

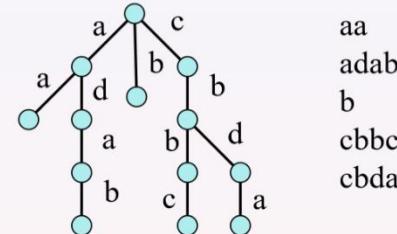
סיכום אלגוריתמים למחוזות / אורי שביט

הרצאה 1

<https://www.youtube.com/watch?v=N70NPX6xgsA> - Suffix trees

– עץ ששומר קבוצה של מחוזות, בעץ יש על כל קשתתו בוודד כאשר כל מסלול מייצג מילה בודדת, יכולנו לתאר לכל מילה ענף בוודד בעץ אך רצינו לחסוך במקומות.

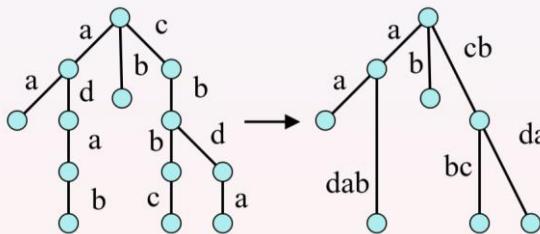
A set of strings can be represented by a **trie**:



דבר זה לא מתקיים באופן כללי, אם יש לנו קבוצה שאין בה מחוזות שמקורם במחוזות ולכן כל אחת מתאימים לעלה בעץ ולכן נקראת **prefix-free**.

אנו רוצים לחסוך בזיכרון כמה שיותר וכך נוכל לבצע כיווץ אם יש מסלול ללא פיצולים:

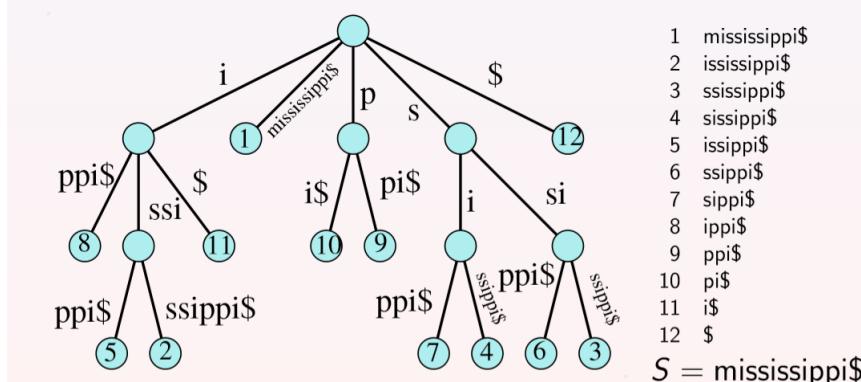
A **compressed trie** is the tree obtained from a trie by removing all vertices with exactly 1 child.



כיווץ הקשיות יוצר trie מכובע, לכל צומת היא עלה או שיש לה לפחות 2 בנים. ניתן לכזוץ רק צמתים שיש להם בן יחיד.

הוספנו את התו \$ כדי שיהיה prefix free

- The **suffix tree** of a string S is a compressed trie of all suffixes of the string S .
- We will usually add a special character $\$$ to the end of S in order to make the set of suffixes prefix-free.



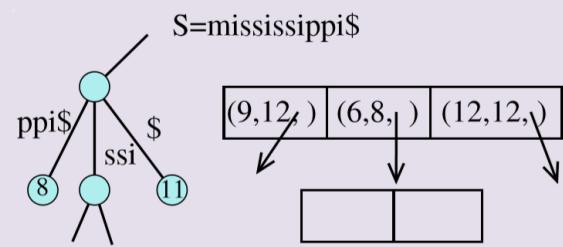
השאלה היא האם המבנה יעיל בזיכרון? יש ח' מחוזות והארוכים חצי ארכיים קצרים וכן נחשב את גודל המבנה : בכל צומת אנו צריכים לשמר בנימ' ותוויות –
נשמר זאת 2n מספרים כנתן מחוזות מקומות מסוימים – אינדקס התחלה והסיום.(בעז סיפות)

Claim

The suffix tree T of a string S of length n can be stored using $\Theta(n)$ space.

Proof

- ① For each vertex, we store a sorted array of its edges.
- ② Each edge can be represented in $\Theta(1)$ space.



בצומת פנימי אנו שומרים מערך אשר שומר את האינדקסים של תת העצים. כמו הדיאלוג היא לנארית במספר הקשיות בעז – ישן מספר העלים הוא בדיק' ח. אין צמתים עם בן בוודד וכן מספר הצמתים הפנימיים הוא לכל היותר 1-ח ולכן יש לכל היותר $2n - 2$ קשיות מכל זאת נctrך בסה"כ (n) זיכרון.

Claim

The suffix tree T of a string S of length n can be stored using $\Theta(n)$ space.

Proof

- ① For each vertex, we store a sorted array of its edges.
- ② Each edge can be represented in $\Theta(1)$ space.
- ③ T has exactly n leaves.
- ④ The number of internal vertices is $\leq n - 1$.
- ⑤ The number of edges is at most $2n - 2$.

כמה זמן לוקח לבנות עץ זהה? מכנים כל סיפה לעז בתורה האלגוריתם הנאייבי הוא (n^2) .

- The naive construction algorithm (add the suffixes of S to the tree one by one) takes $\Omega(n^2)$ time.
- Can be done in $\Theta(n)$ time if $\Sigma = \{1, \dots, n\}$.

יש לנו מחוזת T וaned רוצים לאפשר חיפושים במחוזת למשל מחוזת P

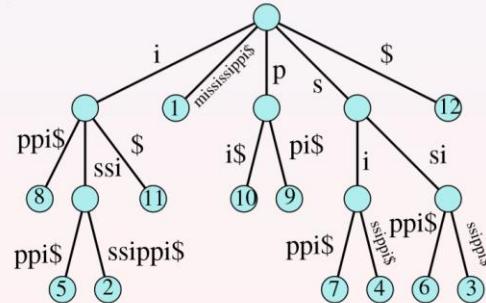
Definition (Text Index)

An index data structure stores a string T and allows the following queries:

- ① Given a string P , does P appear in T ?
- ② Given a string P , find all occurrences of P in T .

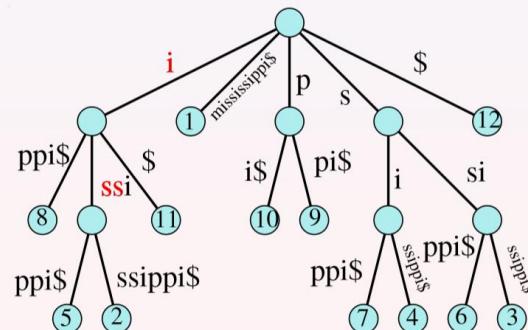
- The data structure: A suffix tree for T .
- Query handling: Traverse the tree according to P .

Example: $P = iss$



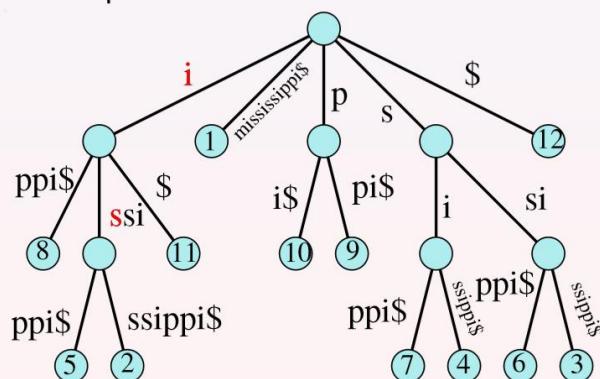
חיפוש כזה יקח זמן לנארו באורך של Δ , יכול להיות מילה מאד ארוכה ואנו לא רוצים להשקיע זמן שלינארו במבנה אלא במחוזת שהוא (בURITY האינדקס), נרד בעץ לפי התווים של P , בדוגמא שלנו יש

