12.3 Exclusion methods
Fast expected running time

2016200648
조유나

Exclusion Method

- k-mismatch, k- difference method
  - text T length *m*, pattern P length *n*
  - Worst case $\theta(km)$

- **Goal**
  - **-faster expected running time than $\boldsymbol{\theta(km)}$**
  - **-**for a reasonable range of *k*

- Baeza-Yates and Perleberg (BYP)
- Chang and Lawler (CL)
- Myers methods

Partition approach to approximate matching

A. Partition
   - partition T (or P) into consecutive regions of a given **length r**

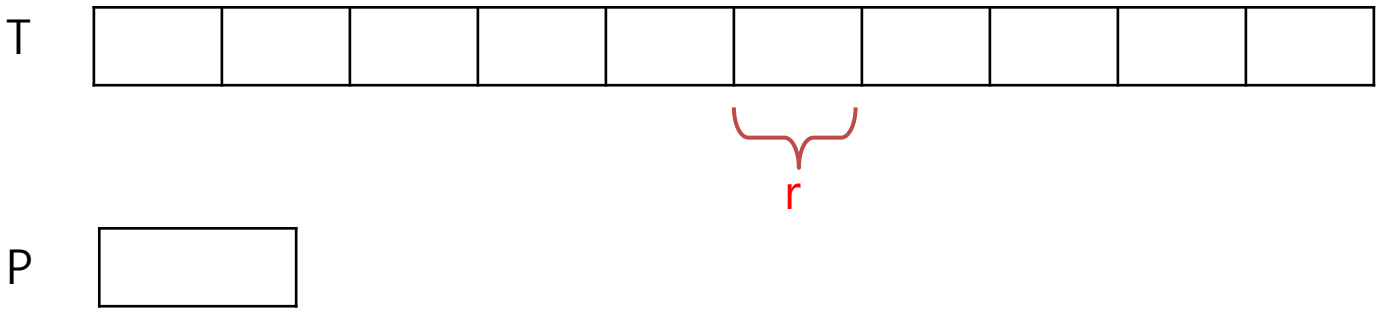
B. Search Phase
   - find length-r intervals of T that could be approximate occurrence of P using **exact matching methods**
   -> *surviving intervals*
   - Eliminate as many intervals as possible

C. Check Phase
   - for each surviving interval R of T, use **approximate matching method** to check approximate occurrence of P in a larger interval R

Partition approach to approximate matching

A. Partition

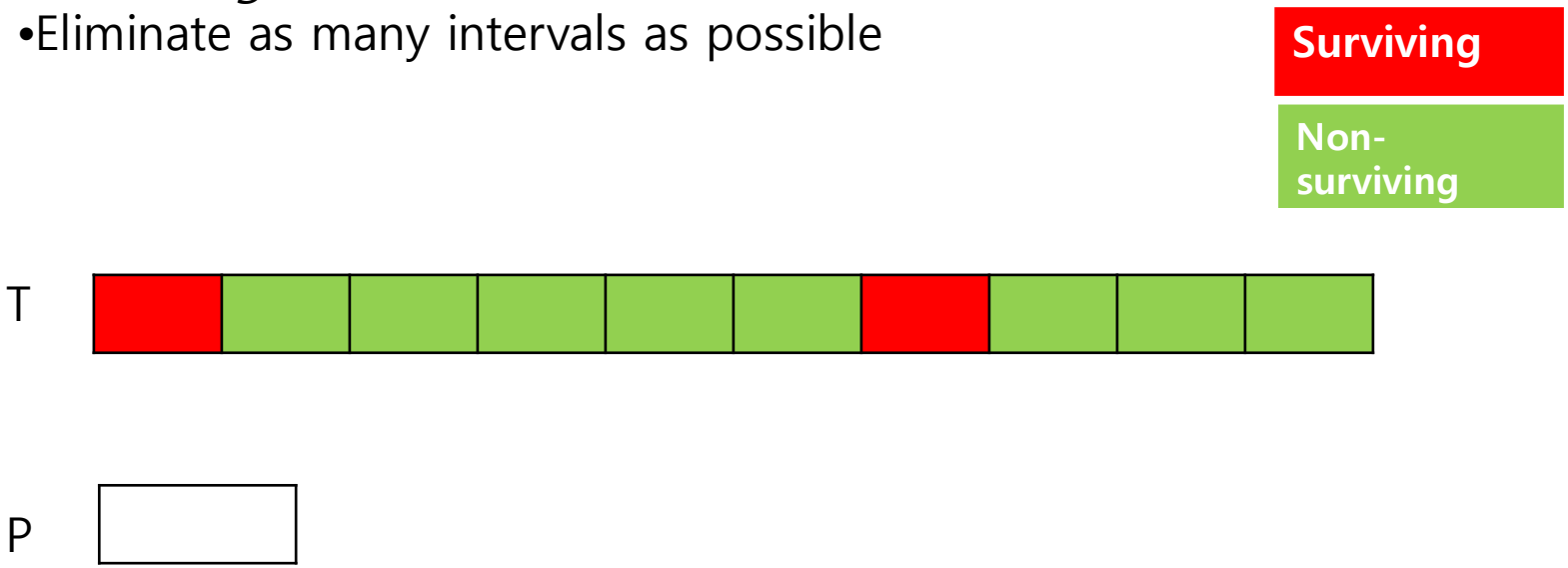
- partition T (or P) into consecutive regions of a given **length r**



Partition approach to approximate matching

B. Search Phase

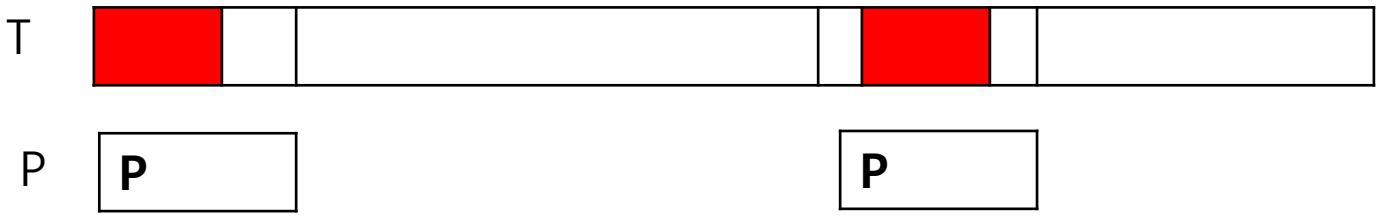
- find length-r intervals of T that could be approximate occurrence of P using **exact matching methods**
  -> *surviving intervals*
- Eliminate as many intervals as possible



Partition approach to approximate matching

C. Check Phase

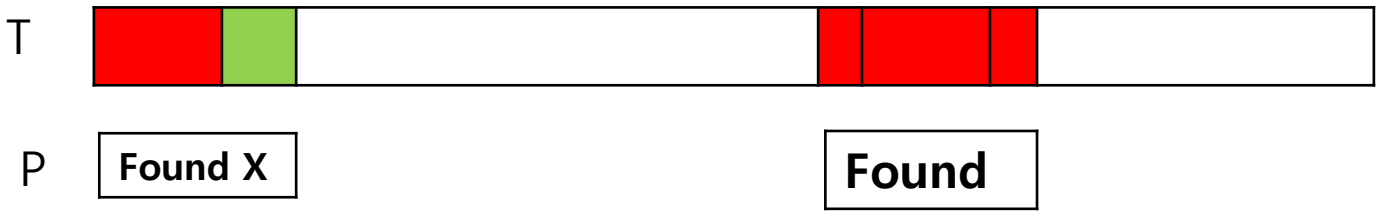
- for each surviving interval R of T, use **approximate matching** method to check approximate occurrence of P in a larger interval R



Partition approach to approximate matching

C. Check Phase

- for each surviving interval R of T, use **approximate matching** method to check approximate occurrence of P in a larger interval R



Partition approach to approximate matching

- **Method differs**
  -choice of r
  -choice of string to partition
  -exact matching method
  -definition of a region in check phase

