Dynamic Pattern Matching with Wildcards

Arshia Ataee Naeini, Amir-Parsa Mobed, Masoud Seddighin, Saeed Seddighin:

Pattern Matching

- Given a text T and a pattern P
- $|T| = n$
- $|P| = m$
- Does P appear in T as a consecutive substring?

Pattern Matching

- Given a text T and a pattern P
- $|T| = n$
- $|P| = m$
- Does P appear in T as a consecutive substring?

$T \longrightarrow$ abasaeedaba

$P \longrightarrow$ saeed 

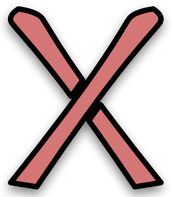
Pattern Matching

- Given a text T and a pattern P
- $|T| = n$
- $|P| = m$
- Does P appear in T as a consecutive substring?

$T \longrightarrow$ abasaeedaba

$P \longrightarrow$ saeed 

$T \longrightarrow$ abasaeedaba

$P \longrightarrow$ bsaeed 

KMP

- Can be solved in time $O(n+m)$ via KMP.

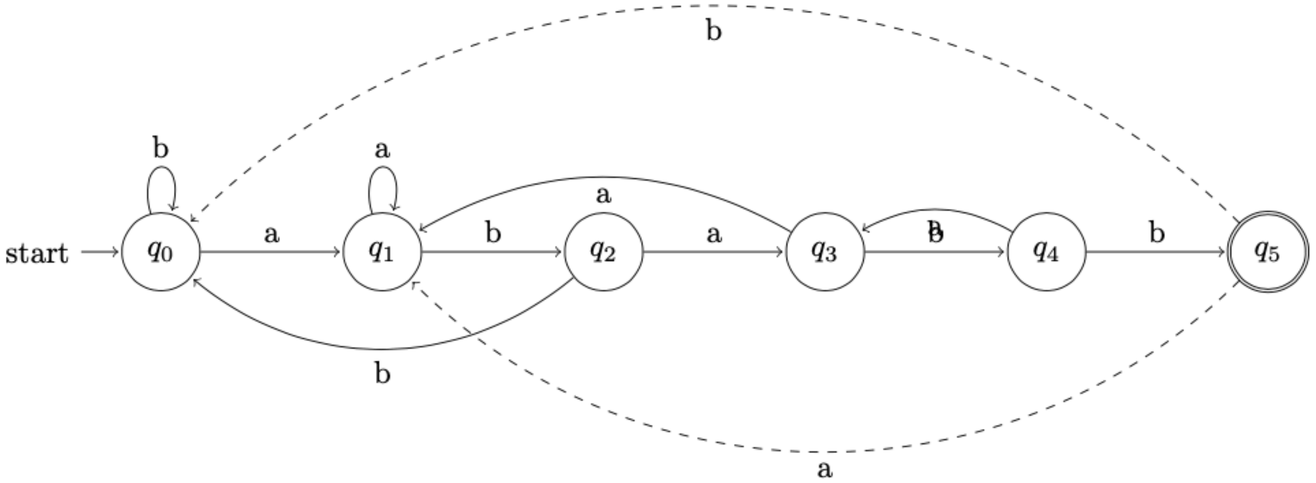
- First make an automata based on the pattern.
- Then walk over the automata as you read T.

KMP

- Can be solved in time $O(n+m)$ via KMP.

- First make an automata based on the pattern.
- Then walk over the automata as you read T.

$P \longrightarrow$ ababb



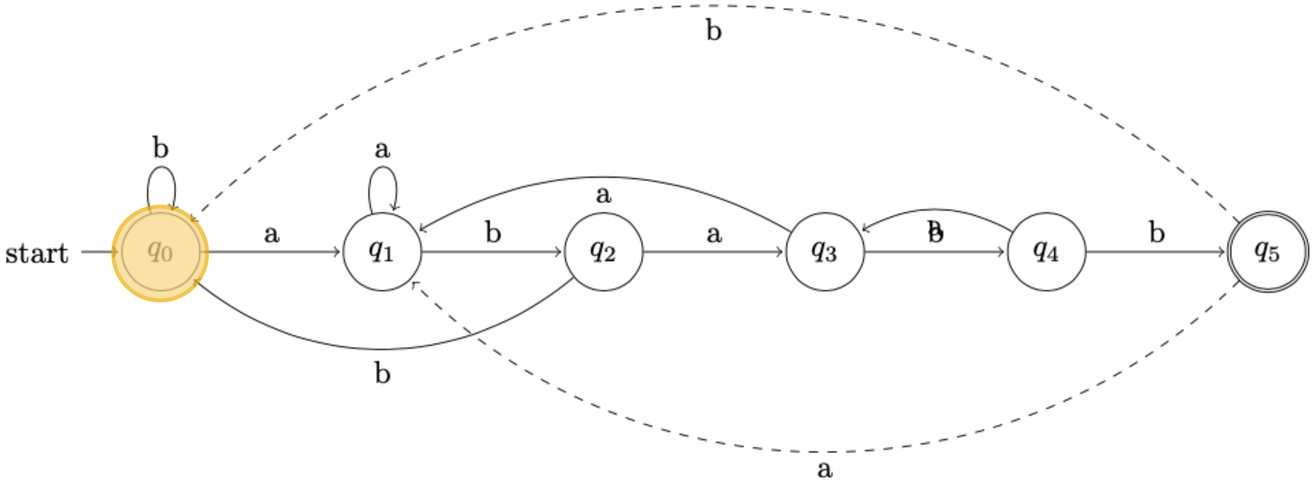
KMP

- Can be solved in time $O(n+m)$ via KMP.