Example: $P = iss$

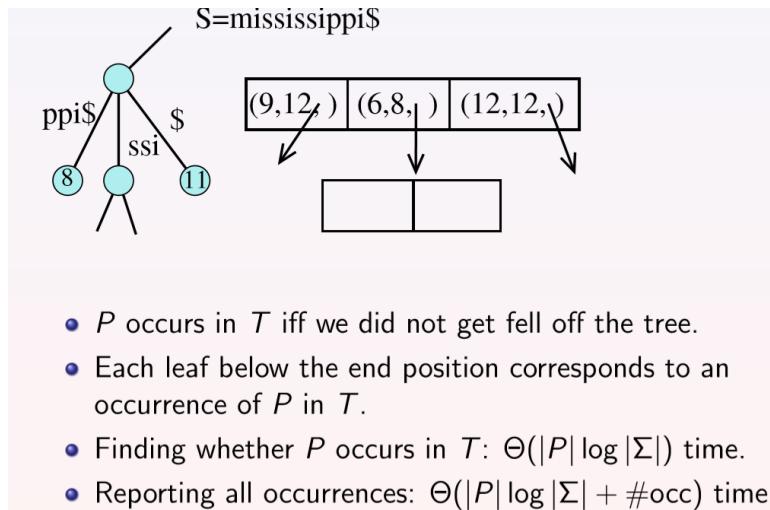


סימנו את הירידה בעץ ונמשיך, אם הצלחנו יש תת מחוזת של T אם נשווה מחוזת אחרת ISP למשל הירידה תכשל וכך לא תופיע בעץ.

Example: $P = isp$



בכל פעם קראנותו של C וירדנו בעץ, כאשר קראנותו הינו צריכים לבחור לאיזה בן לרדת בכל פעם – יש למצאו האם S בן שהתוויות שלו מתאימה – חיפוש במבנה העץ, בצורה נאייה נגשים לפי מבנה הנתונים – בכל פעם לפיה המספר למחוזת T . כמובן לינארי באורך המערך –



לכל תוויות התו הראשון שונה ולכן מספר הבנים של צומת לא יכול להיות גדול מ"א"ב ולכן לינארי בגודל הא"ב, היישומים שונים ולכן למשל שמירת DNA הא"ב בגודל 4.

זמן הירידה בעץ הוא $(|\Sigma| * |P|)O$ ככל אוט B ואז חיפוש עבור כל תו. השוואה של תווים היא קבועה, בכל צעד בחירה של ירידה לבן במקרה הגראע. אם הא"ב גדול אז ניתן לשפר את הזמן – למיין את המערך ע"פ אותיות הקשתות – חיפוש ביניaries על גודל המערך.

המקרה כי גרכו הוא שלכל אות B יש לבחור בן לרדת ולכן $\log |\Sigma| * \log |P|$.
פתרון אחר הוא טבלת גיבוב – ניתן למצוא לאיזה בן לרדת בזמן קבוע $O(|P|)$.

מציאת כל המופיעים ע"י סריקת כל הבנים – כמובן בשלב הירידה בעץ וסרייקת תת העץ, הירידה בעץ כמו קודם וכן צריך את כמות המופיעים OCC. השתמשנו בתוכנה שאין צמתים עם בן בודד – 1-ה עליים לכל היתר.

ניתן לבנות מבנה השומר מספר מחוזות ונבדיל בניהם עם \$ ע"מ ליצור עף סיפות מוכלי

For a set of strings $\mathcal{S} = \{S_1, \dots, S_k\}$ over alphabet Σ , the **generalized suffix tree** is the compressed trie containing all suffixes of the strings $S_1\$_1, \dots, S_k\$_k$, where $\$, \dots, \$$ are distinct characters that do not appear in Σ .

A **palindrome** is a string S such that $S = S^R$.

Maximal Palindrome Problem: Given a string S , find the longest substring of S which is a palindrome.

Naive algorithm: For $i = 2, \dots, n$, find the LCP (longest common prefix) of $S[i..n]$ and $S[1..i-1]^R$, and the LCP of $S[i..n]$ and $S[1..i-2]^R$.

Example

mississippi

הבעיה שלנו היא למצוא פלינדרומ מקסימלי – למצוא תת מחוזת ארוכה ביותר שהיא פלינדרומ, $ississi$, ונסה את כל האפשרויות הראשית – פלינדרומים באורך זוגי ואי זוגי :
לכל פלינדרום אי זוגי יש מרכז – ממנו נלק צעד ימינה וצד שמאליה ונשווה, ולכן נctrur על כל תו במחוזת לעשות זאת, לכל נקודת התחלה נקבל פלינדרום אחר שזו נק' התחלה שלו

A **palindrome** is a string S such that $S = S^R$.

Maximal Palindrome Problem: Given a string S , find the longest substring of S which is a palindrome.

Naive algorithm: For $i = 2, \dots, n$, find the LCP (longest common prefix) of $S[i..n]$ and $S[1..i-1]^R$, and the LCP of $S[i..n]$ and $S[1..i-2]^R$.

Example

mississippi
 LCP(ssippi,ssim)=ssi
 ↑ ↑
 $S[6..n]$ $S[1..4]^R$

פלינדרום זוגי – לולאות על כל מרוח בין אותיות

Example

mississippi

$\text{LCP}(\text{sissippi}, \text{sim}) = \text{si}$

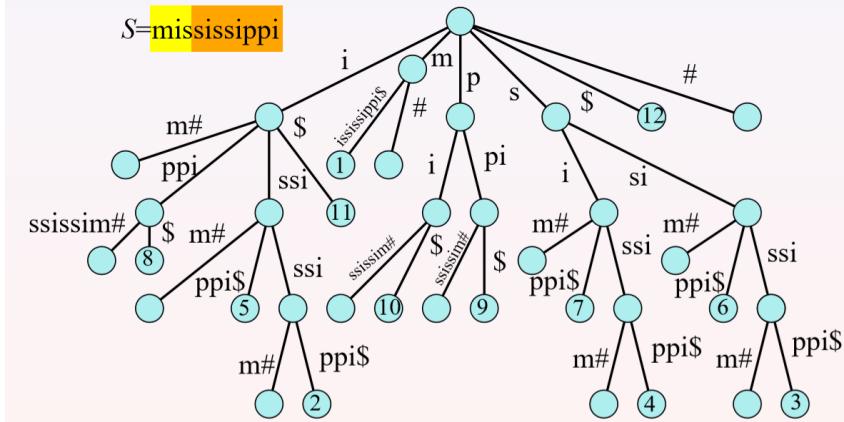
$\uparrow \quad \uparrow$

$S[4..n]$ $S[1..3]^R$

זמן ריצה ($O(n^2)$) כלומר עבור כל אות במילה ועבור כל אחת לכל היותר (n) השוואות.

ניתן לעשות זאת ב($O(n)$) – ונשווה בהםם כלומר הרישא המשותפת הארוכה ביותר – בהינתן המחרוזת נבנה עץ סיפות מוככל עבור המילה וההופכית שלה כאשר השתמש בתווים מבדילים. אנו רוצים למצוא את העלים המשותפים. מבניית העץ המסלול הארוך ביותר – האב הקדמון הći משותף של הענפים הוא הפינדרום הארוך ביותר. למציאת הרישא המשותפת – מוצאים אב קדמון משותף מציאות העלים היא ע"י מצביעים. מציאת האב הקדמון היא גם בעיה קשה – לעיתים בעץ כל פעם יכול להיות ענף ארוך מאוד ונעלה ($O(n)$). עבור כל 2 עליים יש למציא אב קדמון.

If T is a generalized suffix tree for $\{S, S^R\}$ then $S[i..n]\$$ and $(S[1..i - 1])^R \#$ are leaves in T .



בנייה מבני נתונים למציאת אב קדמון בזמן קבוע.

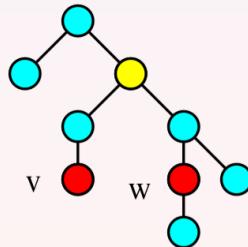
<https://www.youtube.com/watch?v=13m9ZCB8gjw> - LCA

LCA

Lowest Common Ancestor (LCA) data structure

Given a rooted tree T , preprocess T in order to answer the following queries:

- Given two nodes v and w in T , return the lowest common ancestor of v and w .