BYP method

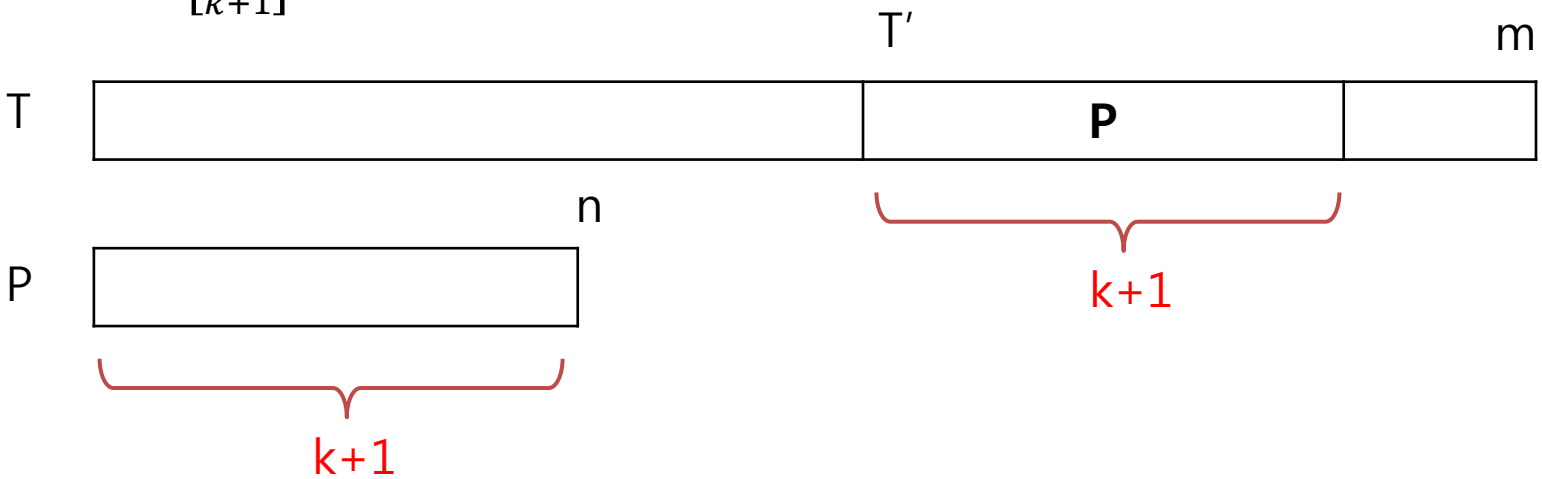
- **Baeza-Yates and Perleberg**

- **O(m) for for** $\boldsymbol{k} < (\frac{\boldsymbol{n}}{\boldsymbol{log\, n}})$

- Partition $\boldsymbol{r} = \left\lfloor \frac{\boldsymbol{n}}{\boldsymbol{k}+\mathbf{1}} \right\rfloor$

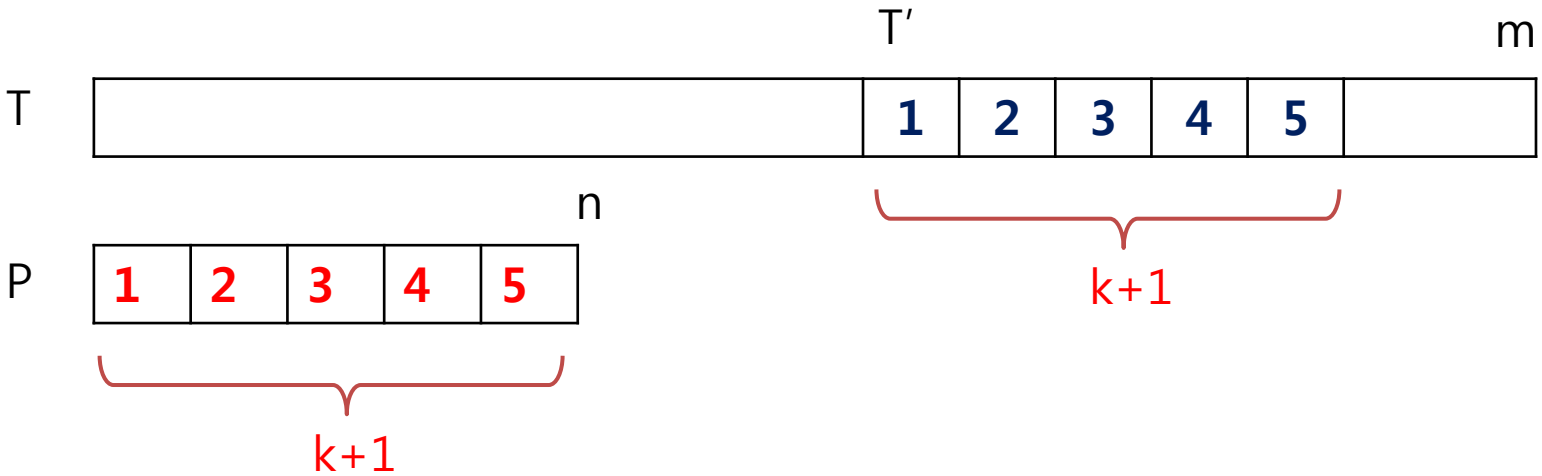
BYP method

- **Lemma 12.3.1**
  - Suppose P matches a substring T′ of T with at most k differences. Then T′ must contain at least one interval of length r that exactly matches one of the r-length regions of the partition of P.
  - $r = \left\lfloor \frac{n}{k+1} \right\rfloor$



BYP method

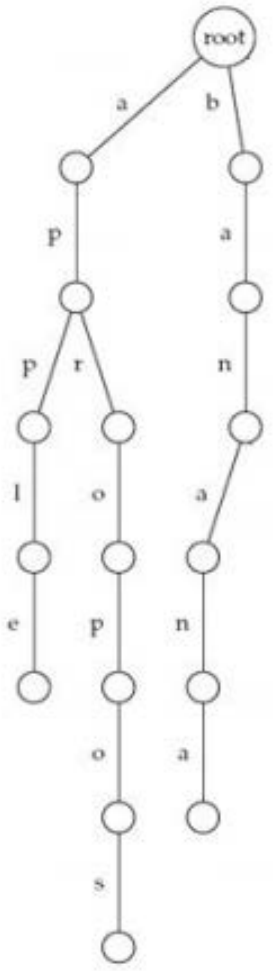
- **Lemma 12.3.1**
  - Suppose P matches a substring T′ of T with at most k differences. Then T′ must contain at least one interval of length r that exactly matches one of the r-length regions of the partition of P.
  - $k = 4$



BYP method

- Keyword tree
  - edge : element

- Aho-Corasick algorithm
  - String searching algorithm
  - time complexity O(m)

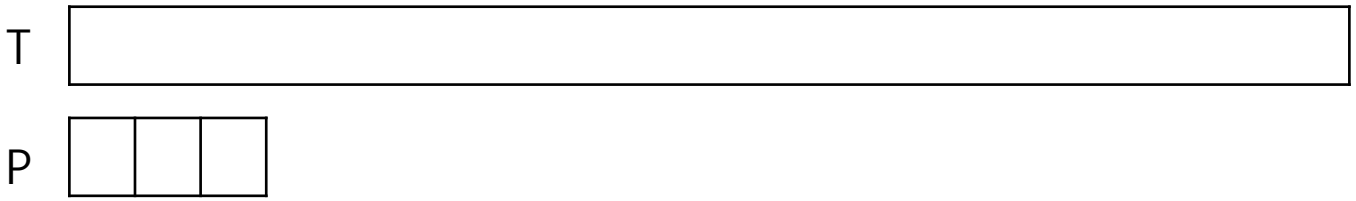


BYP method

- Partition

  -Let ***P*** be the set of k+1 substrings of P taken from the first k+1 regions of P's partition.

  - $r = \left\lfloor \frac{n}{k+1} \right\rfloor$

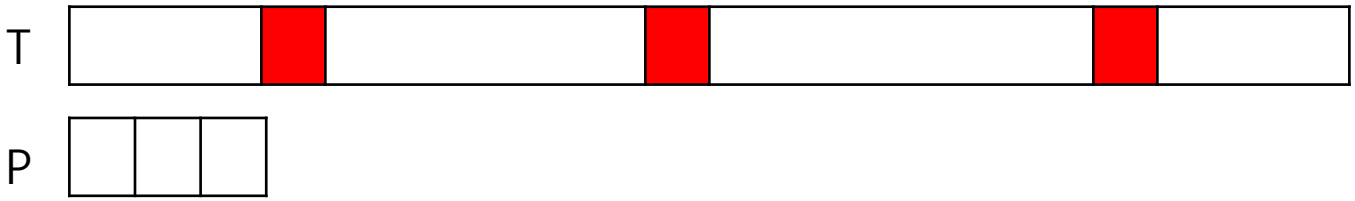


BYP method

- Search Phase

  -Build a keyword tree for the set of "patterns" ***P***

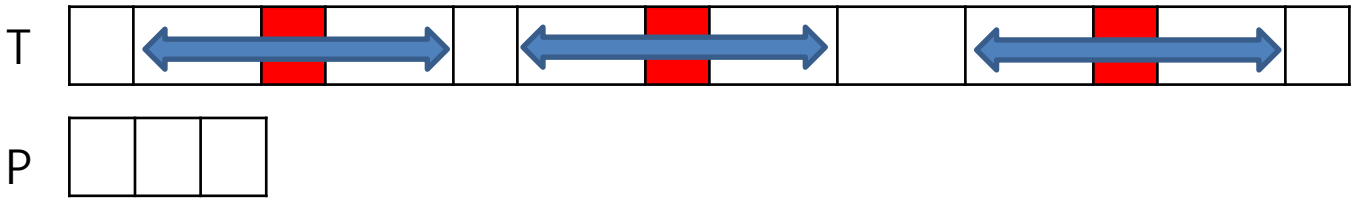
  -Using Aho-Corasik algorithm, find *I*, the set of all starting locations in T where any pattern in ***P*** occurs exactly