T ⟶

P ⟶ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



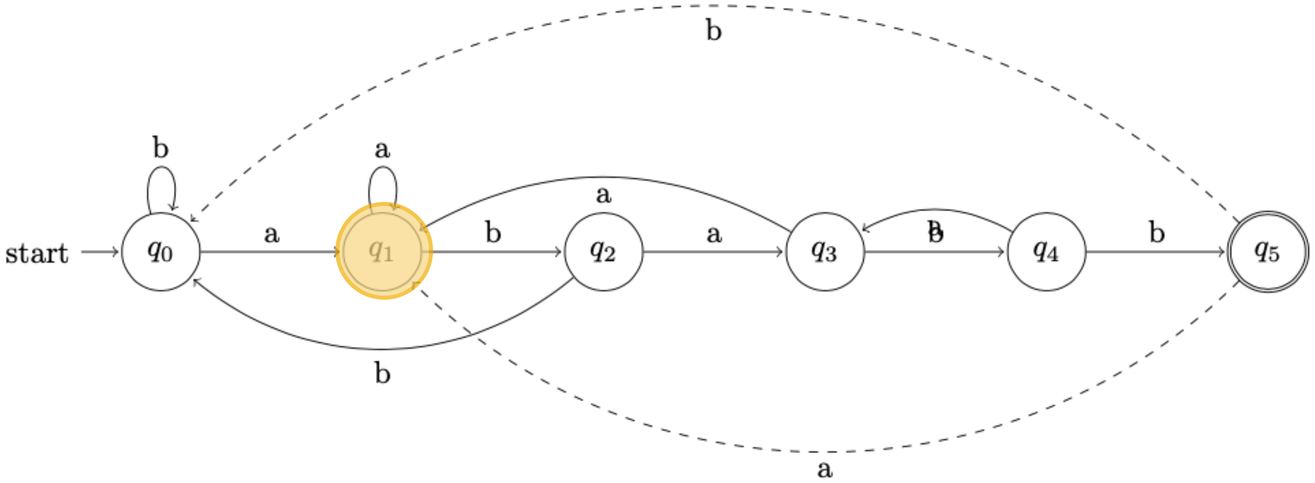
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ a

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



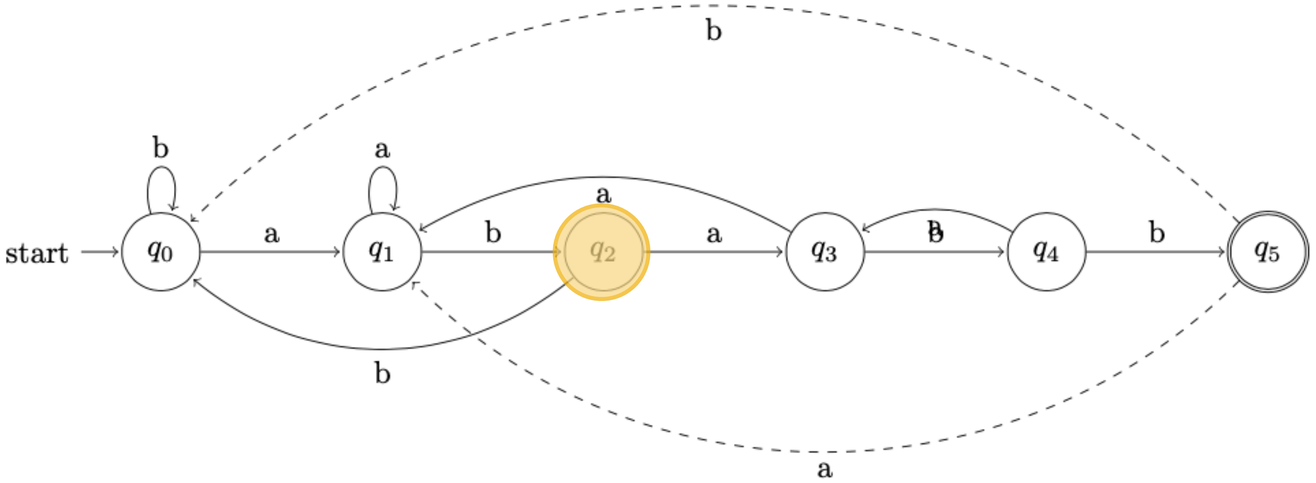
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ ab

