Reporting Consecutive Substring Occurrences Under Bounded Gap Constraints

Gonzalo Navarro

University of Chile, Chile
gnavarro@dcc.uchile.cl

Sharma V. Thankachan

Georgia Institute of Technology, USA
sthankachan@gatech.edu

Combinatorial Pattern Matching
Ischia Island, Italy
June 29, 2015

Introduction

Problem Definition

Index a text $T[1 \ldots n]$ such that whenever a pattern $P[1 \ldots p]$ and an interval $[\alpha, \beta]$ comes as a query, we can report all pairs (i, j) of consecutive occurrences of P in T with $\alpha \leq j - i \leq \beta$.

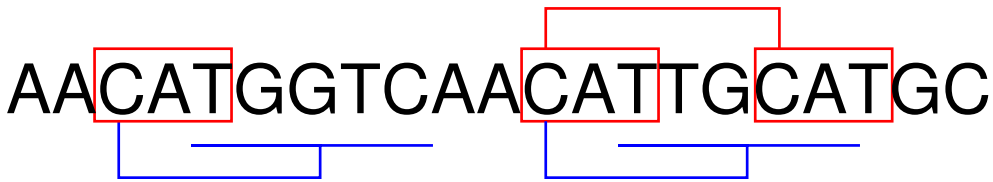
Introduction

Problem Definition

Index a text $T[1 \ldots n]$ such that whenever a pattern $P[1 \ldots p]$ and an interval $[\alpha, \beta]$ comes as a query, we can report all pairs (i, j) of consecutive occurrences of P in T with $\alpha \leq j - i \leq \beta$.

Example

$T[1 \ldots 21] = AACATGGTCAACATTGCATGC$
Occurrences of $P = CAT$ are $\{3, 12, 17\}$
Consecutive pairs of occurrences are $(3, 12)$ and $(12, 17)$
When $[\alpha, \beta] = [2, 6]$, $(12, 17)$ is an output, but not $(3, 12)$.



Introduction

Problem Definition

Index a text $T[1 \ldots n]$ such that whenever a pattern $P[1 \ldots p]$ and an interval $[\alpha, \beta]$ comes as a query, we can report all pairs (i, j) of consecutive occurrences of P in T with $\alpha \leq j - i \leq \beta$.

Example

$T[1 \ldots 21] = AACATGGTCAACATTGCATGC$
Occurrences of $P = CAT$ are $\{3, 12, 17\}$
Consecutive pairs of occurrences are $(3, 12)$ and $(12, 17)$
When $[\alpha, \beta] = [2, 6]$, $(12, 17)$ is an output, but not $(3, 12)$.