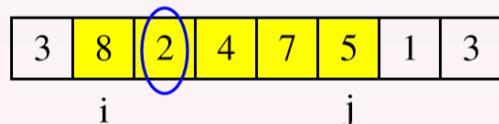


נגד' בעיה שונה : RMQ – נקבל מערך של מספרים ונעשה עיבוד בהין אינדקסים נמצא את האינדקס המינימלי בינם. 0-ui

The Range Minimum Query (RMQ) data structure

Given an array A , preprocess A in order to answer the following queries:

- Given two indices i and j , find a minimum element in the sub-array $A[i..j]$.



אלגוריתם - https://www.youtube.com/watch?v=c5O7E_PDO4U

נראה רדוקציה בין הבעיה ע"מ להראות ששלכות :

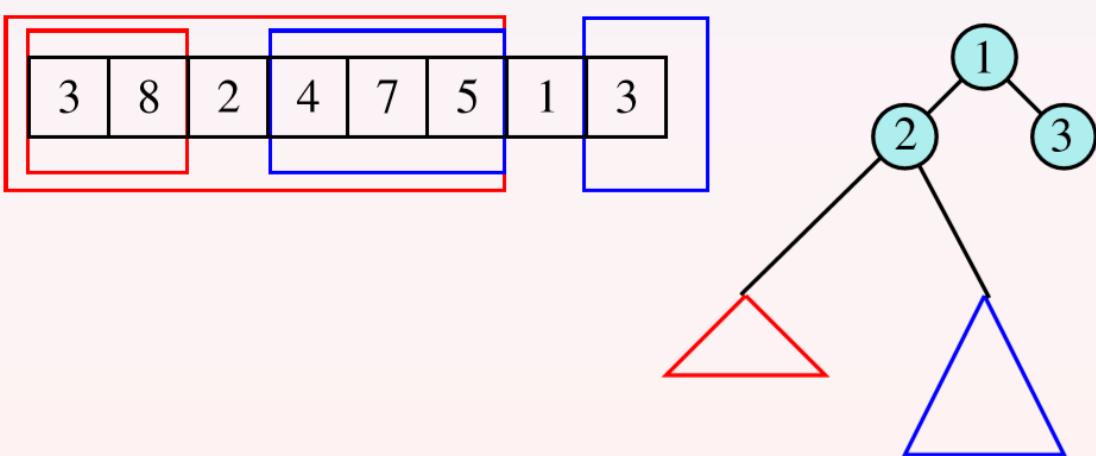
RMQ to LCA

Given an array A , the **Cartesian tree** of A is defined as follows:

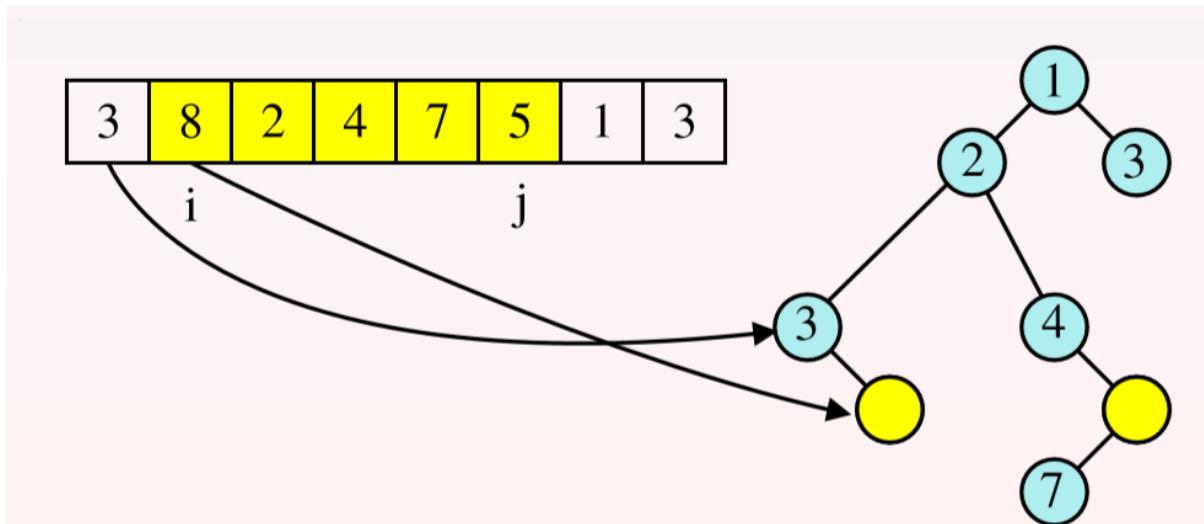
- Let i be an index of the (1st) minimum element in A .
- The root of A has a label $A[i]$.
- If $i > 0$, A has a left child which is the root of a Cartesian tree of $A[1..i - 1]$.
- if $i < n$, A has a right child which is the root of a Cartesian tree of $A[i + 1..n]$.

3	8	2	4	7	5	1	3
---	---	---	---	---	---	---	---

מצאים מספר קטן ביותר במערך , נבנה שורש ותתי מערכים



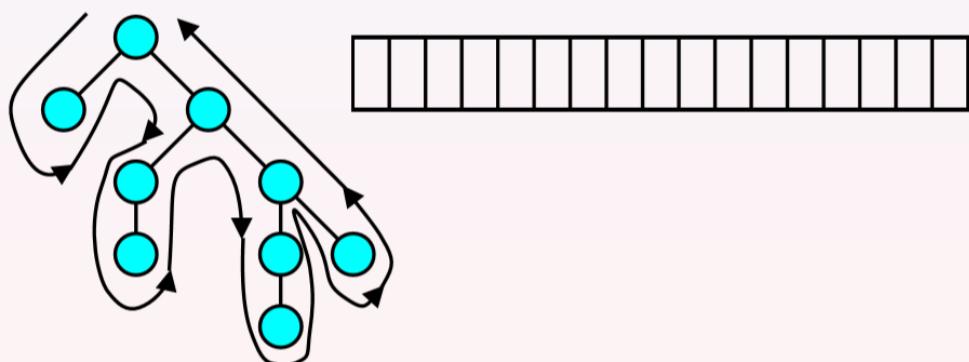
נשמר מצביעים לכל תא לצומת המתאים בעץ, נלך עם הצמתים לאב הקדמון המשותף הקדום ביותר



המספר המינימלי הוא האב הקדמון. נבצע את הרדוקציה בכיוון ההפוך :

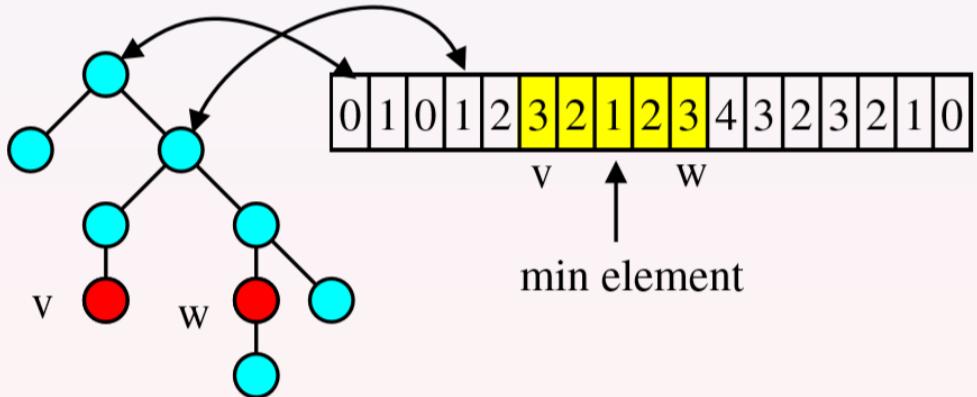
LCA to RMQ

For a tree T , write down the depths of its nodes in a DFS tour of the tree.



נבצע סריקת DFS בכל צומת שנבקר ונרשום את העומק שלו, בכל צעד נרשום את העומק במסלול בו אנו נמצאים, ככל תא במערך נשמר מצביע לצומת שייצר אותו וכן לערך הראשון שייצר אותו.

Given a query v, w , find the minimum element in $A[i..j]$, where i, j are the first corresponding cells. Return the corresponding node



בתת המערך נצטרך למצאו את האיבר המינימלי ונמצא בו את האב הקדמון המשותף ביותר.

