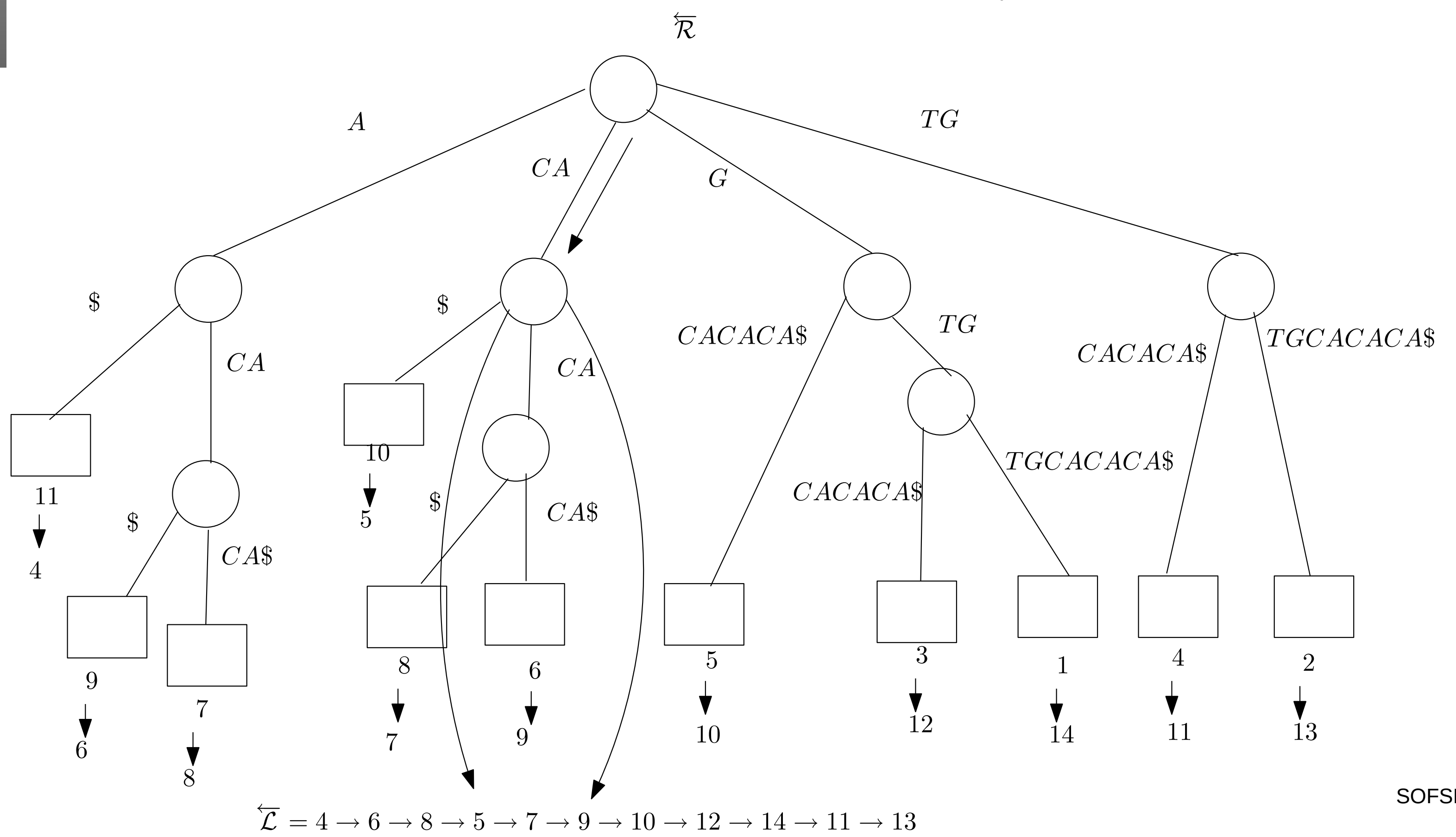


$$\overleftarrow{\mathcal{L}} = 4 \to 6 \to 8 \to 5 \to 7 \to 9 \to 10 \to 12 \to 14 \to 11 \to 13$$

Why the Second Step?

$$\overleftarrow{\mathcal{T}} = GTGTGCACACA, X = AC**GTG$$

Used to find the occurrences of $\overleftarrow{X_f} = CA$.



Third Step

Now we have the followings:

1. We have an array $\mathcal{L}$ and two indices i, j
2. we have an array $\overleftarrow{\mathcal{L}}$ and two indices k, l
3. We want the intersection of elements of $\mathcal{L}[i..j]$ and $\overleftarrow{\mathcal{L}}[k..l]$.