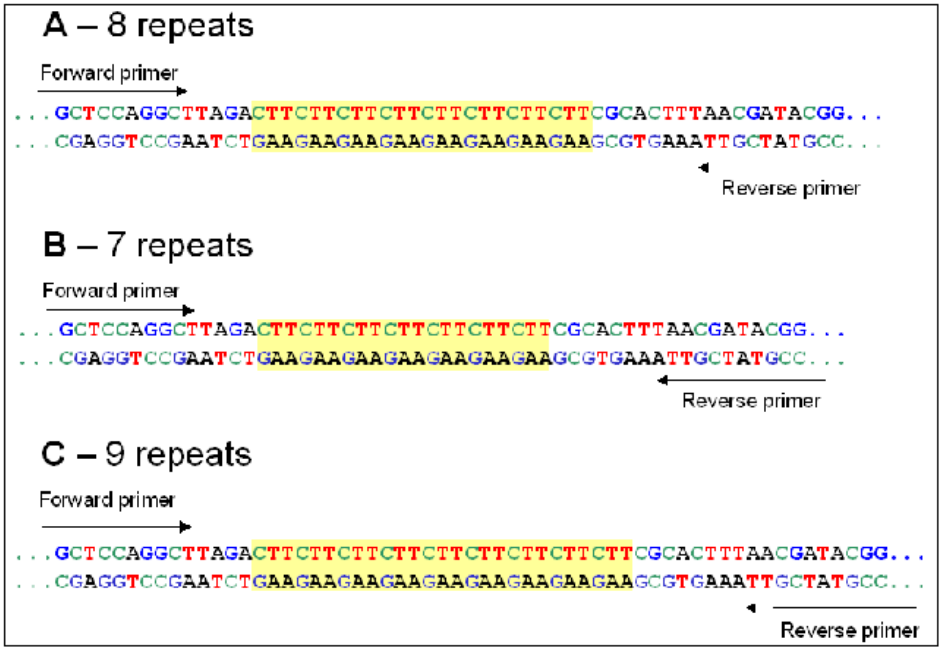
Our Result

An $O(n \log n)$ space data structure with optimal $O(p + k)$ query time, where k is the output size.

Motivations

Detecting consecutive occurrences of a pattern in a text is a problem that arises, in various forms, in computational biology applications.

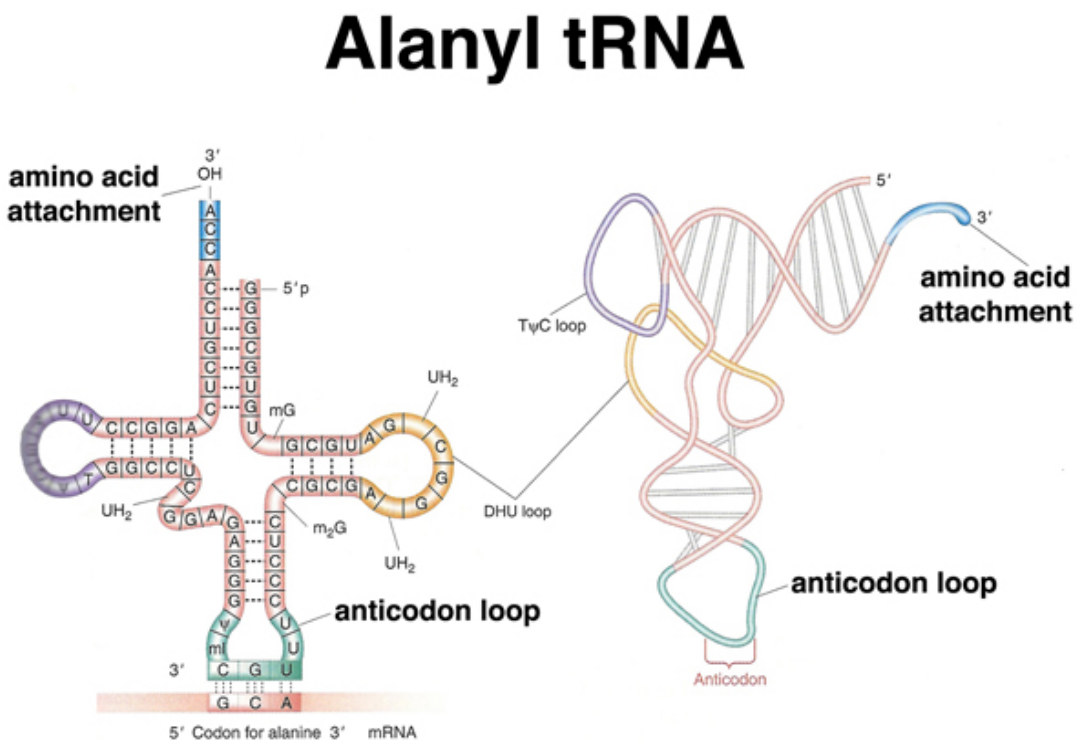
Tandem repeats



Motivations

Detecting consecutive occurrences of a pattern in a text is a problem that arises, in various forms, in computational biology applications.