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



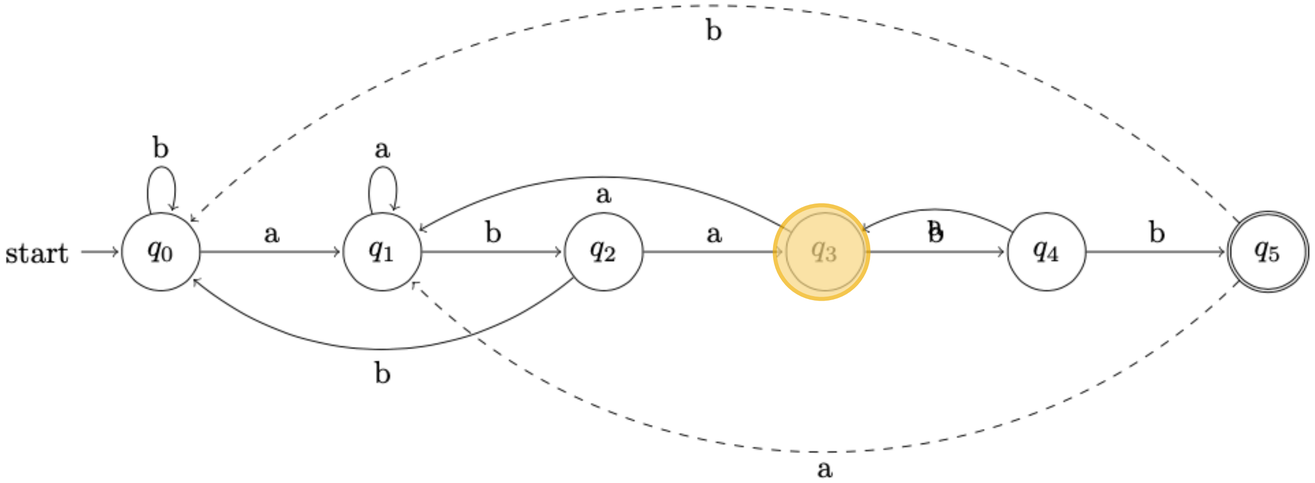
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ aba

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



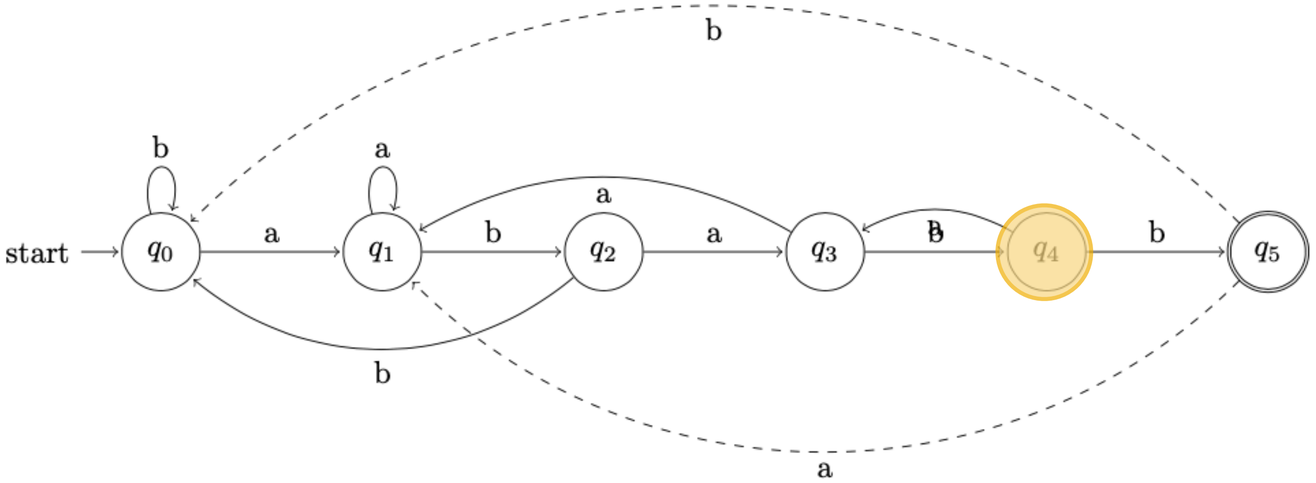
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ abab

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



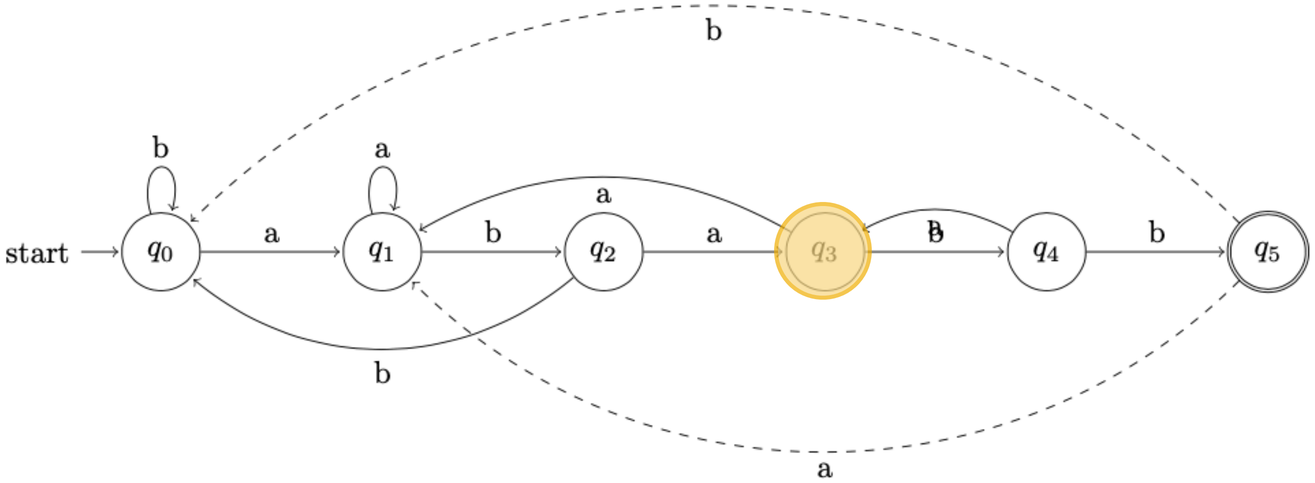
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ ababa