BYP method

- Check Phase

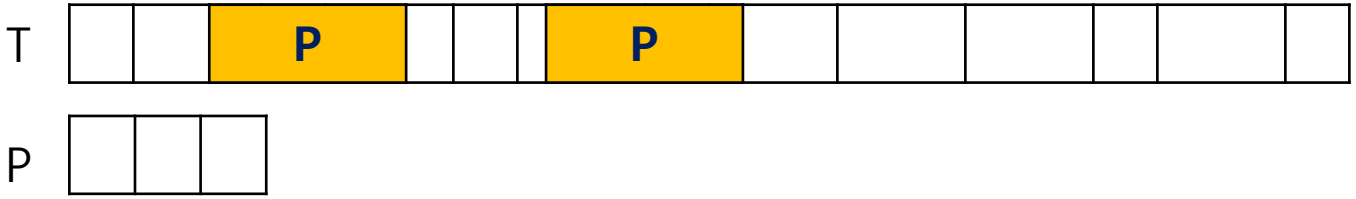
  -For each index $i \in \boldsymbol{I}$ use an approximate matching algorithm to locate the end points of all approximate occurrences of P in the substring T[i-n-k…i+n+k]



BYP method

- Check Phase

  -For each index $i \in \boldsymbol{I}$ use an approximate matching algorithm to locate the end points of all approximate occurrences of P in the substring T[i-n-k…i+n+k]



BYP method

- **Time Complexity**
  a. Let *P* be the set of k+1 substrings of P taken from the first k+1 regions of P's partition.

  b. Build a keyword tree for the set of "patterns" *P*
     - **O(n)**

  c. Using Aho-Corasik algorithm
     - **O(m)**

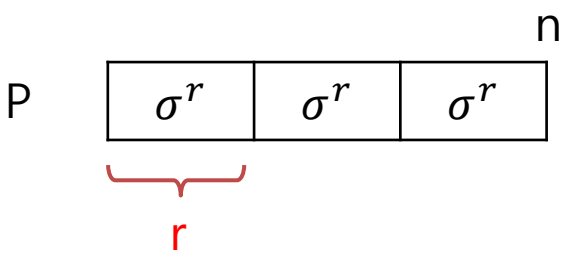
  d. Approximate matching algorithm
     - **O(m)** – no spaces allowed, know exactly where in P any match found

BYP method

- **Expected Time**
  - σ: alphabet size
  - r: partition length
  - any pattern $p \in P$

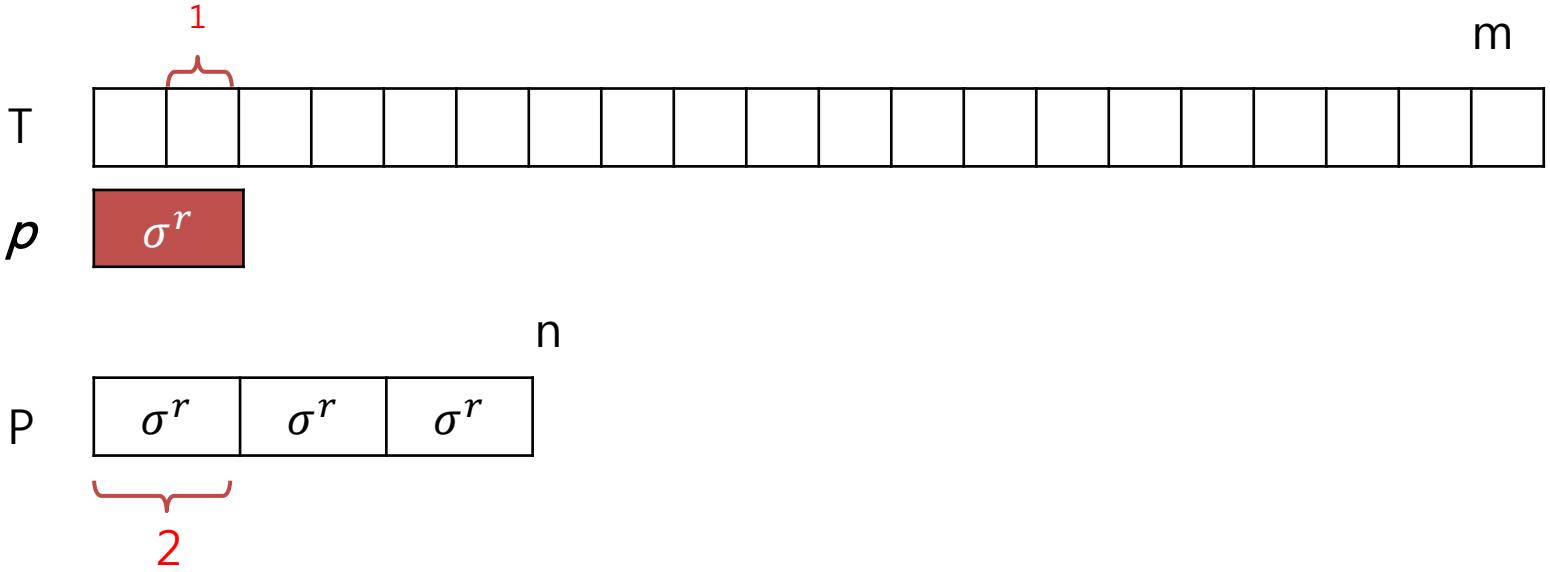
BYP method

- **Expected Time**
  - σ: alphabet size
  - r: partition length
  - any pattern $p \in P$
  - ➢ p has length r
  - ➢ Expected arbitrary string = σ^r



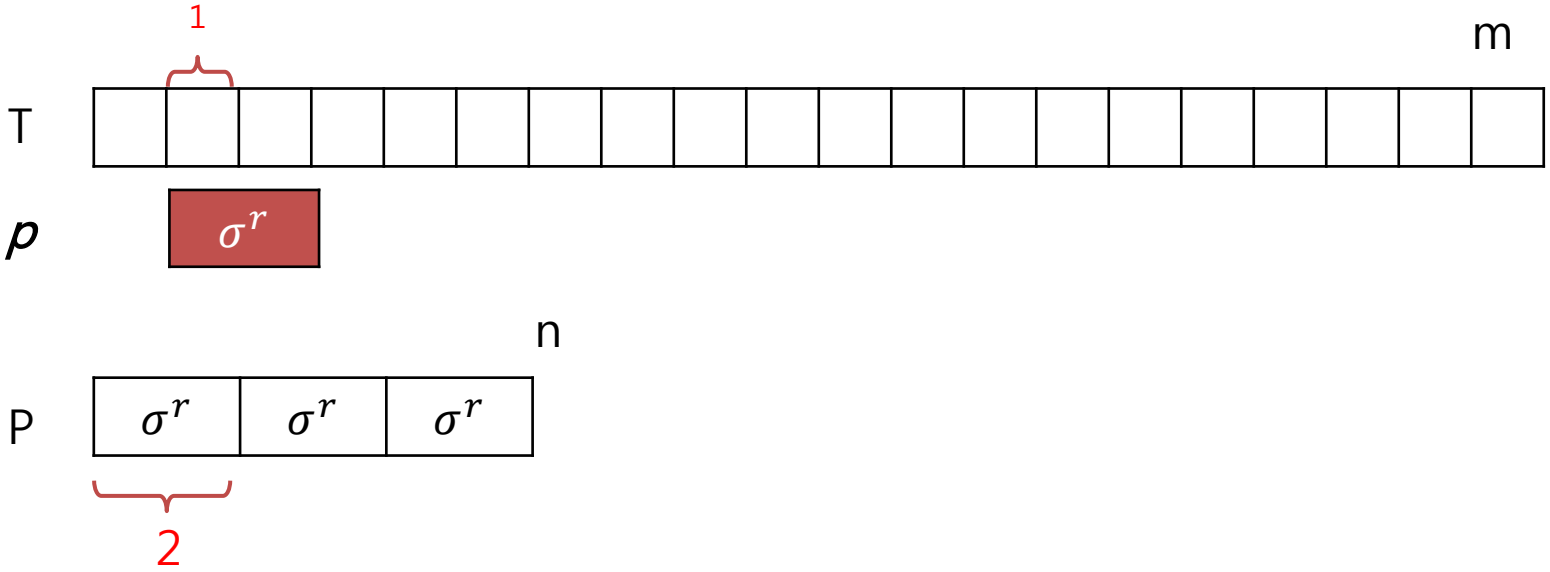
BYP method

- **Expected Time**
  - T contains roughly *m* substrings
  - Expected number of exact occurrences of ***p*** in T : $\frac{m}{\sigma^r}$
  - If k+1 = 3 and n = 6, r = 2