Cloverleaf structures in tRNA



Studied variants of the problem

Two patterns P_1, P_2, $\alpha = \beta$ known at indexing time

Bille and Gørtz [CPM 2011]: $O(n \log^{\epsilon} n)$ space and $O(p + k)$ time

Two patterns P_1, P_2, $\alpha \neq \beta$ known at indexing time

Bille et al. [TCS 2014]: space or time exponential in β

Given an occurrence of P, find the next

Keller et al. [WADS 2007]: $O(n \log^{\epsilon} n)$ space and $O(\log \log n)$ time

Find maximal set of nonoverlapping occurrences of P

Keller et al. [WADS 2007]: $O(n \log^{\epsilon} n)$ space and $O(\log \log n + k)$ time

Studied variants of the problem

Document retrieval, $\alpha = 0$, β known at indexing time

Muthukrishnan [SODA 2002]: $O(n)$ space and $O(p + k)$ time

Document retrieval, $\alpha = 0$

Muthukrishnan [SODA 2002]: $O(n \log n)$ space and $O(p + k)$ time

Discover the patterns P that appear twice at distance $[\alpha, \beta]$

Brodal et al. [CPM 1999]: $O(n \log n + k)$ time

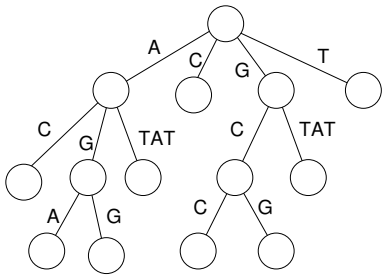
Our Solution

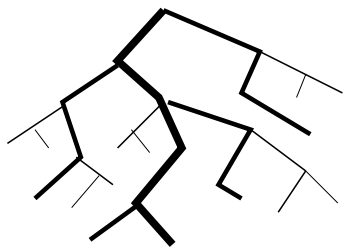
Key Idea

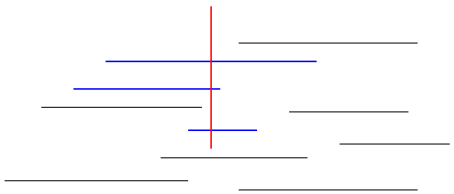
Reduce our pattern matching problem to an equivalent **geometric problem.** Specifically, to the **orthogonal segment intersection** problem

Key Techniques/Structures

- Suffix tree (ST) of T
- Heavy path decomposition of ST
- Linear space data structure for Orthogonal Segment Intersection (OSI) problem.







Our Solution

Data Structure

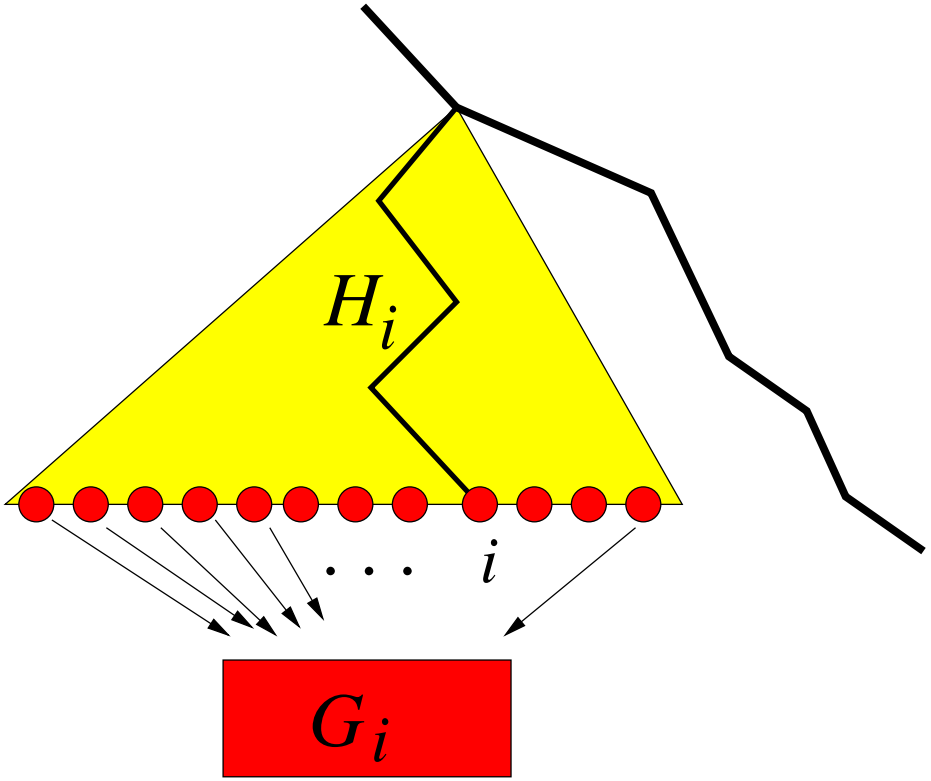
- suffix tree (ST) of T
- heavy path decomposition of ST. Recall that,
  1. # of heavy paths is n.
  2. # of heavy paths intersects with any root to leaf path is $\leq \log n$.