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



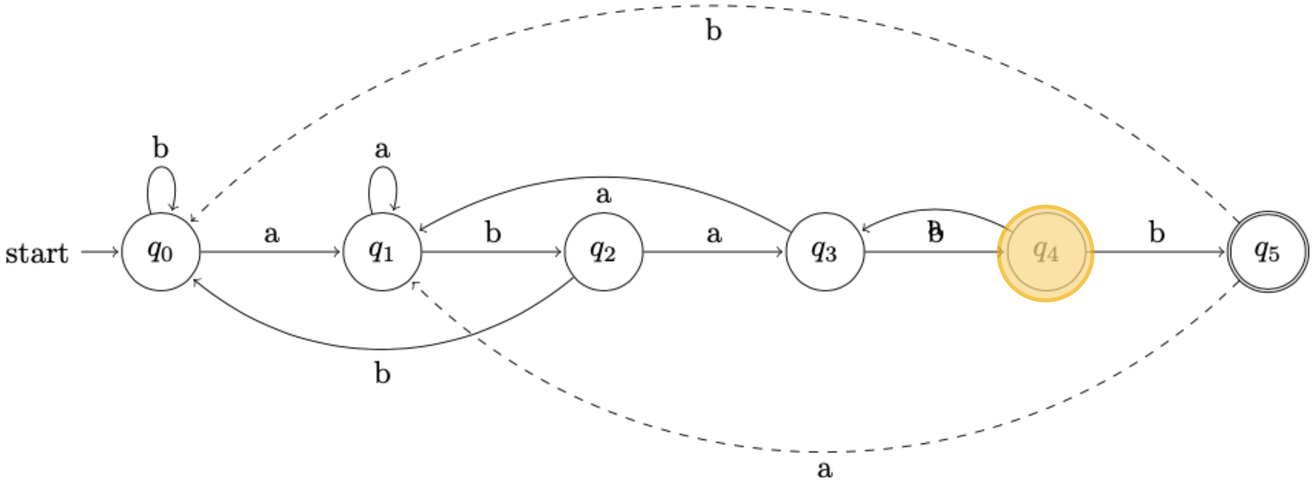
KMP

- Can be solved in time $O(n+m)$ via KMP.

$T \longrightarrow$ ababab

$P \longrightarrow$ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



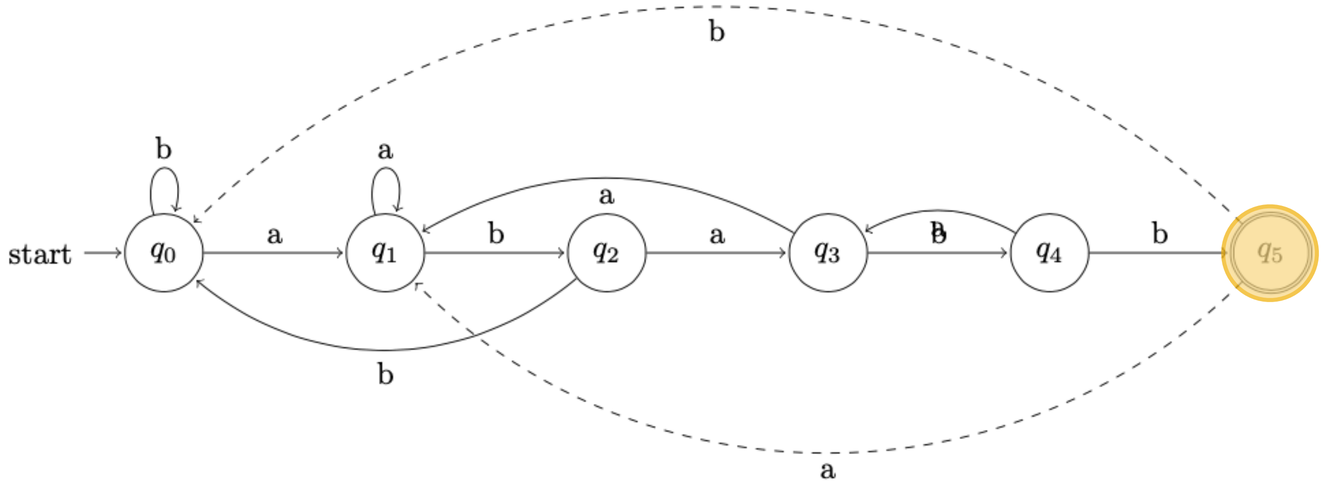
KMP

- Can be solved in time $O(n+m)$ via KMP.

T ⟶ abababb

P ⟶ ababb

- First make an automata based on the pattern.
- Then walk over the automata as you read T.



Pattern Matching with Wildcards

- Special character ? in both text and pattern
- ? can be turned into any character.

Pattern Matching with Wildcards

- Special character ? in both text and pattern
- ? can be turned into any character.