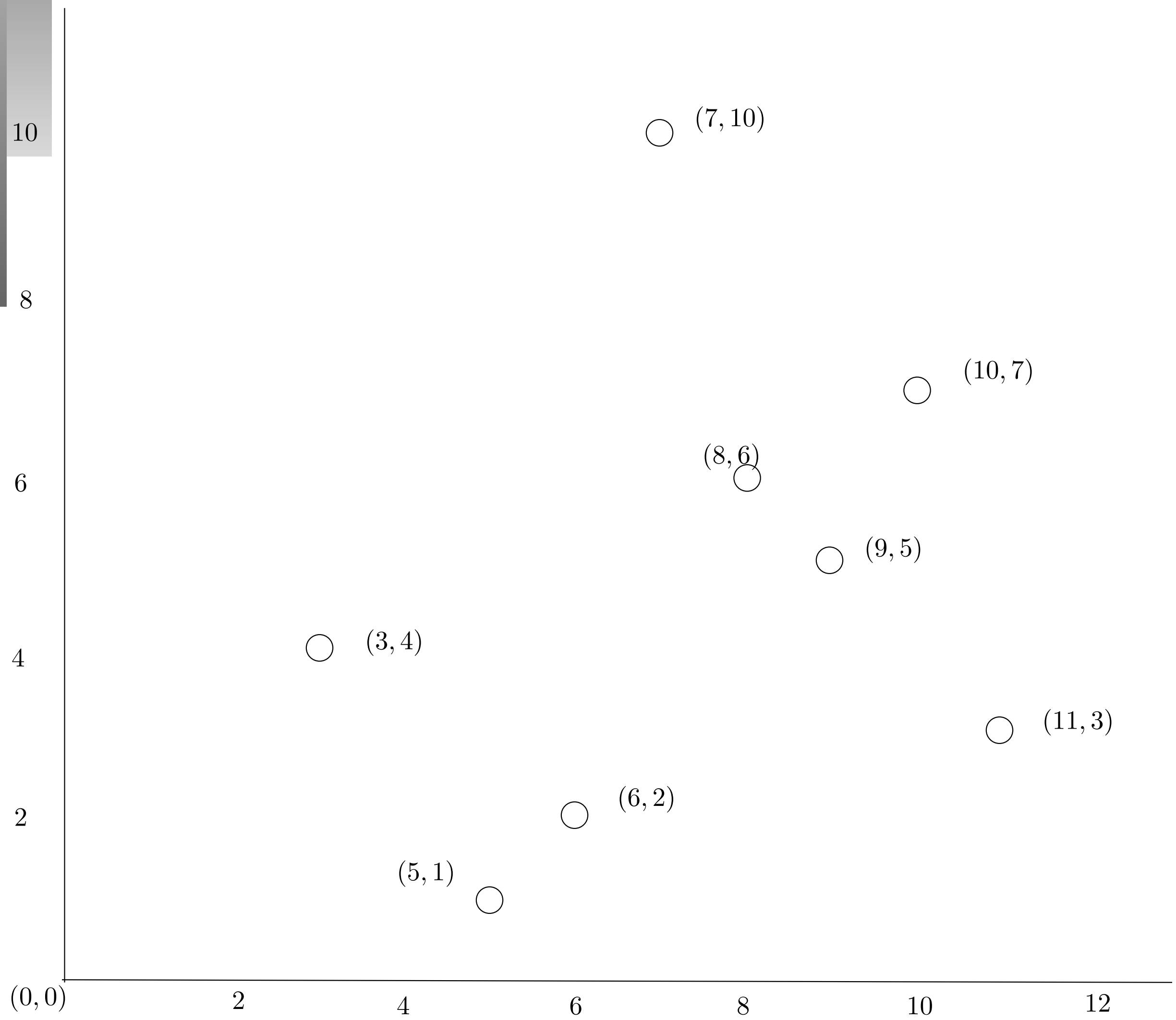
Our Goal: Preprocess $\overleftarrow{\mathcal{L}}$ and $\overleftarrow{\mathcal{L}}$ to give the Range Intersection.