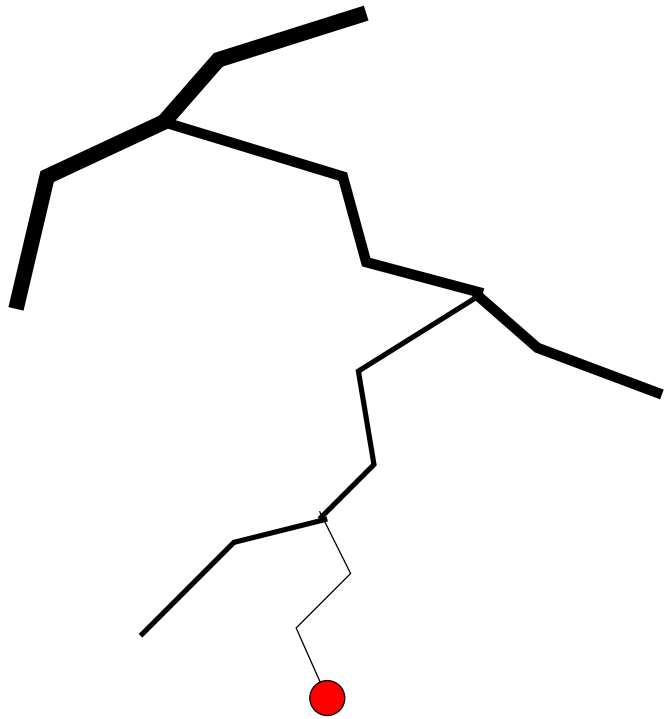
  The heavy path that leads to ith leaf (ℓ_i) is denoted by H_i.
- Sub-structures $G_1, G_2, G_3, \ldots$ corresponding to $H_1, H_2, H_3, \ldots$
- Each G_i is a linear space data structure for OSI problem.
- Space over all G_i's is $O(n \log n)$.

Processing a query (P, α, β)

- Query is type-h, if $locus(P)$ is on H_h.
- A type-h query is answered by the sub-structure G_h.