T ⟶ abasa?edaba

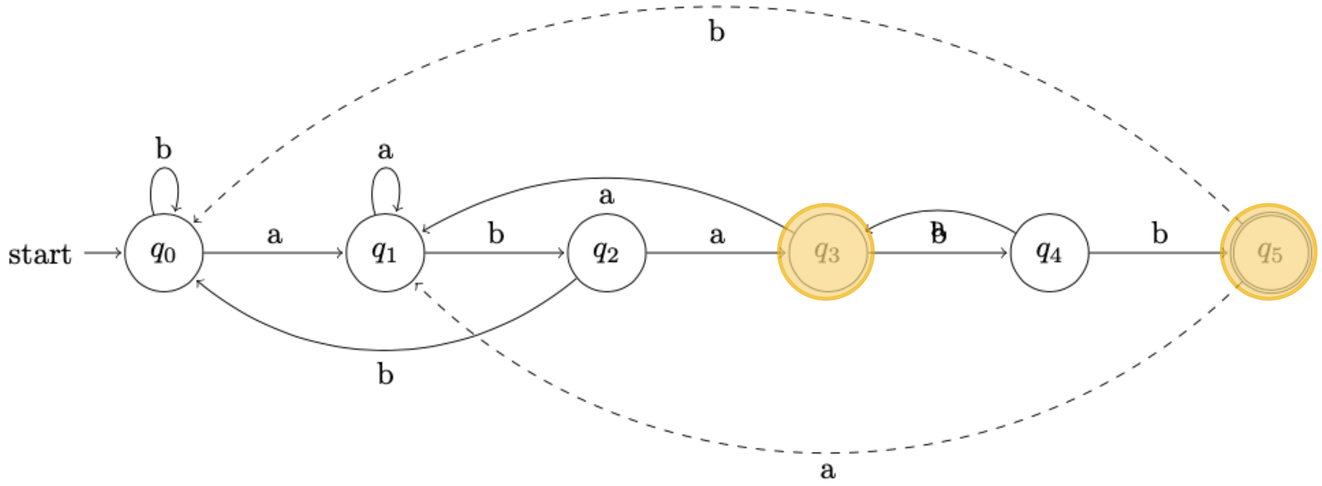
P ⟶ saee?

KMP Fails for Wildcards

- We might be at multiple nodes.
- Instead of linear time search we need quadratic time search.

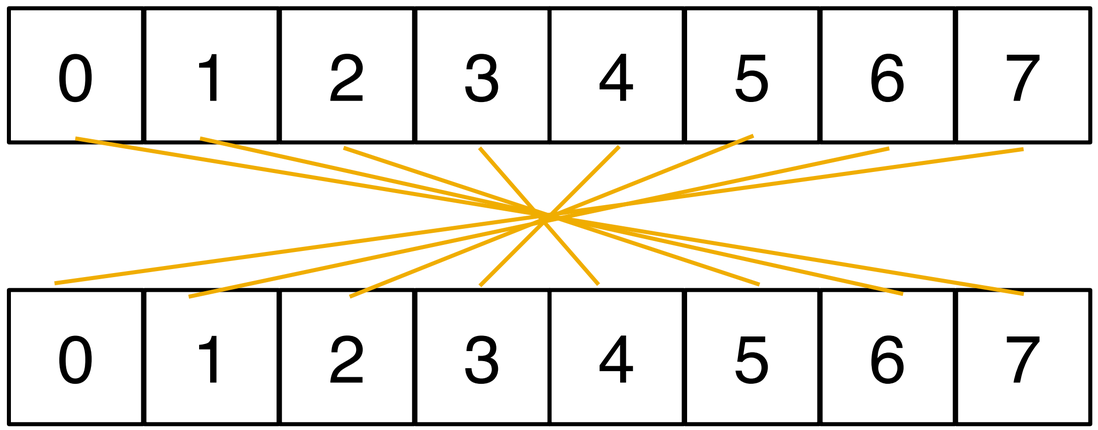
T ⟶ abab?

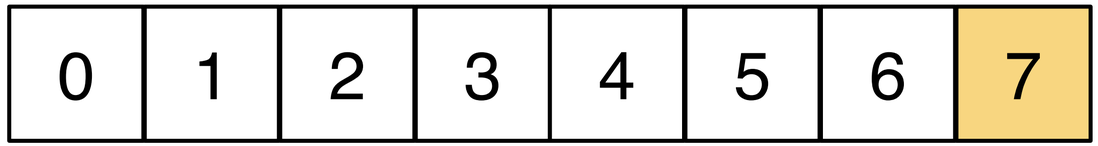
P ⟶ ababb



Wildcard case

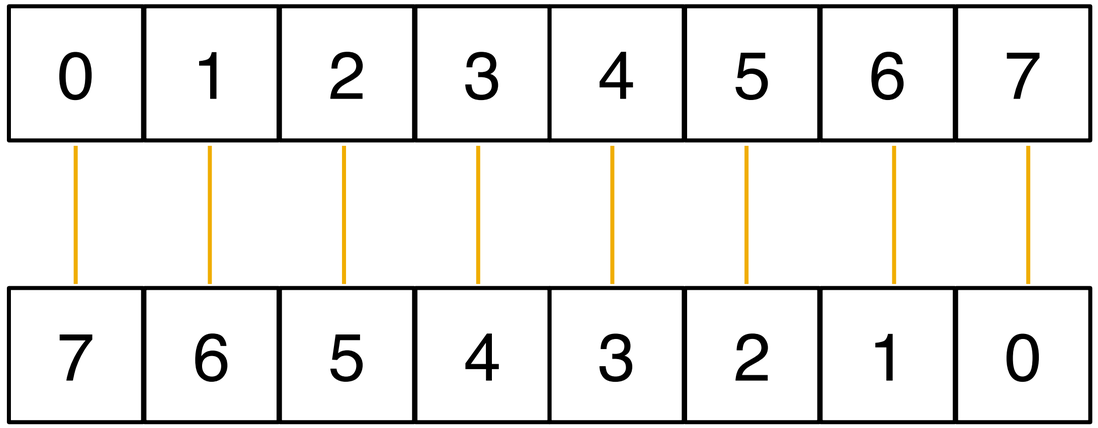
- Can be solved in time $\tilde{O}(n+m)$.
- Completely different approach: polynomial multiplication