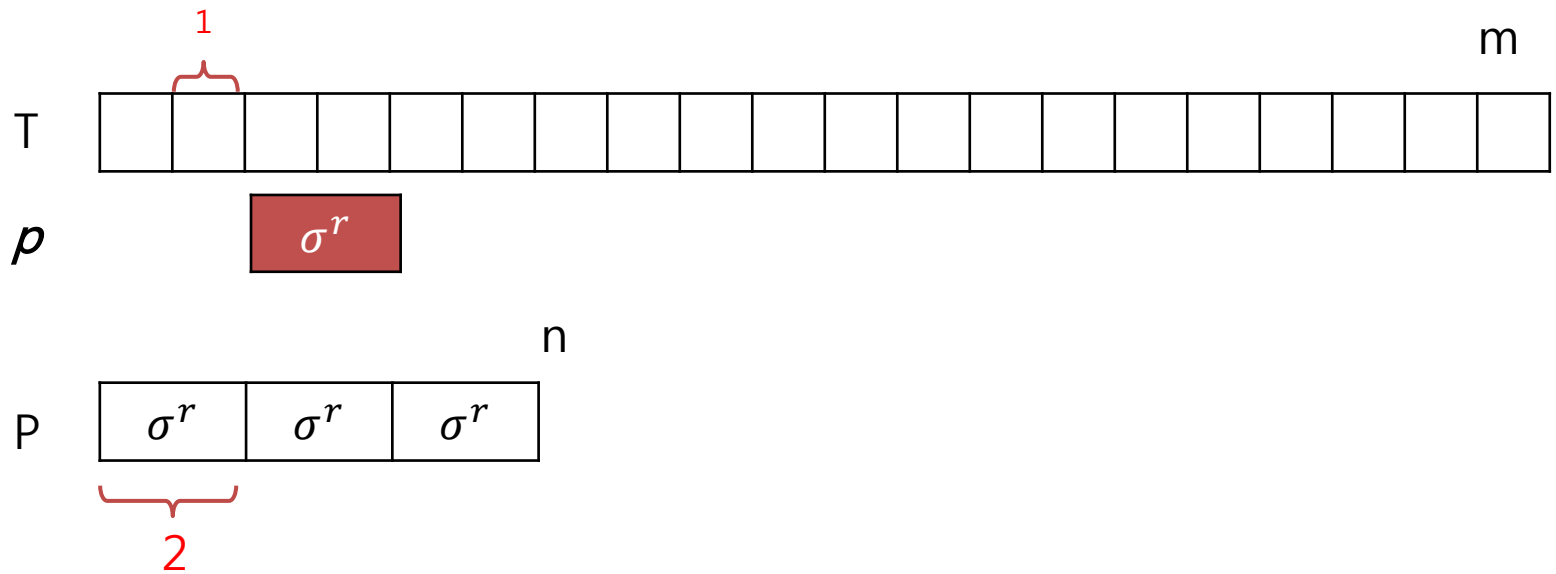
BYP method

- **Expected Time**
  - T contains roughly *m* substrings
  - Expected number of exact occurrences of ***p*** in T $= \frac{m}{\sigma^r}$
  - If k+1 = 3 and n = 6, r = 2



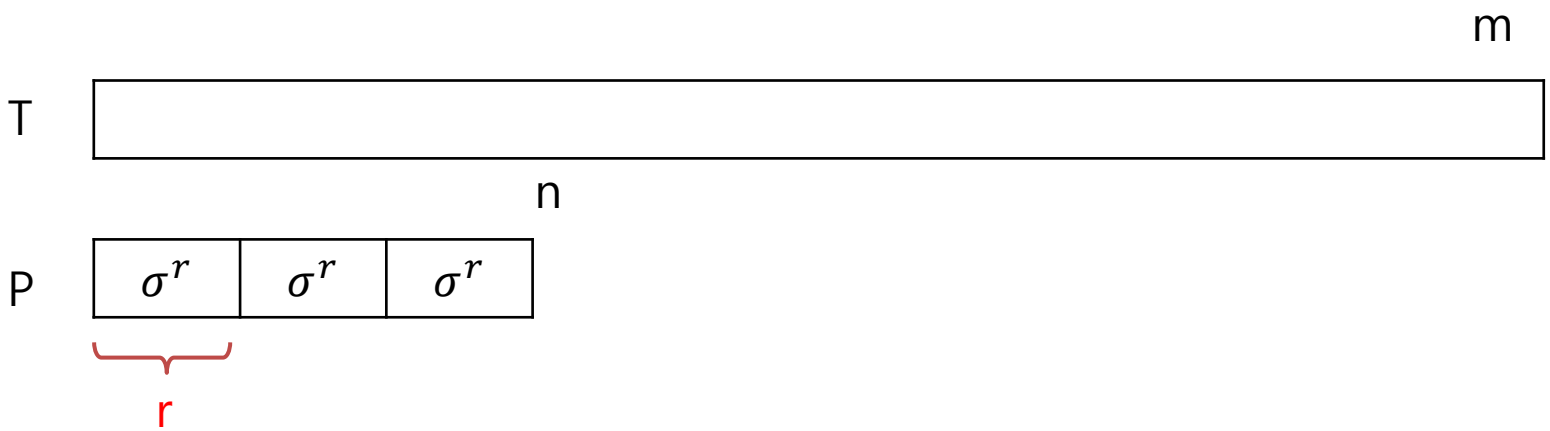
BYP method

- **Expected Time**
  - T contains roughly *m* substrings
  - Expected number of exact occurrences of ***p*** in T $= \frac{m}{\sigma^r}$
  - If k+1 = 3 and n = 6, r = 2



BYP method

- **Expected Time**
  - There are k+1 number of pattern p in set P
  - Expected total number of occurrences in T of patterns from P : $\frac{m(k+1)}{\sigma^r}$



BYP method

- **Expected Time**
  - Expected size of $\mathcal{I}$(set of survival intervals) : $\frac{\boldsymbol{m(k+1)}}{\boldsymbol{\sigma^r}}$
  - Expected Checking time [phase algorithm(O(n^2))] : $\frac{\boldsymbol{mn^2(k+1)}}{\boldsymbol{\sigma^r}}$
- Goal !
  - make the expected checking time linear in m for modest k
  - $\frac{\boldsymbol{mn^2(k+1)}}{\boldsymbol{\sigma^r}} < cm$

BYP method

- **Expected Time**
  - make the expected checking time linear in $\boldsymbol{m}$ for modest $\boldsymbol{k}$

$$\frac{\boldsymbol{mn^2(k+1)}}{\boldsymbol{\sigma^r}} < cm \quad \xrightarrow{k+1 \le n \ \vee} \quad \frac{\boldsymbol{mn^3}}{\boldsymbol{\sigma^r}} = cm$$

$$\boldsymbol{r} = \log_{\boldsymbol{\sigma}} \boldsymbol{n^3} - \log_{\boldsymbol{\sigma}} \boldsymbol{c} \quad \xrightarrow{r \le \frac{n}{k+1}} \quad k+1 \le \frac{n}{3\log_\sigma n - \log_\sigma c}$$

➢ **Algorithm BYP runs in O(m) time for $\boldsymbol{k = O(\frac{n}{log\ n})}$**

BYP method

- **Expected Time**
  - σ: alphabet size
  - r: partition length
  - any pattern $p \in P$

➢ Expected number of exact occurrences of $\boldsymbol{p}$ in T : $\frac{m}{\sigma^r}$

➢ Expected total number of occurrences in T of patterns from $\boldsymbol{P}$: $\frac{\boldsymbol{m(k+1)}}{\boldsymbol{\sigma^r}}$

➢ Expected Checking time [phase algorithm(O(n^2))] : $\frac{\boldsymbol{mn^2(k+1)}}{\boldsymbol{\sigma^r}}$