Our Solution





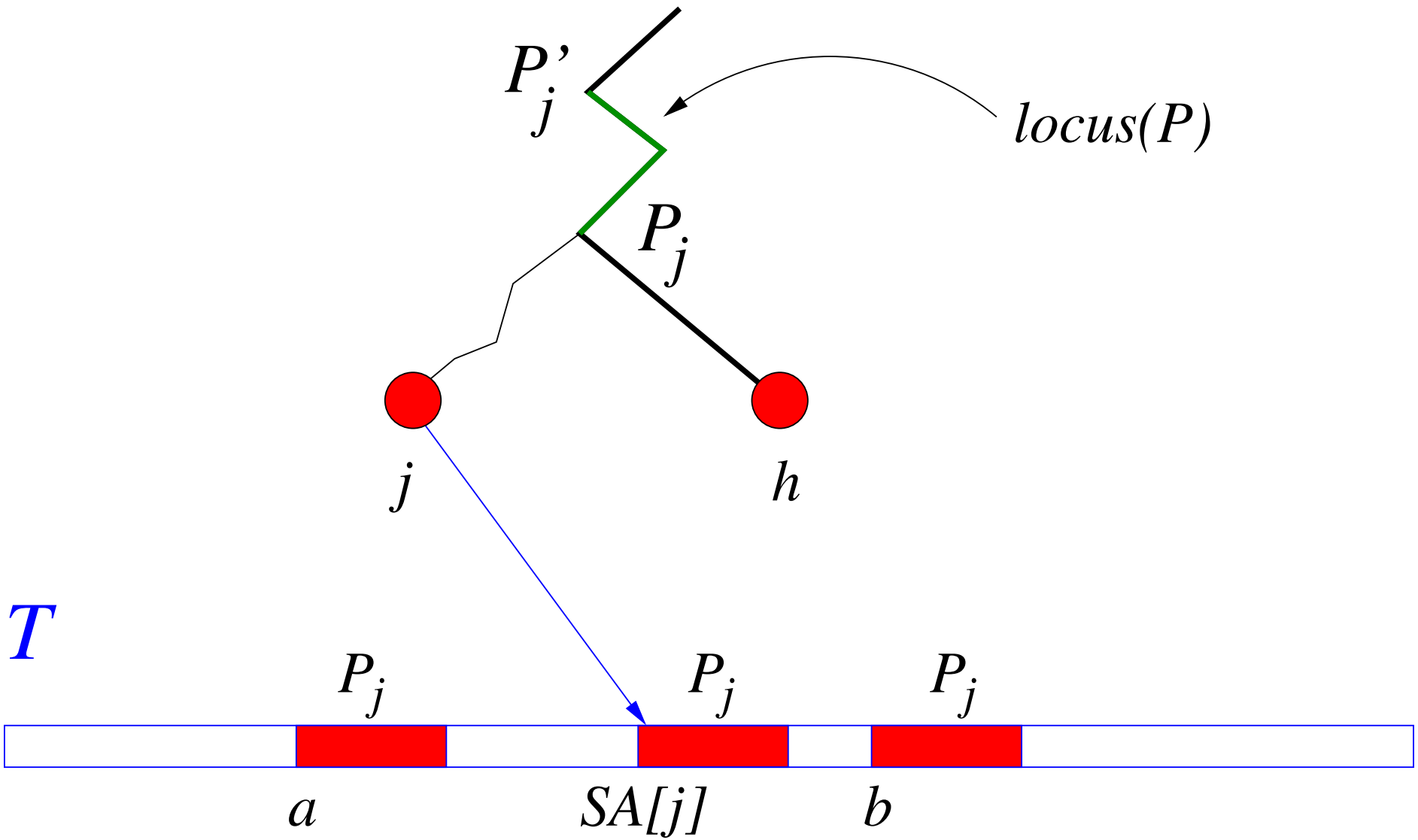
Our Solution

Key Intuition

For a query (P, α, β), which is type-h,

- Let P_j be the LCP of the suffixes corresponding to ℓ_j and ℓ_h.
- Let a and b, respectively are the last and first occurrence of P_j before and after its occurrence at $SA[j]$.
- Clearly, $(a, SA[j])$ and $(SA[j], b)$ are consecutive occurrences of P_j. However,
  - $(a, SA[j])$ is a consecutive occurrence of P iff P is not appearing in $T[(a+1)\ldots(SA[j]-1)]$.
  - Similarly, $(SA[j], b)$ is a consecutive occurrence of P iff P is not appearing in $T[(SA[j]+1)\ldots(b-1)]$.