Third Step

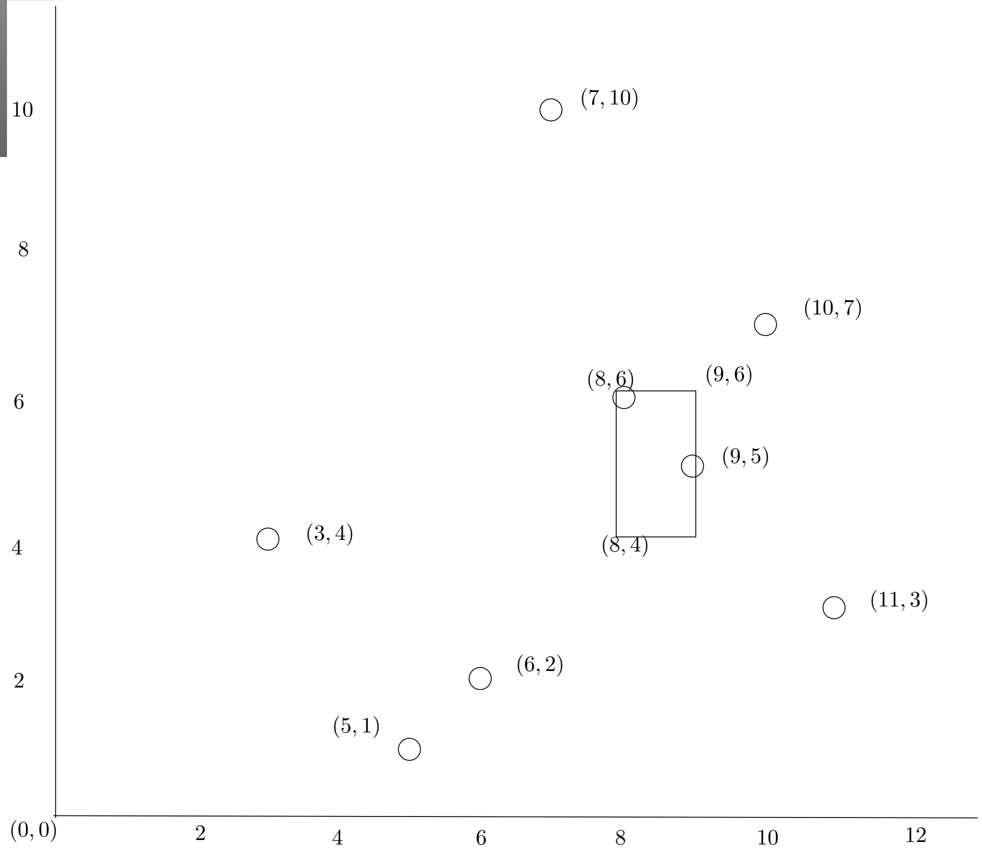
Transformation to Range Search Problem On Grid:

		1	2	3	4	5	6	7	8	9	10	11
$\mathcal{L}$	=	1	3	5	2	4	6	11	9	7	10	8
$\overleftarrow{\mathcal{L}}$	=	4	6	8	5	7	9	10	12	14	11	13
	=	$-$	$-$	$-$	$5, 1$	$3, 4$	$6, 2$	$9, 5$	$11, 3$	$8, 6$	$10, 7$	$7, 10$

Third Step



We have $i, j \equiv 8, 9$ and $k, l \equiv 4, 6$. So we look for the points in the rectangle $(i, k) \times (j, l) \equiv (8, 4) \times (9, 6)$



Third Step

Final result:

								$\checkmark$		$\checkmark$		
		1	2	3	4	5	6	7	8	9	10	11
$\mathcal{L}$	=	1	3	5	2	4	6	11	9	7	10	8
$\overleftarrow{\mathcal{L}}$	=	4	6	8	5	7	9	10	12	14	11	13
	=	$-$	$-$	$-$	$5,1$	$3,4$	$6,2$	$9,5$	$11,3$	$8,6$	$10,7$	$7,10$
	=	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\checkmark$	$\times$	$\checkmark$	$\times$	$\times$

Re-shifting:

$7 \longrightarrow (7 - d - |X_f|) \rightarrow 3$

$9 \longrightarrow (9 - d - |X_f|) \rightarrow 5$

Final Result

So the occurrences are 3 and 5 as can verified below:

		1	2	3	4	5	6	7	8	9	10	11
$\mathcal{T}$	=	A	C	A	C	A	C	G	T	G	T	G
X	=			A	C	$*$	$*$	G	T	G		
X	=					A	C	$*$	$*$	$*$	$*$	G