➢ Algorithm BYP runs in O(m) time for $k = O(\frac{n}{\log n})$

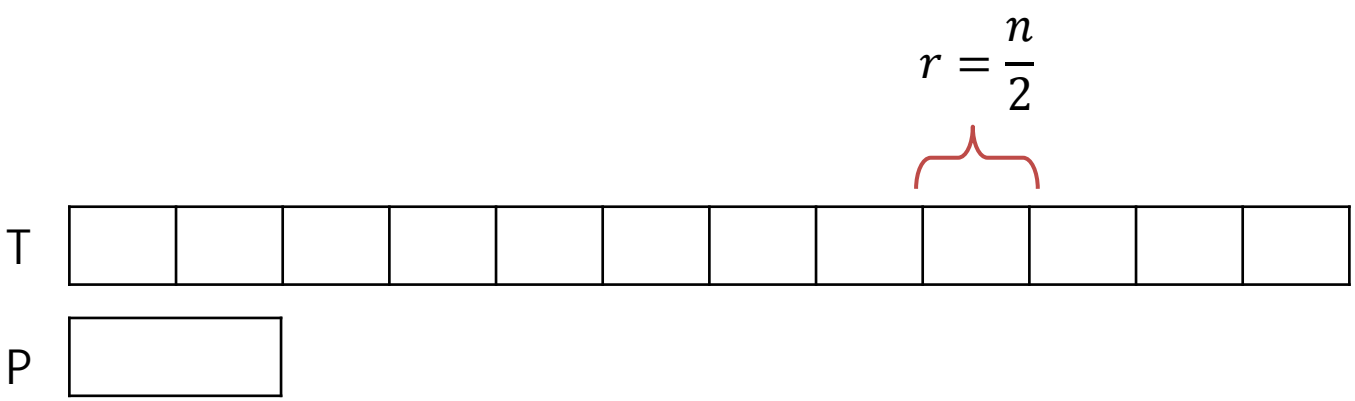
CL method

- **Chang-Lawler method**
- **O(m) for k < n/$\log_\sigma$n**
- **Partition $r = \frac{n}{2}$**

CL method

- Partition

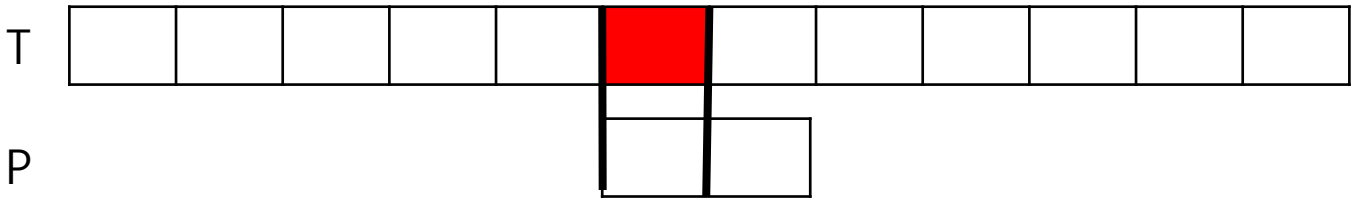
  -string T is partitioned into consecutive fixed regions of length $r = \frac{n}{2}$



CL method

- Partition

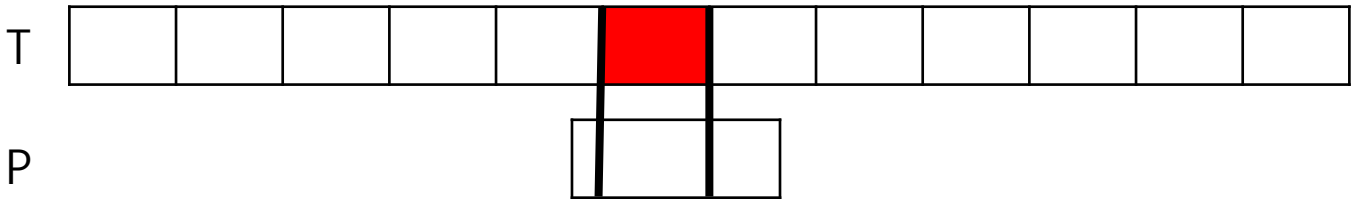
  -string T is partitioned into consecutive fixed regions of length $r = \frac{n}{2}$



CL method

- Partition

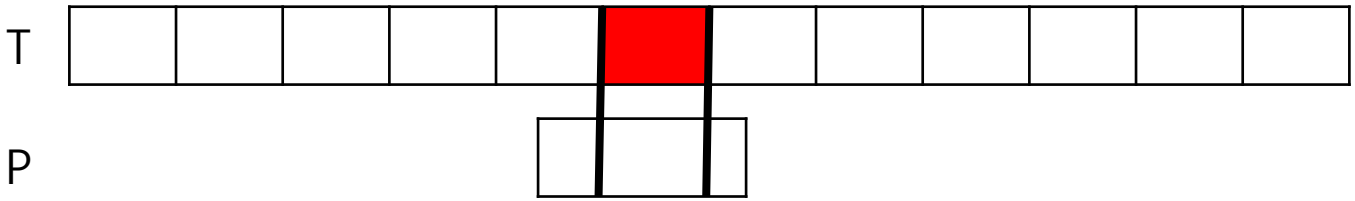
  -string T is partitioned into consecutive fixed regions of length $r = \frac{n}{2}$



CL method

- Partition

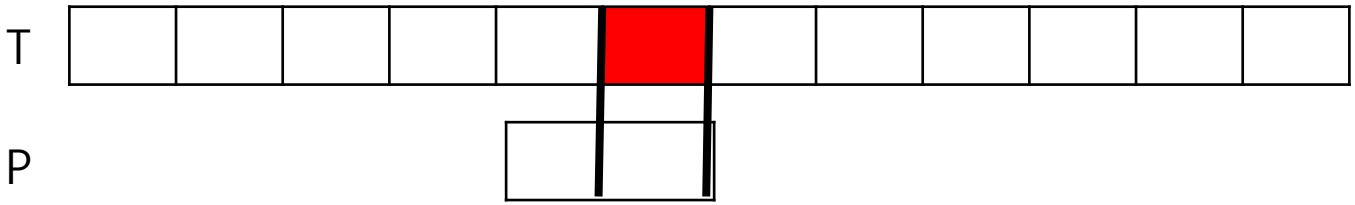
  -string T is partitioned into consecutive fixed regions of length $r = \frac{n}{2}$



CL method

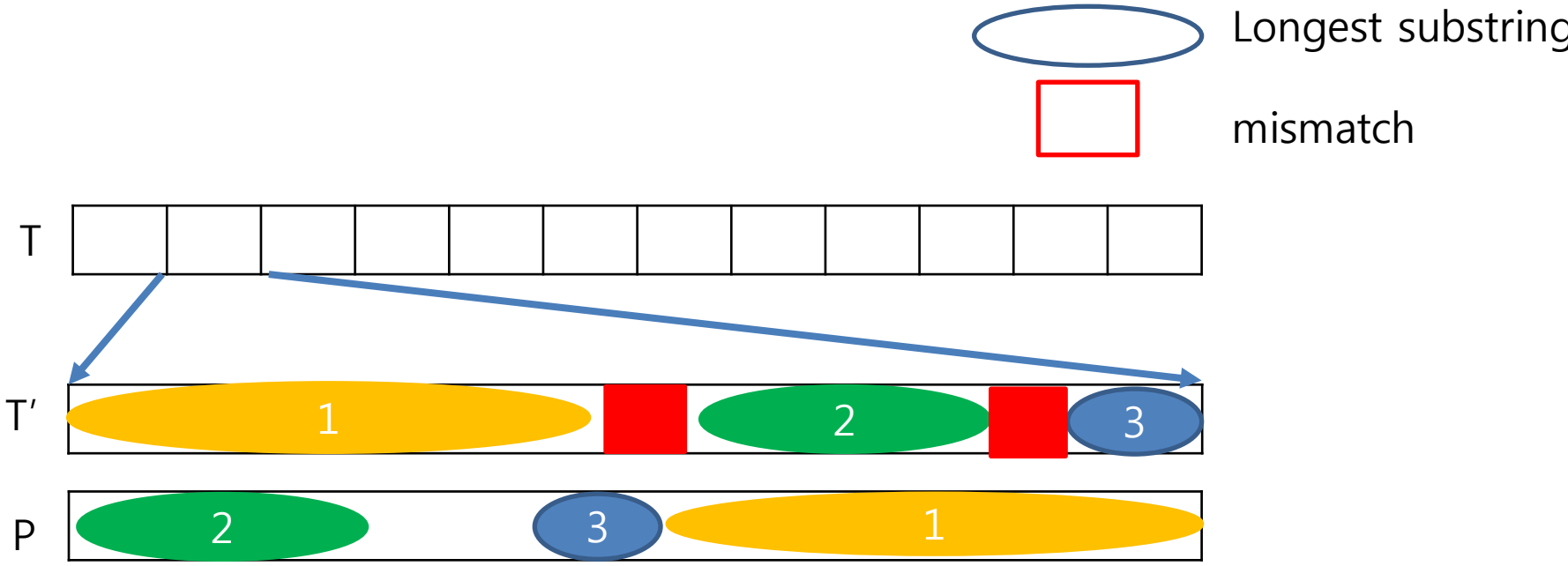
- Partition

  -string T is partitioned into consecutive fixed regions of length $r = \frac{n}{2}$