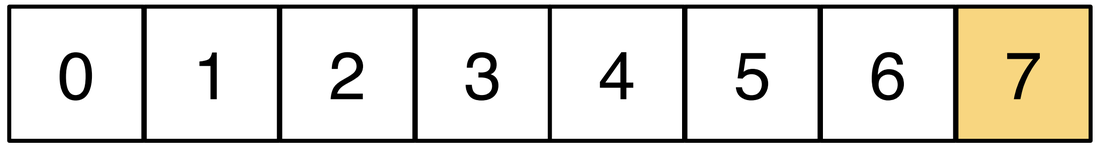




Wildcard case

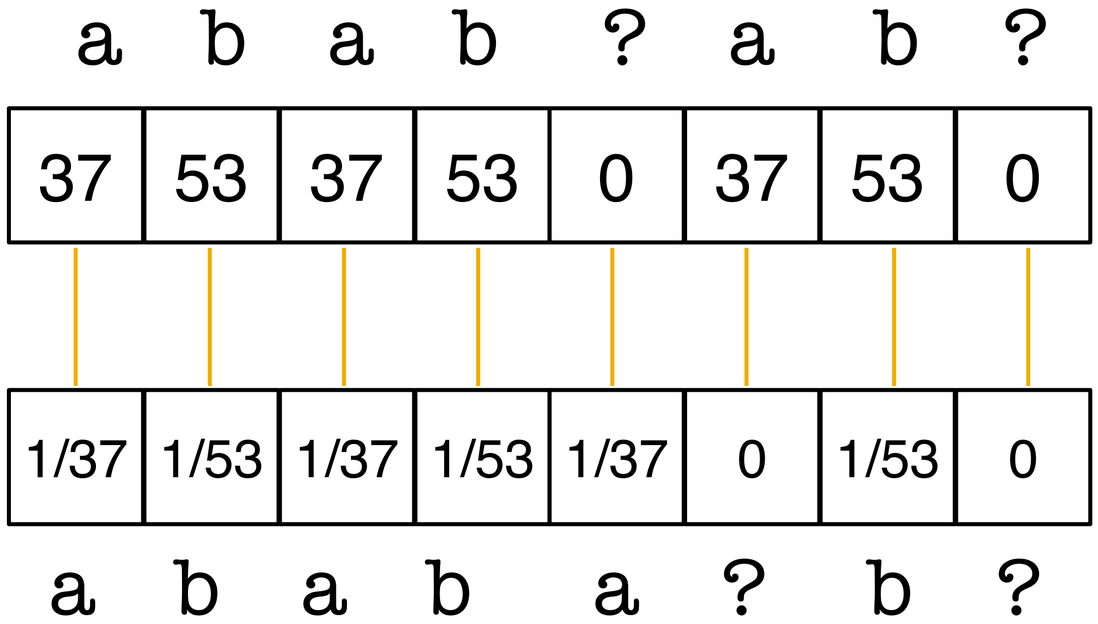
- Can be solved in time $\tilde{O}(n+m)$.
- Completely different approach: polynomial multiplication

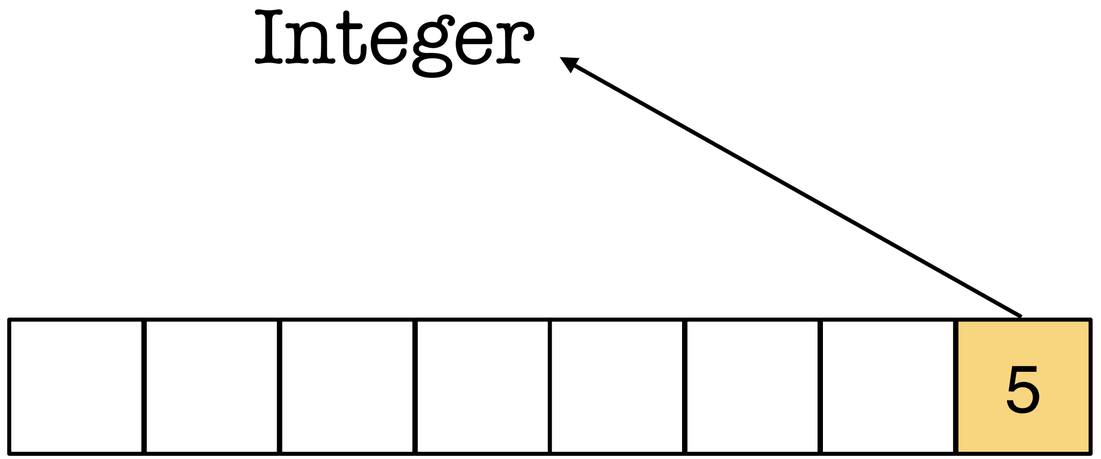




Wildcard case

a : 37
B : 53





Dynamic Setting

- Start with two empty strings.
- Each time a character is added or removed from/to P or T.
- Upon each change we should report whether P appears in T.