זמן עיבוד – קבוע

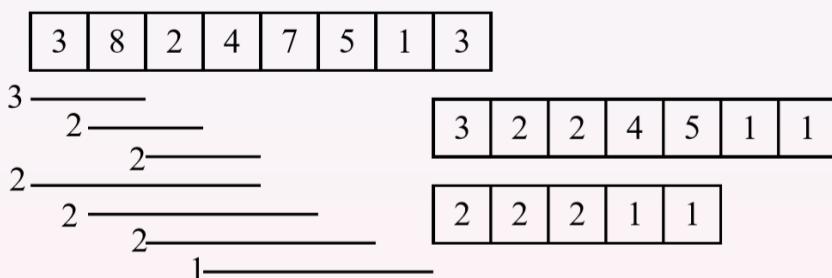
זמן בנייה - $O(n^2)$

מקום - $O(n^2)$

לפנינו RMQ ניתן לשמר מערך דו מימדי. ובכך לענות על שאלות בזמן קבוע, כן ניתן לבנות אותו בזמן ריבועי. נשים לב שאלות RMQ – ע"מ לחסוך מקום לא נשמר את התשובות לכל השאלות, נשמר לכל $a \log_2 n = k$ כלומר לכל תתי מערכim באורך $2, 3, \dots$ וכן נשמר את כל התשובות האפשריות במערכות של המינימום.

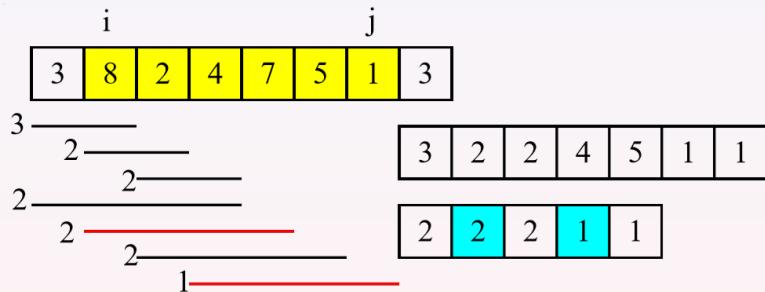
Better structure for RMQ (RMQ1)

- For each interval in A of length 2^k for $k = 1, \dots, \lfloor \log_2 n \rfloor$, store the minimum element in the interval.



נכשין לתאים ומבצעים 2 שאילותות לכל אחד מהצדדים, נחלק ל-2 שאילותות חפות כאן.

- For each interval in A of length 2^k for $k = 1, \dots, \lfloor \log_2 n \rfloor$, store the minimum element in the interval.



Given a query i, j :

- Let L be the maximum power of 2 s.t. $L \leq j - i + 1$.
- Find the 2 intervals of length L whose union is $[i, j]$.
- Return the minimum element on these intervals.

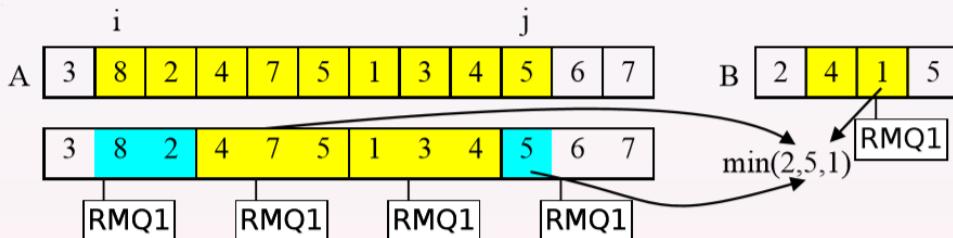
נימוח :

- There are
 - $n - 2 + 1$ intervals of length 2^1 .
 - $n - 2^2 + 1$ intervals of length 2^2
 - \vdots
 - $n - 2^{\lfloor \log n \rfloor} + 1$ intervals of length $2^{\lfloor \log n \rfloor}$.
- Therefore, the space complexity is $\Theta(n \log n)$.
- It is not hard to show that construction time is $\Theta(n \log n)$.

2RMQ

לכל בлок של RMQ 1 נקח מינימום וgam עבורי נקח RMQ .

- Partition A into blocks of size k .
- $B = \text{array containing the min. element in each block}$.
- Store an RMQ1 structure for B , and an RMQ1 structure for each block.



Given a query i, j :

- Find the min. element in the blocks contained in $[i, j]$ using an RMQ query on B .
- Find the min. element in intersection of $[i, j]$ and the block containing i .
- Find the min. element in intersection of $[i, j]$ and the block containing j .



נמצא עבור תת בлок עם 1RMQ את התשובה עבור המינימום, עבור כל תת בлок ניתן לעשות זאת ועבור השלמים ניתן לגשת למערך B אשר מכיר את האיבר המינימלי בכל תת בлок בגודל K , ולכן בו ניתן לקבל תשובה עבור תת מערך עם 1RMQ.

- The RMQ1 structure for B uses $\Theta(\frac{n}{k} \log \frac{n}{k})$ space.
- If we take $k = \Theta(\log n)$, then the space is $\Theta(n)$.
- The space for the RMQ1 structure for each block is $\Theta(\frac{n}{k} \cdot k \log k) = \Theta(n \log k) = \Theta(n \log \log n)$.

פרק 3 שאלות תמיד. זמן מקום – עבור כל בлок בגודל k $k \log k$ ולכן יש $\frac{n}{k}$ מבנים סה"כ

$\log\left(\frac{n}{\log n}\right) = O(\log n)$, אם $n = k^{\log n}$ אז קיבל את הרשים לעיל.

לכן קיבל את המקום הוא $O(n \log \log n)$. יש $\frac{n}{k}$ בלוקים ועבור כל אחד ...

הבעיה היא במבנה הבלוקים אשר צריך מקום $k \log k$

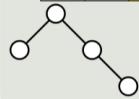
Handling the blocks

Let A and B be arrays with the same Cartesian tree.

Then $\text{RMQi}(A, i, j) = \text{RMQi}(B, i, j)$ for all i, j ($\text{RMQi}(A, i, j)$ is the index of the minimum element in $A[i..j]$).

Example

A	<table border="1"> <tr><td>3</td><td>1</td><td>2</td><td>6</td></tr> </table>	3	1	2	6	$\text{RMQi}(A, 3, 4) = 3$
3	1	2	6			
B	<table border="1"> <tr><td>4</td><td>1</td><td>5</td><td>7</td></tr> </table>	4	1	5	7	$\text{RMQi}(B, 3, 4) = 3$
4	1	5	7			

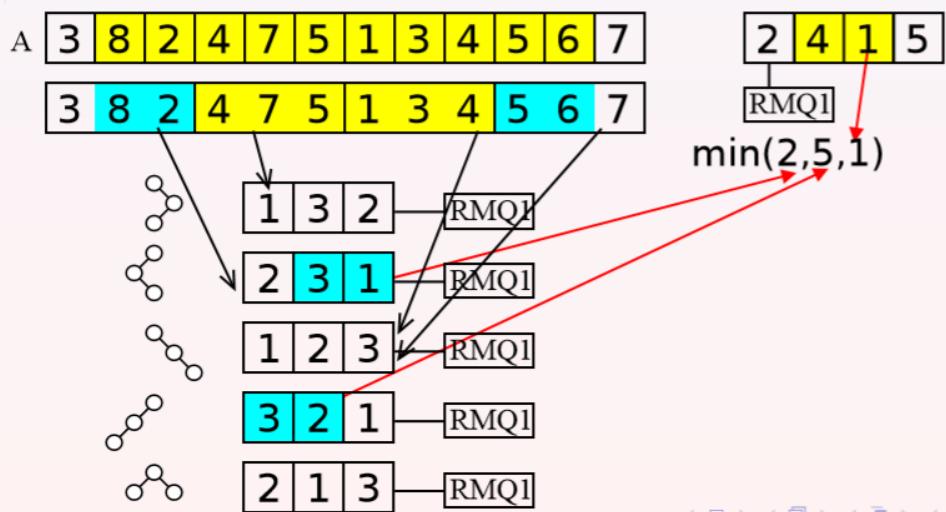


אם נצייר את העץ הקרטזי – אבות מינימליים ברכורסיה – נקבל את אותו עץ מבחינת מבנה, ולכן RMQ שליהם יהיה ה-LCA זהה במיקום.