CL method

- Search Phase
  - Use *matching statistics*
  - ***ms(i)*** : length of the longest substring starting at position *i* of T that matches a substring *somewhere* in *P*

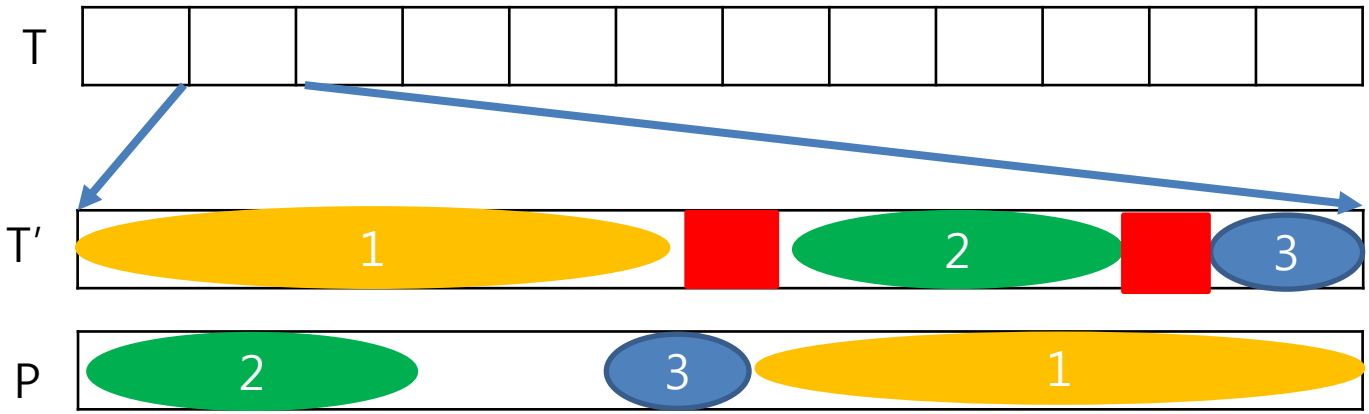


CL method

- Search Phase
  - If mismatch exceed k,

    stop searching and exclude region from set *R*(surviving region)
  - If T′ has finished being mapped,

    leave the region in set *R* and go onto the next region.

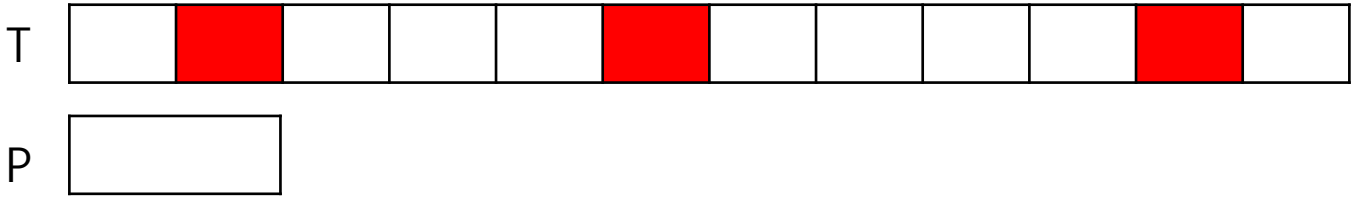


CL method

- Search Phase
  - If mismatch exceed k,

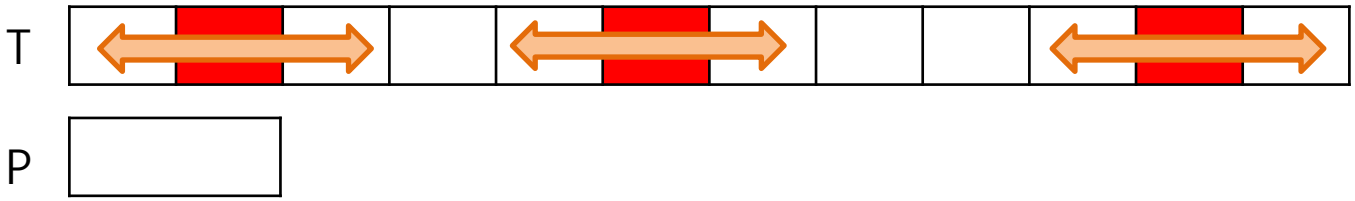
    stop searching and exclude region from set ***R***(surviving region)
  - If T′ has finished being mapped,

    leave the region in set ***R*** and go onto the next region.



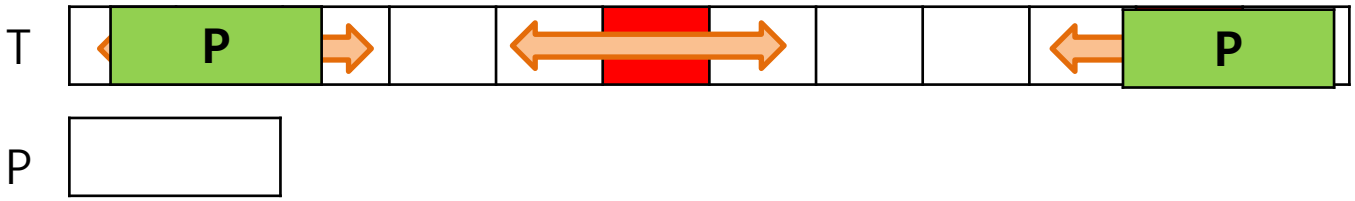
CL method

- Check Phase
  - R is a surviving region
  - Executes approximate matching algorithm for P against a neighborhood of T that starts n/2 positions to the left of R and ends n/2 positions to its right



CL method

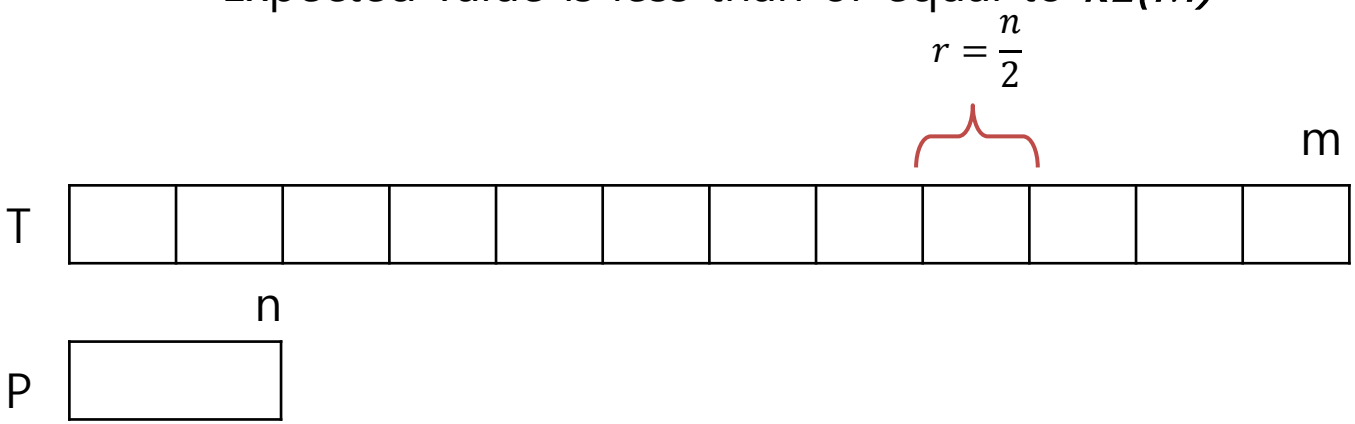
- Check Phase
  - R is a surviving region
  - Executes approximate matching algorithm for P against a neighborhood of T that starts n/2 positions to the left of R and ends n/2 positions to its right



CL method

- Time Complexity
  - Search phase
    - number of regions = m(n/2) = 2m/n
    - ms(i) is repeated until either mismatches exceed k or the whole region was searched.
    - E(M) : expected value of matching statistics.