Our Solution



Our Solution

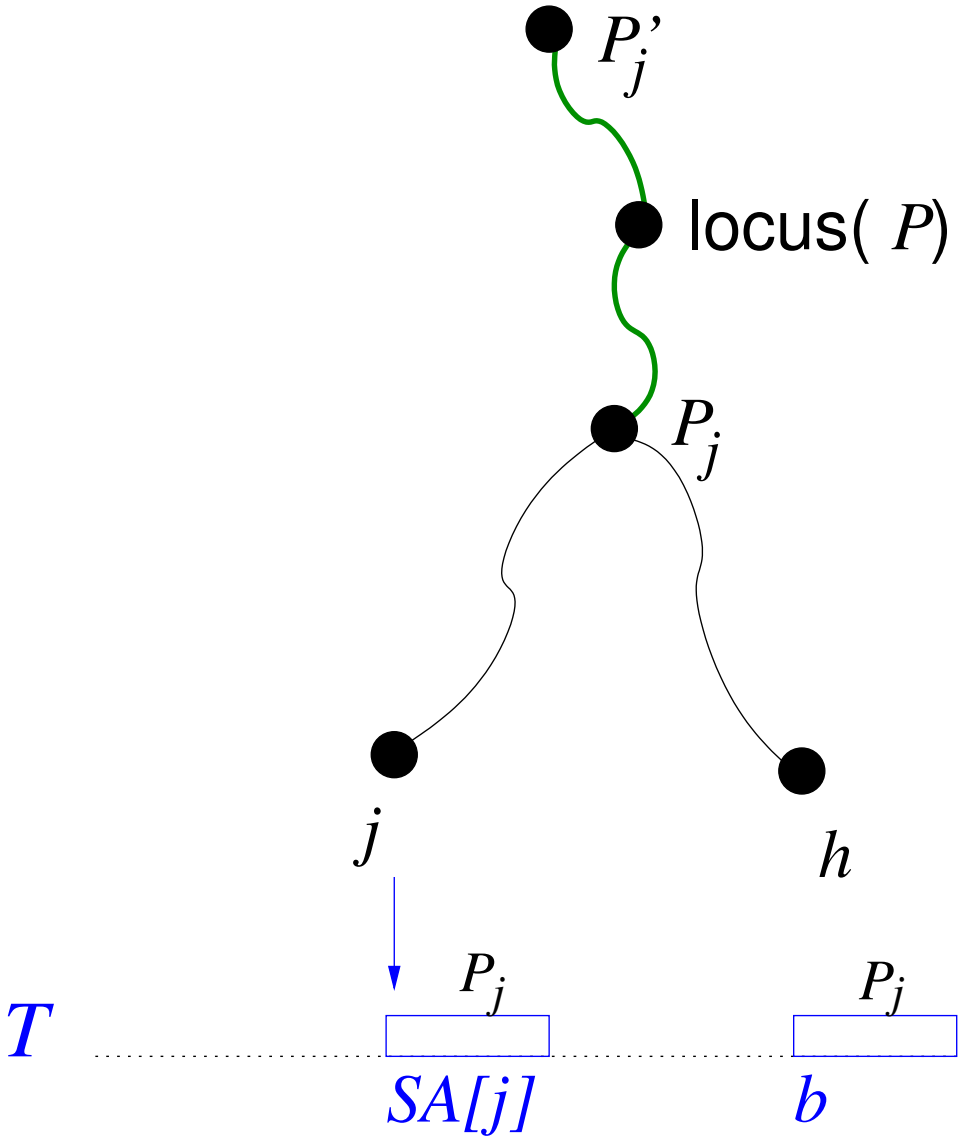
Equivalent Geometric Interpretation

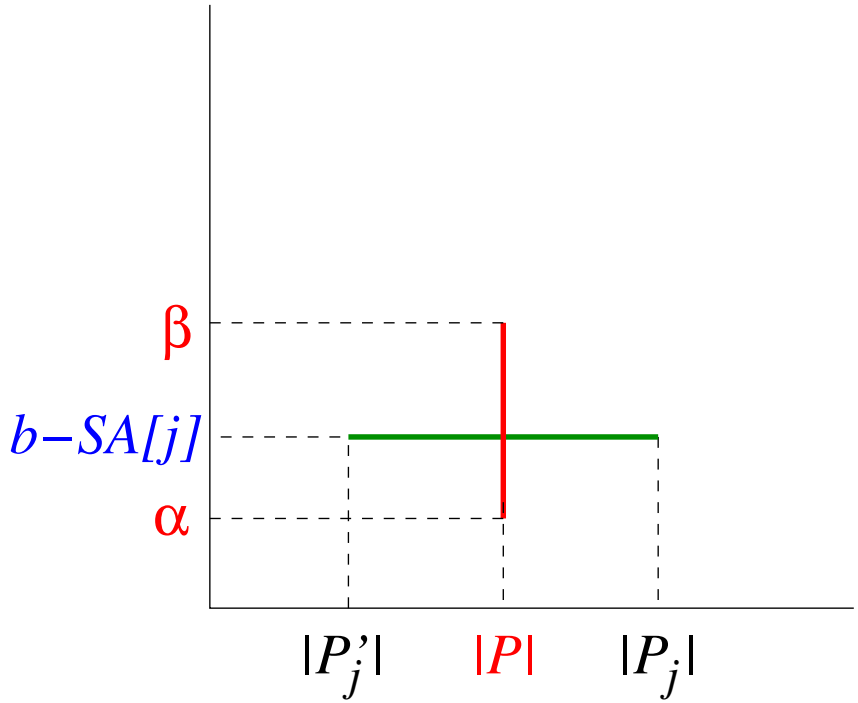
Let P'_j be the shortest prefix of P_j without any occurrence in $T[(a+1)\ldots(SA[j]-1)]$. Then,

1. $(a, SA[j])$ is a consecutive occurrence of P iff $|P| \geq |P'_j|$.
2. the condition ℓ_j is in the subtree of $locus(P)$ is true, iff $|P| \leq |P_j|$.
3. Then, $(a, SA[j])$ is an output iff the above two conditions and then gap constraint: $(SA[j]-a) \in [\alpha, \beta]$ is satisfied.

Analogous for $T[(SA[j]+1)\ldots(b-1)]$

Our Solution





Putting Things Together

1. Based on the above geometric interpretation, problem can be reduced to OSI problem.
2. Let n_h be the number of leaves whose path intersect with H_h. Then, G_h takes $O(n_h)$ space and query on G_h takes $O(\log n_h + output)$ time.
3. Therefore, overall space $= O(\sum_i n_i) = O(n \log n)$.
4. Total query time (including the pattern search) is $O(|P| + \log n + output)$, which is optimal when $|P| > \log n$.

Putting Things Together

To handle the case where $|P| < \log n$, we maintain a separate 1-D structure for every ST node of depth $< \log n$.

1. Each node stores all the distances between its consecutive occurrences.
2. They are n per string length, so $O(n \log n)$ total space.
3. To query, we find all the distances in $[\alpha, \beta]$.
4. Optimal-time linear-space structure for this problem exists.

Thus the time is optimal, $O(|P| + \mathit{output})$.

Conclusions

- A clean problem related to more complex ones that have been studied.
- $O(n \log n)$ space, optimal time.
- Can be generalized to find reverse complemented P (cloverleafs).
- Can be used to mine patterns P, but gives the same $O(n \log n + k)$ time of Brodal et al.
- Can the space be reduced to $o(n \log n)$?
  - The geometric structure can use $O(\log \log n + k)$ time
  - Thus the solution for short strings can use $O(n \log \log n)$ space
  - Bottleneck is the heavy path decomposition

THANKS