במקרה לבנות RMQ לכל בלוק, נבחר ערך K וניצר כל עצים אפשריים עם 3 צמתים :

An $\Theta(n)$ space structure for RMQ (RMQ3)

- For each binary tree T with k nodes let $C_T =$ array whose Cartesian tree is T .
Build an RMQ1 structure on every C_T .
- For each block in A whose Cartesian tree is T , store a pointer to the RMQ1 structure of C_T .



לכל עץ אפשרי נבנה מערך שהוא העץ הקרטזי שלו. ועבור כל אחד נבצע RMQ1, במקומות לכל בלוק בא' לקחת RMQ1 ניקח את העץ הקרטזי ונשמר מצביע מהבלוק לבלוק של העץ הקרטזי שבינו מראש.

זמן המוקם עבור מערך B כבר רأינו, אנו צריכים לראות כמה עצים יש עבור K צמתים – אשר 2^{2k} , لكن מבנה :

- The structure stores

- An RMQ1 structure for every binary tree with k nodes.
The number of such trees is $\leq 2^{2k}$, so the memory is $\Theta(2^{2k} \cdot k \log k)$.
- A pointer for each block in A. $\Theta(\frac{n}{k})$ memory.
- An RMQ1 structure for B. $\Theta(\frac{n}{k} \log \frac{n}{k})$ memory.
- Space complexity: $\Theta(2^{2k} \cdot k \log k + \frac{n}{k} \log \frac{n}{k})$.
- Taking $k = \frac{1}{4} \log n$ we get
 $\Theta(\sqrt{n} \cdot \log n \log \log n + n) = \Theta(n)$.

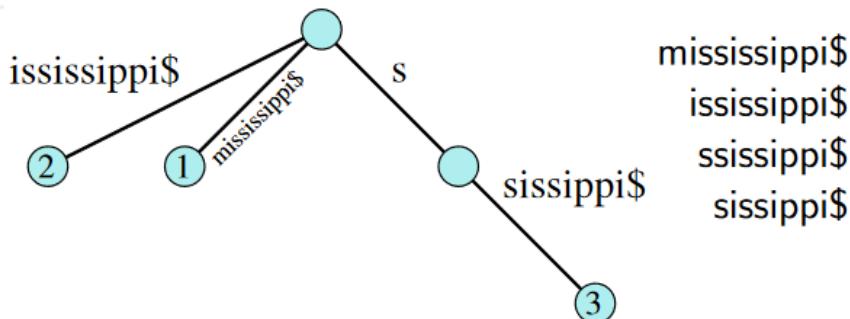
$$\text{הчисוב : } 2^{\frac{1}{4} \log n} = \sqrt{n}$$

- There is a data structure for the RMQ problem that uses $\Theta(n)$ space and answers queries in $\Theta(1)$ time. The data structure can be constructed in $\Theta(n)$ time.
- There are linear time reductions from RMQ to LCA, and from LCA to RMQ.
- There is a data structures for the LCA problem that use $\Theta(n)$ space and answer queries in $\Theta(1)$ time. The data structures can be constructed in $\Theta(n)$ time.

הרצאה 2

ע"ז סיפא – שומר את כל הסיפות של מחזורת
השאלה היא איך בונים אותו, נרצה להראות אלגוריתם יעיל, נראה בצורה נאיבית ראשית:
מעבר אחת אחת ונכנסי אותה לע"ז – ראשית זו Mississipp
לאחר מכן sissippi\\$ ואין לנו רישא משותפת, וכן sissippi\\$ כתת תיה רישא S

The naive algorithm to construct a suffix tree is to add the suffixes to the tree, one by one.



בכל פעם אנו מוסיפים רישא אחד, השאלה היא מה הסיבות?

בכל פעם נצטרך לרדת בע"ז לפי התווים שאנו קוראים במלילה לפי האורך שלה עד שנגיע לפיצול מתאים ונוסיף ענף. יש לנו ח איטרציות וכל אחת יכולה לחתム מעבר בירידה על הע"ז, במקרה הגרוע $O(n * n^2) = O(n^3)$.

- The naive algorithm to construct a suffix tree is to add the suffixes to the tree, one by one.
- Adding a suffix to the tree can take $\Theta(n)$ time. Therefore, the total time is $\Theta(n^2)$.
- We will see a $\Theta(n)$ -time algorithm for constructing a suffix tree, assuming that the alphabet is constant.

נראה כתעת אלגוריתם בזמן לינארי, נניח שהוא קבוע. כל פעם נחזיק ע"ז סיפות עבור הגדלת המחרוזת עבורה בונים את ע"ז הסיפות.

המחשה - https://www.youtube.com/watch?v=VA9m_I6Lqwa

הסברם - https://www.youtube.com/watch?v=uxA__b23t2w

****להסיף** שקוביות 5**

نبנה עברו וואז נוסיף זוז, בכל פעם נוסיף את התו הבא ונבנה, מתחילה מסיפה ריקה.
בכל פעם אנו מרחיבים את הע"ז הקויים.

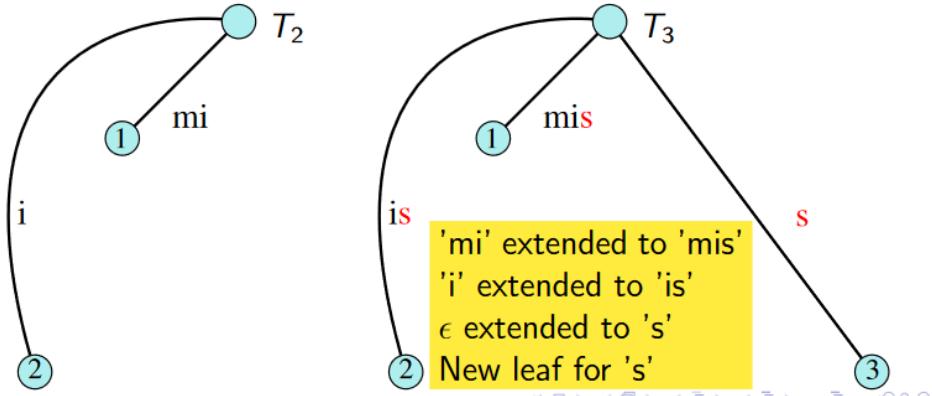
Linear time algorithm

Denote T_i = the suffix tree of $S[1..i]$

The algorithm

```
for  $i = 1, \dots, n$ 
    build  $T_i$  from  $T_{i-1}$ 
```

$S = \text{mississippi\$}$ $S[1..2] = \text{mi}$ $S[1..3] = \text{mis}$

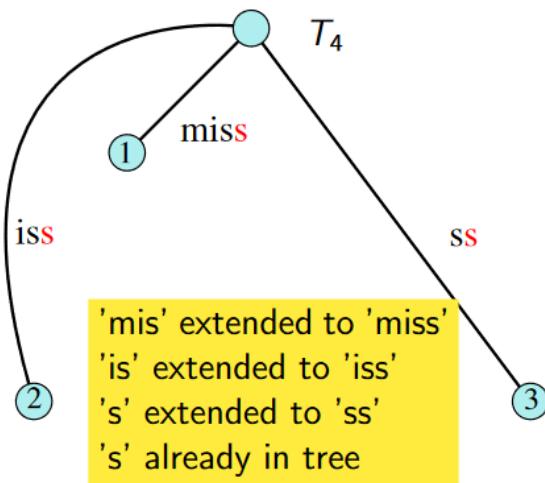


```
for  $i = 1, \dots, n$ 
    for  $j = 1, \dots, i$ 
        find  $S[j..i - 1]$ 
        extend  $S[j..i - 1]$  to  $S[j..i]$ 
```

The case $j = i$ is for adding the suffix $S[i]$ to the tree, which can be viewed as an extension of ϵ to $S[i]$.