    => Expected value is less than or equal to ***kE(M)***

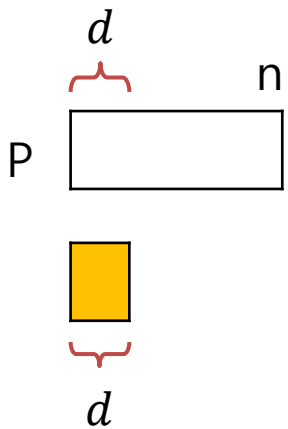


CL method – search phase

- **Lemma 12.3.3**

  E(M), the expected value of a matching statistic, is $O(\log_\sigma n)$

- **Proof>**
  - For fixed length d, roughly n substrings of length d in P

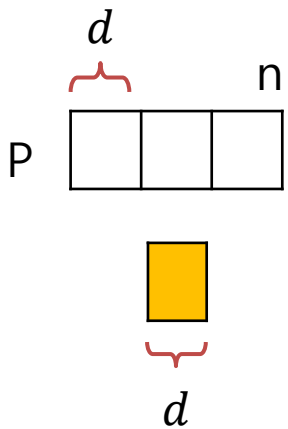


CL method – search phase

- **Lemma 12.3.3**

  E(M), the expected value of a matching statistic, is $O(\log_\sigma n)$

- **Proof>**
  - For fixed length d, roughly n substrings of length d in P

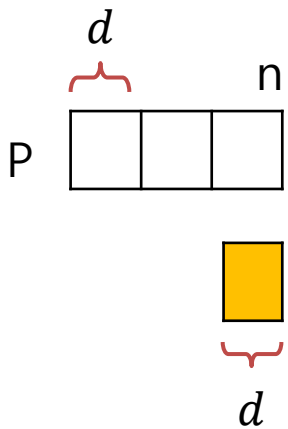


CL method – search phase

- **Lemma 12.3.3**

  E(M), the expected value of a matching statistic, is $O(\log_\sigma n)$

- **Proof>**
  - For fixed length d, roughly n substrings of length d in P



CL method – search phase

- **Lemma 12.3.3**

  E(M), the expected value of a matching statistic, is $O(\log_\sigma n)$

- **Proof>**
  - For fixed length d, roughly n substrings of length d in P
  - For any specific string α of length d,

    probability that α is found somewhere in P $< \frac{n}{\sigma^d}$
  - unless σ^d=n

CL method – search phase

- **Lemma 12.3.3**

  E(M), the expected value of a matching statistic, is $O(\log_\sigma n)$

- **Proof>**
  - Let X be the random variable that has value $\log_\sigma n$ for ms(i)≤$\log_\sigma n$
  - Fixed d until infinite
  - Then E(M) = $O(\log_\sigma n)$

$$E(M) < E(X) < \log_\sigma n + \sum_{l=\log_\sigma n}^{\infty} \frac{l}{\sigma^l} = \log_\sigma n + 2.$$

CL method

- Time Complexity
  - Search phase
    - number of regions = m(n/2) = 2m/n
    - Matching statistics: kE(M) = $O(k\log_\sigma n)$
    - Total expected search phase

      = $O(2mk E(M)/n)$

      = $O(2mk \log_\sigma n/n)$

      same process as the BYP method

      = O(m) for k < $n/\log_\sigma n$

CL method

- Time Complexity
  - Check Phase
    - Expected number of regions that survived : e
    - Total check phase time

      $= O(kne)$

      too difficult to present analysis of e
    - $= O(km/n^3)$ when $e = m/n^4$

      $=$ **O(m)** when $k = O(n/\log_\sigma n)$,

Problems with previous methods

1. They permit a large expected number of surviving regions compared to the expected number of true approximate matches

2. When a surviving region is first located, the method move directly to full dynamic programming computations

Myers method

- Handle insertion, deletion as well as mismatches

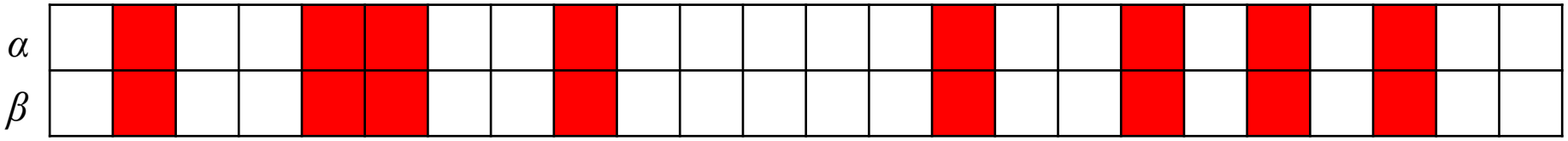
- $O(km^{p(\varepsilon)}\log m)$ when length of the sub pattern is $\log_{\sigma} n$

- Too complex for detailed discussion

Myers method

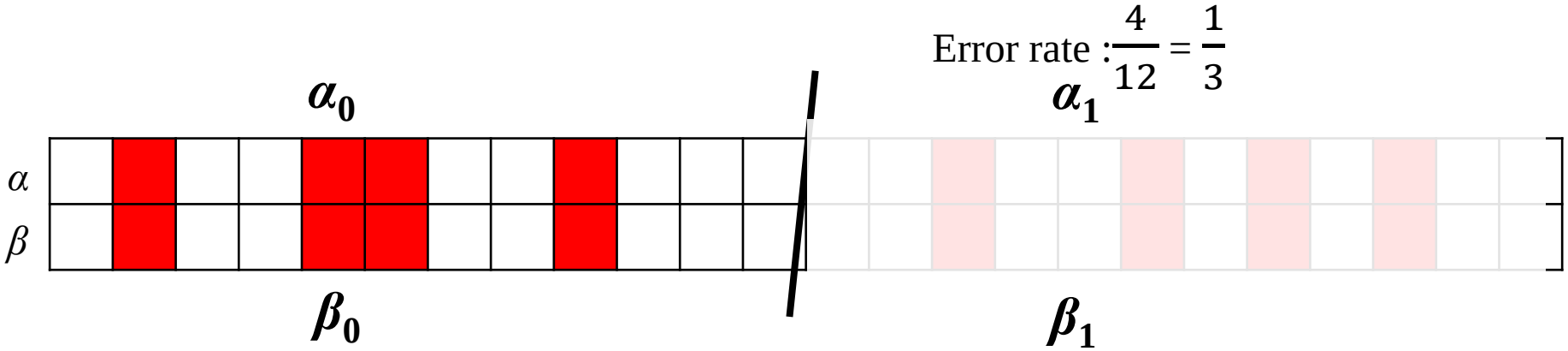
- **Lemma 12.3.4**
  - Suppose α ε-matches β. Then β can be divided into two substrings β_0 and β_1 such that $\beta=\beta_0\beta_1$, and either α_0 ε-matches β_0 or α_1 ε-matches β_1.
  - Let α = α0α1, where |α0| is assumed equal to |α1|.

$$\text{Error rate}: \frac{8}{24} = \frac{1}{3}$$



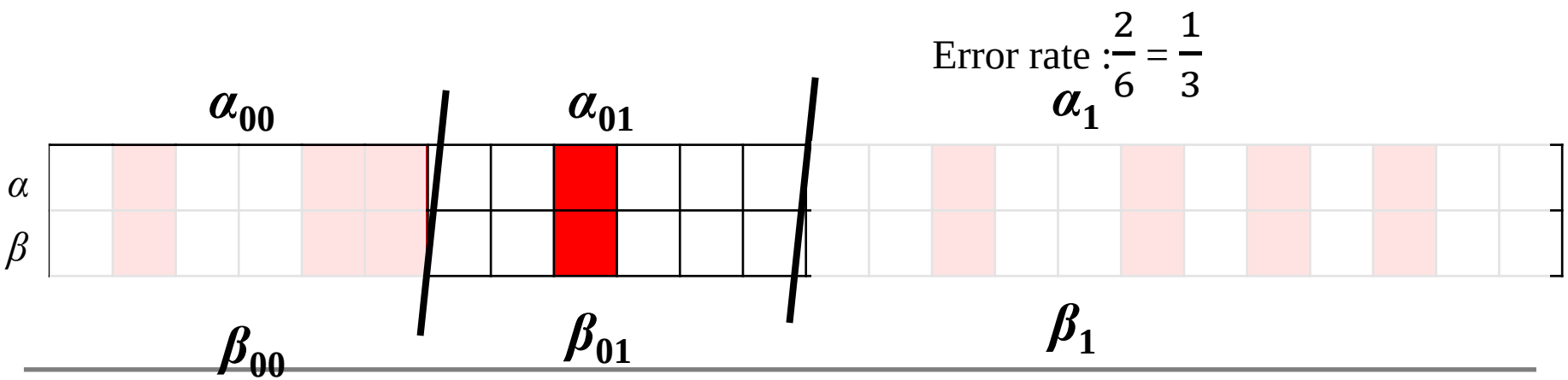
Myers method

- **Lemma 12.3.4**
  - Suppose α ε-matches β. Then β can be divided into two substrings β_0 and β_1 such that $\beta=\beta_0\beta_1$, and either α_0 ε-matches β_0 or α_1 ε-matches β_1.
  - Let α = α0α1, where |α0| is assumed equal to |α1|.