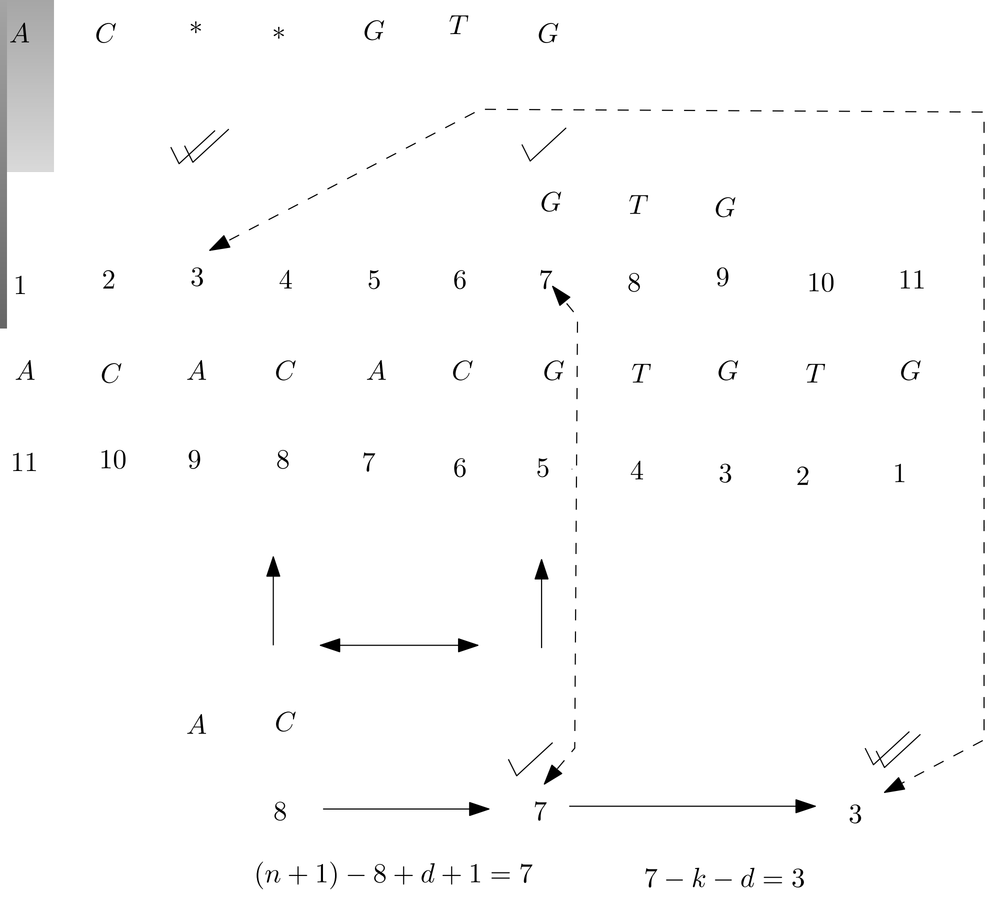
Recap: GFI Construction Steps

- Construct suffix tree of $\mathcal{T}$ and do some preprocessing to get the list $\mathcal{L}$
- Construct suffix tree of $\overleftarrow{\mathcal{T}}$ and do some preprocessing to get the list $\overleftarrow{\mathcal{L}}$
- Preprocess for Range Search on the Grid for $\mathcal{L}$ and $\overleftarrow{\mathcal{L}}$

Recap: Search Steps

- Find the occurrences of X_ℓ implicitly as two pointers i, j
- Find the (shifted) occurrences of $\overleftarrow{X_f}$ implicitly as two pointers k, ℓ
- Find the points in the rectangle $(i, k) \times (j, l)$.

A Pictorial Description



Running Time of GFI Construction

1. Both Suffix trees construction and preprocessing on them: $O(n)$.
2. Preprocessing for the range search on Grid: $O(n \log^{1+\epsilon} n)$ (Alstrup et al.)

So total time: $O(n \log^{1+\epsilon} n)$ ($0 < \epsilon < 1$)

Search Time

1. Finding occurrences of X_ℓ: $O(|X_\ell|)$
2. Finding occurrences of X_f according to the shift: $O(|X_f|)$
3. Finding the intersection results: $O(\log\log n + K)$, where K is the number of output.

So total time: $O(m + \log\log n + Occ_X^{\mathcal{T}})$

Previous Work and Comparison

Peterlongo et. al gave a data structure, GFT, to index gapped factor:

- Construction cost: $O(\Sigma n)$
- Search Cost: $O(m + Occ_X^{\mathcal{T}})$

• *Previous Work and Comparison*

- Construction cost of GFI is much better than that of GFT
- Search Cost GFI is slightly worse ($\log \log n$) than that of GFT
- GFT is fixed for k, k', d. GFI is only fixed for d.

Other Results in the paper

- Extension of GFI to handle multiple strings
- Extension of GFI to handle document listing problem

End of Presentation

THANK YOU
FOR YOUR PATIENCE