Previous Work

Problem Variant	Preprocessing Time	Algorithm/Query Time
Static Matching		
Exact Matching [Knuth et al., 1977, Boyer and Moore, 1977]	$\mathcal{O}(m)$	$\mathcal{O}(n)$
Wildcard Matching [Fischer and Paterson, 1974, Indyk, 1998, Kalai, 2002]	$\mathcal{O}(m \log m)$	$\mathcal{O}(n \log m)$
k-mismatches(constant $-\Sigma-$) [Abrahamson, 1987, Clifford and Clifford, 2007]	$\mathcal{O}(m \log m)$	$\mathcal{O}(n \log m)$
Dynamic Exact Matching		
Dynamic Text [Amir et al., 2007a]	$\mathcal{O}(n \log \log m)$	$\mathcal{O}(\log \log m)$ per update
Fully Dynamic [Alstrup et al., 2000]	$\mathcal{O}(n \log^2 n)$	$\mathcal{O}(\text{polylog } n)$ per query

Our Results

Problem Variant	Preprocessing Time	Algorithm/Query Time
Dynamic Matching with Wildcards (This Work)		
General (Randomized) **[Theorem 5.1]**	$\mathcal{O}(n \log^2 n)$	$\tilde{\mathcal{O}}(kn^{\frac{k}{k+1}} + k^2 \log n)$
Sparse Pattern (at most 2 non-wildcards) **[Theorem 6.3]**	$\mathcal{O}(n^{9/5})$	$\mathcal{O}(n^{4/5} \log n)$
Sparse Pattern (fixed wildcards) **[Theorem 6.9]**	$\mathcal{O}(wn)$	$\mathcal{O}(w + \log n)$
Conditional Lower Bound **[Theorem 7.4]**	–	No $\mathcal{O}(n^{1-\varepsilon})$ for $k = \Omega(\log n)$

Lower Bound

- Each update takes time $\tilde{\Omega}(n)$ even if the text remains intact.

- Reduction from Orthogonal Vector

Orthogonal Vectors

- n 0/1 vectors of size $O(\log n)$ called A
- n 0/1 vectors of size $O(\log n)$ called B

- Are there vectors $a \in A$ and $b \in B$ such that

$$a \cdot b = 0$$

- Orthogonal Vectors cannot be solved in time $\tilde{O}(n^{1.99})$ unless Strong Exponential Time Hypothesis fails.

Reduction from OV to Dynamic Pattern Matching with Wildcards

- Put the n vectors of A next to each other separated by #.
- Start with #0000…00# for the pattern.
- Iterate over B and for each vector $b \in B$:
  - Change the pattern to $\#\hat{b}_1, \hat{b}_2, \hat{b}_3, \ldots, \hat{b}_{\log n}\#$ where:

$$b_i = 1 \rightarrow \hat{b}_i = 0 \qquad \text{and} \qquad b_i = 0 \rightarrow \hat{b}_i = ?$$

Reduction from OV to Dynamic Pattern Matching with Wildcards

- If there is a match then b is orthogonal to some vector in A.
- Update time $O(n^{0.99})$ implies solving OV in time $O(n^{1.99})$.

Reduction from OV to Dynamic Pattern Matching with Wildcards

- If there is a match then b is orthogonal to some vector in A.
- Update time $O(n^{0.99})$ implies solving OV in time $O(n^{1.99})$.

- Theorem [this work] Dynamic String Matching with Wildcards cannot be solved with update time $O(n^{0.99})$ unless Strong Exponential Hypothesis is refuted.

Reduction from OV to Dynamic Pattern Matching with Wildcards

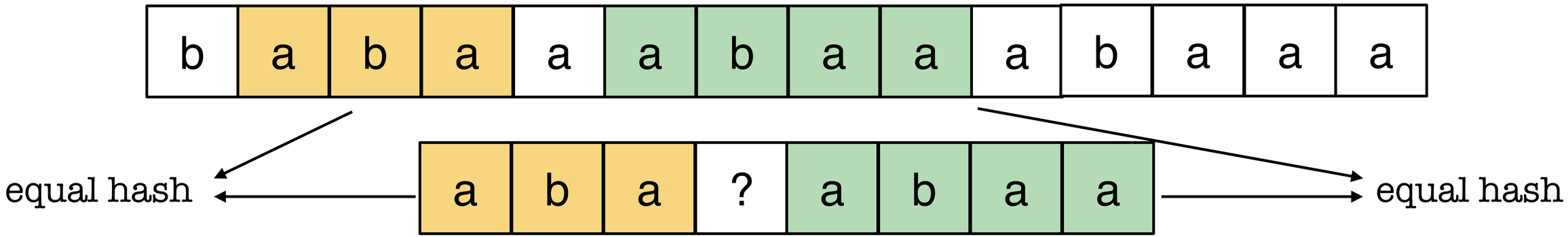
- Text remains constant throughout the lifetime of the algorithm.
- At most $O(\log n)$ wildcards at each point in time

- Theorem [this work] Dynamic String Matching with Wildcards cannot be solved with update time $O(n^{0.99})$ unless Strong Exponential Hypothesis is refuted.

Case with Only One Wildcard

- Use dynamic hashing so that the hash values of the substrings of the pattern can be found in time $\tilde{O}(1)$.

- Whether or not a given substring of the text matches the pattern can determined in time $\tilde{O}(1)$.



Case with Only One Wildcard

- Call a character rare if it appears at most $O(\sqrt{n})$ times and frequent otherwise.
- If any rare character contributes to the solution, the solution can be found in time $O(\sqrt{n})$.
- Difficult case:
  - All non wildcard characters of the pattern are frequent!
  - Two cases
    - 1- the only wildcard matches a frequent character.
    - 2- the only wildcard matches a rare character.

1- Wildcard Matches a Frequent Character

- Each frequent characters appears in at least $O(\sqrt{n})$ positions.
- Their total count is at most $O(\sqrt{n})$.
- Can be reduced to $O(\sqrt{n})$ queries in the non-wildcard setting.
- Queries can be handled in time $\tilde{O}(\sqrt{n})$.