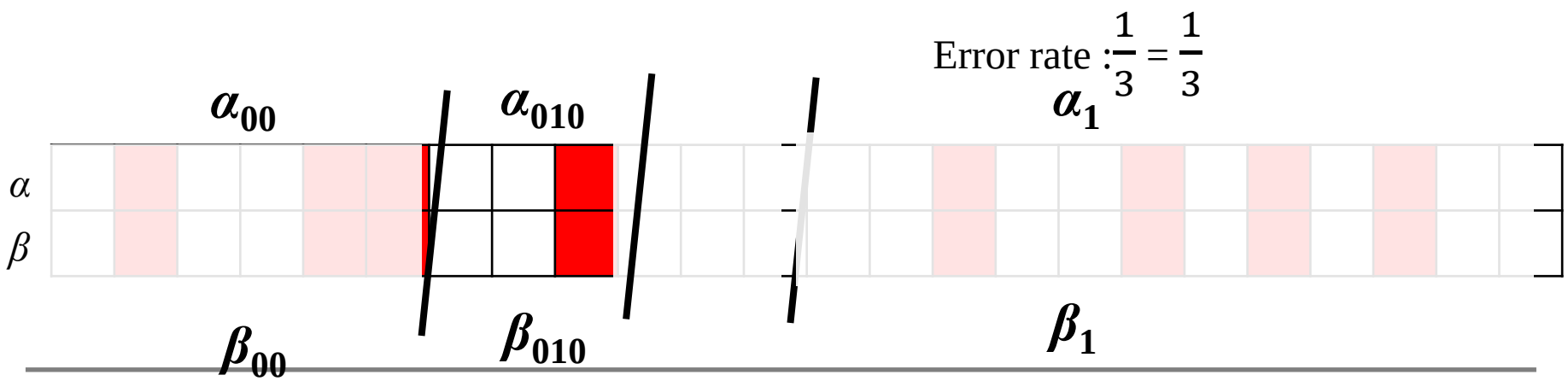
Myers method

- **Lemma 12.3.4**
  - Suppose α ε-matches β. Then β can be divided into two substrings β_0 and β_1 such that $\beta=\beta_0\beta_1$, and either α_0 ε-matches β_0 or α_1 ε-matches β_1.
  - Let α = α0α1, where |α0| is assumed equal to |α1|.



Myers method

- **Lemma 12.3.4**
  - Suppose α ε-matches β. Then β can be divided into two substrings β_0 and β_1 such that $\beta=\beta_0\beta_1$, and either α_0 ε-matches β_0 or α_1 ε-matches β_1.
  - Let α = α0α1, where |α0| is assumed equal to |α1|.



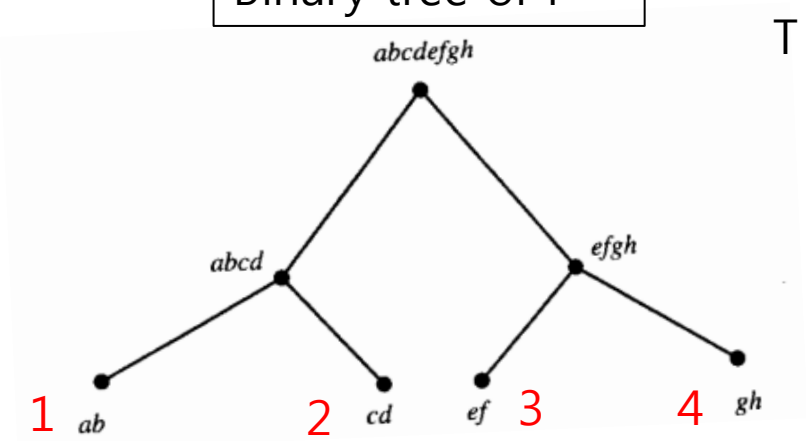
Myers method

- **d-neighborhood**
  - **example>**
  - only {a,b}, S = aba, d=1
  - 1-neighborhood : {bba,aaa,abb,aaba,abaa,baba,abba,abab,ba,aa,ab}
  - Condense(remove prefix): {bba, aaa, aaba, abaa, baba, abba, abab}

Myers method

- First iteration
  - T has already been preprocessed into some index structure
  - Partition P into subpatterns of length $\log_\sigma m$
  - Construct the condensed d-neighborhood for each subpattern in $\boldsymbol{P}$
  - Find all locations of substrings in test T that exactly match one of the substrings in one of the condensed d- neighborhoods.

Binary tree of P



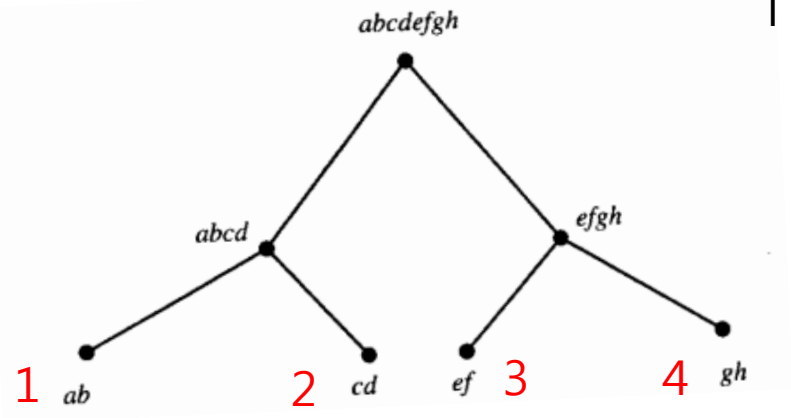
T

1							4				3					2				

Myers method

- Successive iteration
  - extend each initial surviving match to become a ε-match between substring twice as long as those in the current surviving match

Binary tree of P



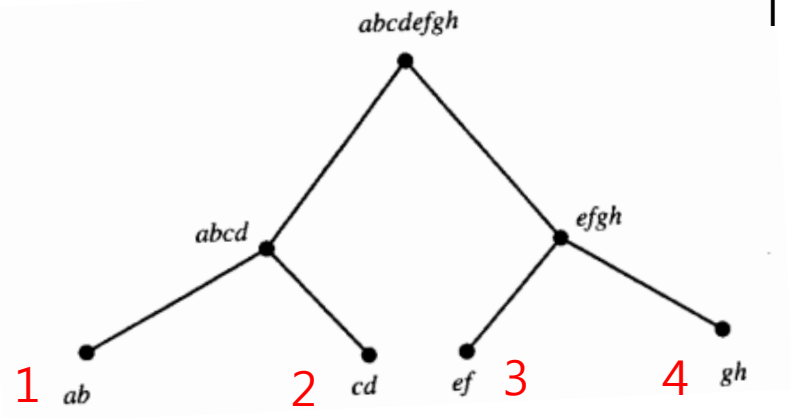
T

1	2					3	4				3	4			1	2			

Myers method

- Successive iteration
  - extend each initial surviving match to become a ε-match between substring twice as long as those in the current surviving match

Binary tree of P



T

1	2	3	4	1	2	3	4		1	2	3	4			1	2	3	4		

Myers method

- Time Complexity
  - when length of the sub pattern is $\log_{\sigma} n$
  - total runtime is $O(km^{p(\varepsilon)} \log m)$
  - $p(\epsilon)$ convex increases very slowly

Myers method

- Exposition given above is only the outline of Myer's method, without any analysis

- Unlike BYP and CL, error rates that establish sublinear running times do not depend on the length of P

- Although expected running times for both CL and Myers' are sublinear
  - CL : due to multiplicative factor that is less than one
  - Myers : due to an exponent that is less than one

Final comment of exclusion method

- Developed with motivation of searching large DNA and protein databases for approximate occurrences of query strings

- weak for the case of **protein database search,** error rates as high as 85% are great interest when comparing protein sequences

- The most effective practical database search methods can be considered as exclusion methods
  ex) BLAST, FASTA, and variants