יש לנו עץ עבור mis ואנו רוצים עבור miss

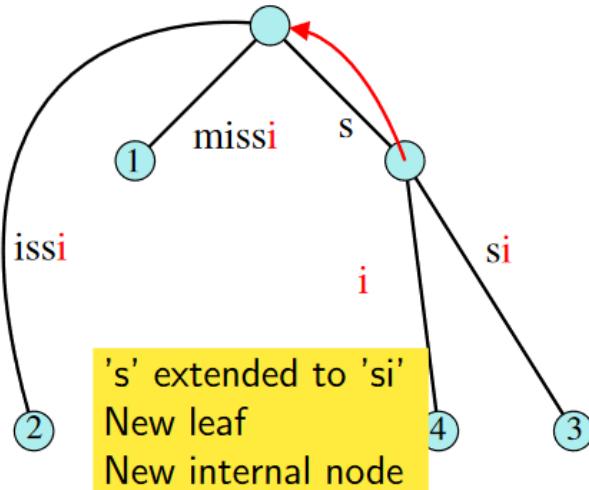
$S = \text{mississippi\$}$ $S[1..4] = \text{miss}$



עבור הענף SS יש בו את הרישא S ולכן הרישות נמצאות גם על הענפים, על כן לא נדרש ענף עבור S.

עבור missi נדרש נספח כמה שלבים, נוספת לכך מילה וכן פירוק הענף SS

$$S = \text{mississippi\$} \quad S[1..5] = \text{missi}$$



כדי להרחיב את הסיפא S נצטרך לפצל את הענף. עבור המילה הבדיקה היא כבר קיימת בזוס. וכך נמשיך

זמן ריצה – נסתכל על פaza – מעבר בין הוספות אחרות, בכל אחת יש לנו איטרציות כאשר גודל כל פעם איטרציה אחת אחת – find ו שינוי extend, בצדקה נאיבית נרד בשורש($b(n)$) ויש ח פазות וכפולה מציאת המקום ח - ($O(n^3)$). גם אם הינו מוצאים את המקום בזמן קבוע אז הינו בזמן ריבועי שבו כבר הינו.

- A **phase** is the extension of T_{i-1} to T_i .
- There are n phases
- In phase i , there are i suffix extensions
- A suffix extension costs $\Theta(n)$ time in the worst case to find the location of $S[j..i-1]$, and $\Theta(1)$ time to make the modification.
- Total time complexity: $\Theta(n^3)$

כללי הרחבה לעז

[אלגוריתם - U](https://www.youtube.com/watch?v=x6j44AtzFmU)

Extension rules

Rule 1 $S[j..i - 1]$ ends at a leaf.

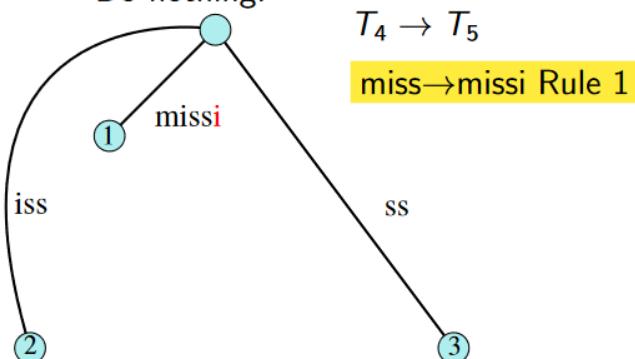
Add a character on the edge entering the leaf.

Rule 2 $S[j..i - 1]$ ends at an internal location, and $S[j..i]$ does not exist in the tree.

Add a new leaf. If $S[j..i - 1]$ doesn't end at a node, add new internal node.

Rule 3 $S[j..i]$ exists in the tree.

Do nothing.



מקרה 2 – מיקום פנימי בענף, עדכון העץ, ייצור צומת חדש ועליה חדש.

מקרה 3 – האות קיימת בעץ ולכן לא נצטרך לעשות כלום.

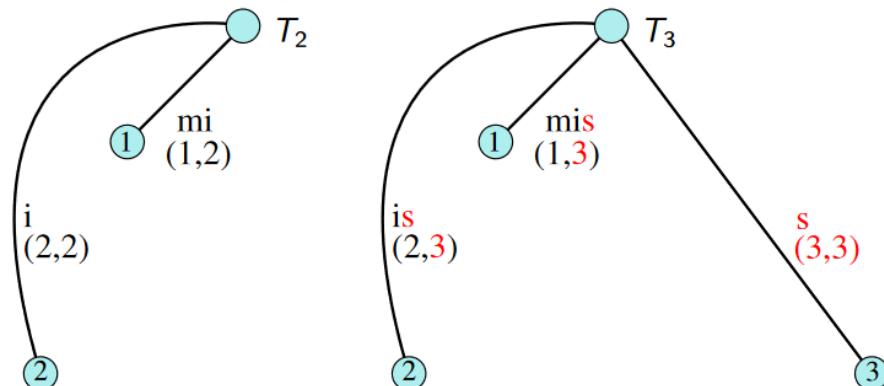
עלות המקרים

כלל 3 – לא עלוה

* כלל 1 – לשנות תווית של קשת, אנו שומרים זוג אינדקסים ולכן רק נצטרך לשנות אינדקס, בכל פעם נחזיק זהה למשהו ערך אורך המחרוזת שבנוינו עד כה, ככלומר מספר פaza.

Cost of an extension

- Extension with Rule 3 doesn't have a cost.
- Extension with Rule 1 also doesn't have a cost if we store the label of the edge to leaf i as $(i, *)$.



כלומר בכלל 1 וכלל 3 אנו לא צריכים לעשות שום שינוי. אנו רק רוצים להבחן האם אנו בכלל 1 או 3.

Claims

Rule 2 $S[j..i - 1]$ ends at an internal location, $S[j..i]$ is not in the tree.

Rule 3 $S[j..i]$ is in the tree.

Claim 1

Consider some phase i . If Rule 2 is used in extension j , then Rule 2 or Rule 3 is used in extension $j + 1$.

Suppose extension j is $abc \rightarrow abcX$.

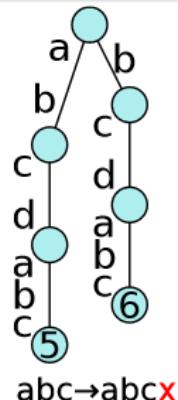
' abc ' ends at an internal location means that ' $abca\alpha$ ' is in the tree for some character α ($\alpha = d$ in the example).

This implies that ' $abca\alpha$ ' is a prefix of some suffix of S , say suffix 5.

' $bca\alpha$ ' is a prefix of suffix 6, so ' $bca\alpha$ ' is in the tree.

Therefore, ' bc ' ends at an internal location.

If ' bcx ' is not in the tree extension $j + 1$ uses Rule 2, o/w Rule 3.



בביצוע ההרחבות לפי הסדר הכלליים משתנים מונוטונית, כלומר ...3...2,2,2...1,1,1.

נניחiamo נמצאים בפazaה i , בהרחבה $j+1$ נשתמש בכלל אחד או הבא, רוצים להרחיב $abc \rightarrow abcx$.

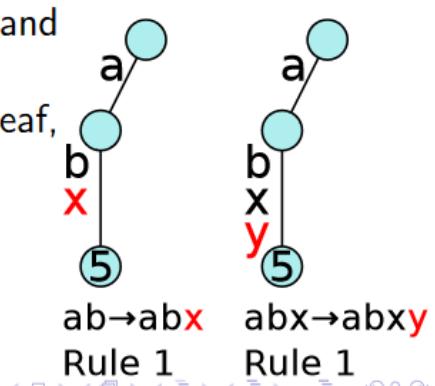
אם מבצעים כלל 3 בפazaה j נבעצע אותו גם $j+1$, הייתה כבר סייא יותר ארוכה בעז. גם הסיפה $abc \rightarrow abcx$ וגם ההרחבה נמצאות. כתע עבר $ab \rightarrow abx \rightarrow abxy$ ולכן המחרוזת הארוכה מכילה את bcx ולכן נדרש להפעיל את כלל 3.

Claim 3

If Rule 1 or Rule 2 is used in extension j of phase i , then Rule 1 is used in extension j of phase $i + 1$.

Suppose extension j is $ab \rightarrow abX$ in phase i and $abX \rightarrow abXY$ in phase $i + 1$.

After the extension $ab \rightarrow abX$, abX ends at a leaf, so in phase $i + 1$ Rule 1 is used.



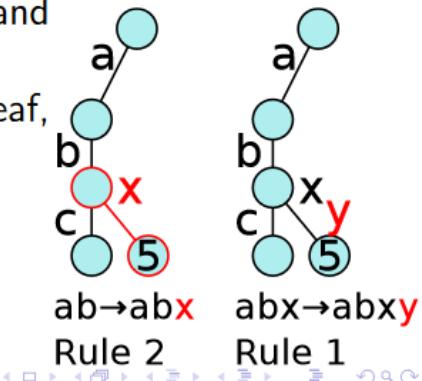
טענהה- אם משתמשים בכלל 1 או 2 ...