2- Wildcard Matches a Rare Character

- Maintain a parallel instance of the problem where rare characters are replaced with a special character #.
- Replace the only question mark with # in this instance of the problem.
- Since the only character that matches with # is wildcard, it doesn't matter which one it is.
- Each query can be solved in time $\tilde{O}(1)$.

Case with Only One Wildcard

- Theorem [this work] Dynamic String Matching with

  One Wildcards can be solved with update time $\tilde{O}(\sqrt{n})$.

Case with Only One Wildcard

- Theorem [this work] Dynamic String Matching with

  One Wildcards can be solved with update time $\tilde{O}(\sqrt{n})$.

What if we have more wildcards?

Case with Only k Wildcard

- Whether or not a given substring of the text matches the pattern can be determined in time $\tilde{O}(k)$.
- Redefine rare characters by a new threshold $n^{\frac{k}{k+1}}$.
- If there is any rare character in the pattern, previous approach still works.
- There are $n^{\frac{1}{k+1}} + 1$ options for each wildcard:
  - 1- a rare character (1 option)
  - 2- any of the frequent characters ($n^{\frac{1}{k+1}}$ options)

Case with Only k Wildcard

- Whether or not a given substring of the text matches the pattern can determined in time $\tilde{O}(k)$.
- Redefine rare characters by a new threshold $n^{\frac{k}{k+1}}$.
- If there is any rare character in the pattern, previous approach still works.
- There are $n^{\frac{1}{k+1}} + 1$ options for each wildcard:
  - 1- a rare character (1 option)
  - 2- any of the frequent characters ($n^{\frac{1}{k+1}}$ options)

Total cases to consider: $O(n^{\frac{k}{k+1}})$

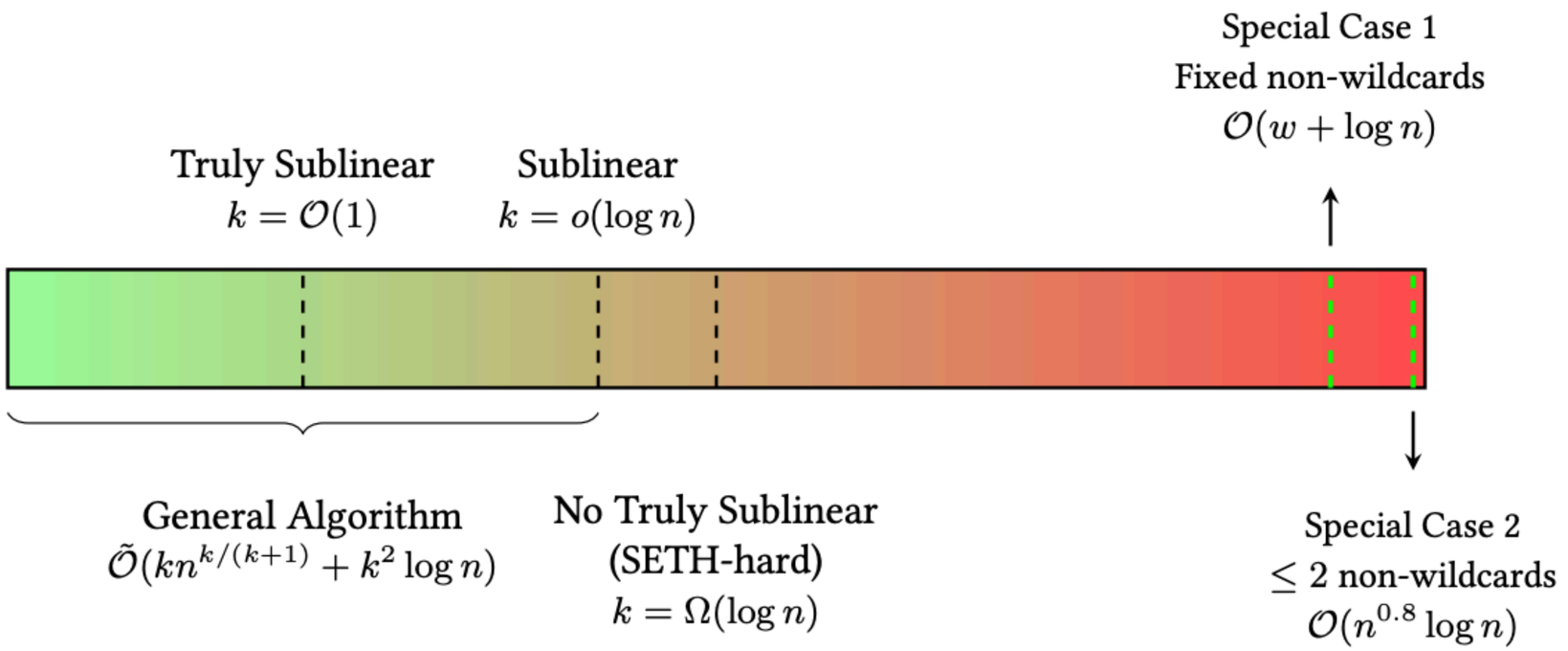
Case with Only k Wildcard

- Theorem [this work] Dynamic String Matching with

  k Wildcards can be solved with update time $\tilde{O}(kn^{\frac{k}{k+1}})$.

- For constant k the update time is truly sublinear.
- For $k = o(\log n)$ the update time is sublinear.

Thank You!



- Can we get truly sublinear for $k = o(\log n)$ (as opposed to just sublinear)