Claim 3

If Rule 1 or Rule 2 is used in extension j of phase i , then Rule 1 is used in extension j of phase $i + 1$.

Suppose extension j is $ab \rightarrow abx$ in phase i and $abx \rightarrow abxy$ in phase $i + 1$.

After the extension $ab \rightarrow abx$, abx ends at a leaf, so in phase $i + 1$ Rule 1 is used.



נגיד שציריך להרחיב $abx \rightarrow ab$ בפazaה הבאה נצטרך להוסיף לו תו חדש y , ולכן לא נוסיף תווים מתחת לעלה, אז בפazaה הבאה אנו נהיה בעלה ונוסיף לו אות נוספת ונשמר עליו לעלה.

כל פazaה מונטונית לא יורדת, בכל פazaה מה שהיא 1,2 יופיע לך 1. ברגע שאנו מגיעים לכלל 3 אין עוד מה לעשות בפazaה, ברגע שהגענו לכלל 3 אנו יכולים ליצור אוצה כיון שלא משנה את העץ ואנו יודעים שככל השאר יהיו כלל 3.

תחילת הפazaה – ב1 לא צריך לעשות שינוי על העץ ורק צריך להתחיל מהאינדקס בו עצרנו כזכור נתחיל רק אחרי כלל 1 שלא משנה את העץ וביצע את המשך השינוי כולם כללי 2 עד כלל 3 הראשון.

Improved algorithm

Claim 1 Rule 2 in extension $j \implies$ Rule 2/3 in extension $j + 1$.

Claim 2 Rule 3 in extension $j \implies$ Rule 3 in extension $j + 1$.

Claim 3 Rule 1/2 in phase $i \implies$ Rule 1 in phase $i + 1$.

Phase 10: 11~~2223~~3333

Phase 11: 11111~~223~~333

Phase 12: 1111111~~23~~333

- We can end phase i when Rule 3 is first applied (clm. 2)
- We can start phase i from the index in which phase $i - 1$ ended (clm. 3)

$j \leftarrow 1$

for $i = 1, \dots, n$

while $j \leq i$

 find $S[j..i - 1]$ in T

if $S[j..i]$ exists in T , **break**

 extend $S[j..i - 1]$ to $S[j..i]$

$j \leftarrow j + 1$

While loop is executed $\Theta(n)$ times.
Time complexity: $\Theta(n^2)$.

כעת האלגוריתם נראה, j מסמן האיטרציות – j נשאר משתנה בין האיטרציות כלומר מהמשתנה של הפעזה הקודמת עד לביצוע שניי – כלל 2 – אם הסיפה קיימת בעצם עוצרים ולכן נדרש לעדכן את העץ ולהמשר.

האלגוריתם עכשו הוא יותר יעיל כי אנו מבצעים לא 3⁸³ אלא הרשות הפנימיות מבוצעות לכל היותר 2 פעמים, כיון בכל עמודה יש להלה היותר 2, בכל שורה יש לבדוק 3 פנימי אחד ולכן סה"כ 2 איטרציות של לולאת `while`, כלומר סה"כ n פעמים ($O(n)$ ולכן זמן ריבועי).

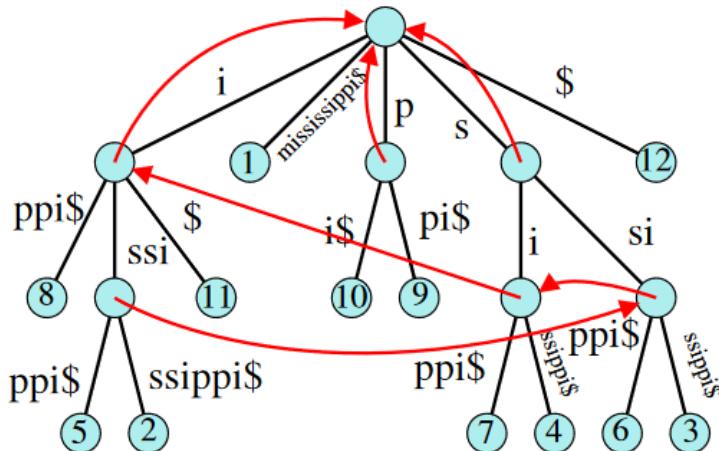
נרצה לשמר את המיקומות בהם הרחכנו, מצביעים למציאות הספויות שאנו רצים להרחיב בצרה עיליה.

מוסיף מצביעים link suffix – לכל צומת פנימית יהיה מצביע לצומת אחר,コレmur מהצומת שלosi נוריד אתosi וונעבור ל*zos* וכך הלאה BWT.

Definition

The **suffix link** of an internal node with string x is a pointer to the node with string $x[2..|x|]$ (if it exists).

The suffix link of the root is the root.



טענה : לכל הצלתיים יהיה suffix link פרט לאחרון

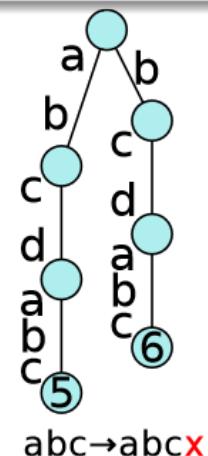
Claim 5

In the current tree, all internal nodes have suffix links, except perhaps the last created internal node.

Suppose extension j is $\text{abc} \rightarrow \text{abcx}$.

A new node is created when Rule 2 is used, corresponding to the string 'abc'.

In extension $j + 1$, a node corresponding to 'bc' will be added if it didn't already exist.



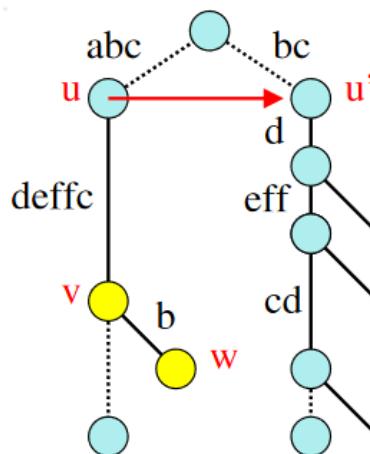
להוסיף

$\text{abc} \rightarrow \text{abc}$ הינו צריכים רק להוסיף c ולכן יש לצומת לפניו מצביע סiphot. ע"מ למצוא את המיקום הבא

כל צומת בגודל χ מצביע לצומת בגודל $1-\chi$ (תתי מחוזות), מצביע הסiphot נתון לנו את המיקום בו אנו צריכים לטפל באירועה הבאה בכל פעם. המקרה הייתר מסובך הוא בפיזול צומת (לאבא יש).

Locating $S[j + 1..i - 1]$ — hard case

- If v was created on extension j , let u be the parent of v .
- u has a suffix link to u' .
- To find the location of $S[j + 1..i - 1]$, go down from u' according to the label of (u, v) .



אך של u יש מצביע סיפות, כתה נצטרך לרדת בעץ עד שנגיע, בירידה אנו צריכים לרדת למקום הנכוון ולפצל את הקשת, ייצור עלה חדש וצומת חדש ועדכון מצביע הסיפות של v למקום שמצאנו.

הירידה בעץ – אנו רוצחים להרחיב את הסיפה $dbceff$ ולמצואו את המיקום שלה בעץ, עבורנו דרך מצביע הסיפות וצריך למצאו את $ceff$, $bceff$, נרד למטה ובירידה כשהגענו ל ff אנו יודעים שהתוים יהיו שוויים כיוון שהיבת להיות קיימת בעץ! המחרוזת שאנו מחפשים חיבת להיות בעץ כלומר בתווית ולכן חיבבים לבדוק את האות הראשונה ולפיה אנו יודעים לאן לרדת. זמן הירידה תלוי בכמות הצמתים ולא תלוי באורך הסיפה.

ננתן איז זמן כל הירידה בכל האלגוריתם :

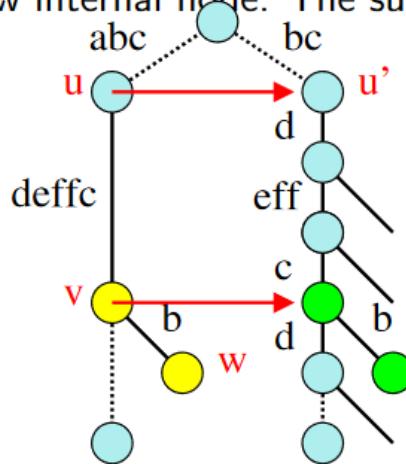
זמן היריצה תלוי במספר התוצאות :

למטה – יורדים לבן (בעייתו) העומק גדול ב-1

למעלה – עומק קטן ב-1 ועלולים לכל היותר ח, בכל איטרציה עושים עלייה אחת לכל היותר

הצדיה – מעבר מ- s למצביע הסיפות עושים זאת לכל היותר ח פעמיים, כיצד משפיעה על העומק?

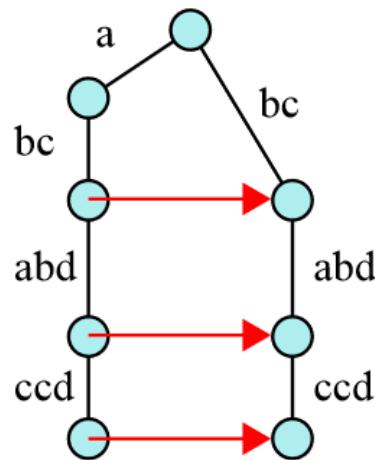
- If v was created on extension j , let u be the parent of v .
- u has a suffix link to u' .
- To find the location of $S[j + 1..i - 1]$, go down from u' according to the label of (u, v) .
- In extension $j + 1$, if xb is not in the tree, a new leaf is created, and perhaps a new internal node. The suffix link of v is created.



המשך ניתוח

Let v a node, and w be its suffix link.

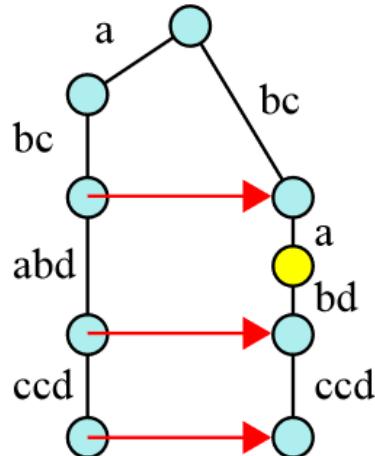
Every node, except perhaps the second, on the path from the root to v has a corresponding node on the path from the root to w .



עוביים לפי מציין הסימול ולקן העומק קטן ב-1, אם a לא היה קיים היו יכולם להשאר באותו מקום, העומק יכול לגודל כלומר יש צמתים שלא קיימים במסלול שלנו, אך יכול רק לקטן ב-1 לכל היותר.

There may be nodes on the path from the root to w that have no corresponding nodes in the path to v .

Thus, when moving from a node to its suffix link, the depth can decrease by 1, not change, or increase.



תשוז למעלה והצדה לכל היותר n פעמים. כמשמעותם העומק הוא 0 ובסיום העומק הוא לכל היותר n , כי גובה העץ חסום באורך המילה ולכן לא יכולות להיות יותר מידי תשוזות למטה. כל פעם שמבצעים תשוזה למעלה יכולים לבצע עוד אחת למטה וכן נוכל לבצע 2 תשוזות למטה וכן עברו הצדה גם כן ולכן חסום ב3 !.

קיים שמספר התשוזות שהוא עושים חסום ב(α) 0 ולכן זמן הריצה לינארי!

There are 3 types of movements:

- Down: Moving from a node to its child.
 - Increases the depth by 1.
- Up: Moving from a node to its parent.
 - Performed at most n times.
 - Decreases the depth by 1.
- Side: Moving from a node to its suffix link.
 - Performed at most n times.
 - Depth can decrease by 1, not change, or increase.

At the start the depth is 0, and at the end $\leq n$. Therefore, the total number of down movements is $\leq 3n$.

הרצאה 3

ע"ז סיפות – מבנה בו שומרים את כל הסיפות של מילה.
חישוב מחוזת ק בתוך המחרוזת.

suffix array - אם נמספר את הסיפות של מחרוזת S (הכי נמוך ו-\$ הכי גבוהה), ה-\$array של S יהיה מערך עם מספרי הסיפות ממוחנים לפי סדר לексיקוגרפי של הסיפות.

מערך סיפות – לוקחים את כל הסיפות של מחרוזת (האינדקס – באיזה אינדקס מתחילה הסיפה) שומרים את הפרמטרציות, מערך הסיפות הוא מערך של מספרים. אם מצירים את ע"ז היסופות יהיו ממוחנים לפי הסדר של האות הראשונה בתוויות.

The **suffix array** of a string S contains all suffixes of S sorted in lexicographic order.

Example

$S = \text{mississippi\$}$

1	mississippi\$	8	ippi\$
2	ississippi\$	5	issippi\$
3	ssissippi\$	2	ississippi\$
4	sissippi\$	11	i\$
5	issippi\$	1	mississippi\$
6	ssi\$	10	pi\$
7	sippi\$	9	ppi\$
8	ippi\$	7	sippi\$
9	ppi\$	4	sissippi\$
10	pi\$	6	ssi\$
11	i\$	3	ssissippi\$
12	\$	12	\$

The suffix array:

8	5	2	11	1	10	9	7	4	6	3	12
---	---	---	----	---	----	---	---	---	---	---	----

אם עוברים על העלים משמאלי לימין הם אלו בדיקת המספרים של הtree suffix. היתרון של המערך על העץ הוא המקום, שומר רק את האינדקסים, שומר בדיקת ח.

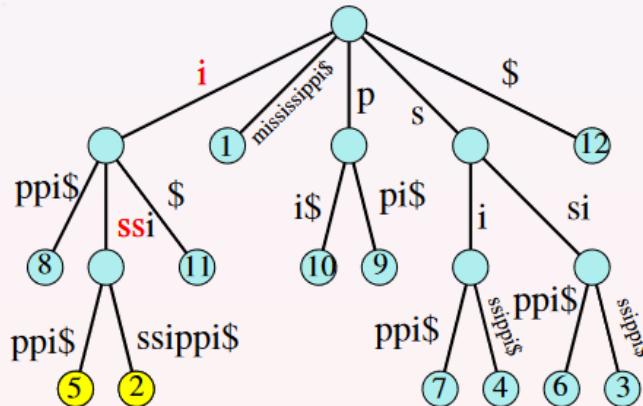
לכל צומת נשור מערך לבנים, כל מערך מכיל שלשה – אינדקס הקשת ומצבי לבן.

ראינו בשיעור הראשון שמספר הצטמים הפנימיים לכל היוטר ח, מספר הקשותות לכל היוטר ח'2 ולכן סה"כ יהיו לכל היוטר ח'2 שלשות, סה"כ 6 ולכל מערך את הגודל שלו – ח סדר גודל של ח'7 תא זיכרנו.

כשרצינו לחפש מחוזת בע"ז הסיפות ירדנו בע"ז הסיפות לפי מחרוזת החישוב ואם לא יצאנו מהעץ למשוז מופיעה בע"ז.

- If P appears in T then the indices of its occurrences form an interval in the suffix array of T .

8	5	2	11	1	10	9	7	4	6	3	12
---	---	---	----	---	----	---	---	---	---	---	----



- How can we find the endpoints of the interval?

במערך הסיפות 5, 2 יוצרים קטע רצוף במערך הסיפות ולכן רצחים להדפיס את כל המופיעים צריך למצוא את כל המופיעים, את האינטראולים ואז להדפיס. המטרה – למצוא את האינטראול הצהוב עבור מערך חיפוש.

נעשה זאת ע"י חיפוש ביןארי.

נסמן (j) הסיפה בז בגודלה, j המינימלי,

אם עושים השוואה בין (i) T לק התשובה תהיה לא, לא, לא...כן, כן... ולק החיפוש מונוטוני וניתן לעשות חיפוש ביןארי.

$$T = \text{mississippi\$}, P = \text{iss}.$$

8	5	2	11	1	10	9	7	4	6	3	12
---	---	---	----	---	----	---	---	---	---	---	----

- Denote $T(j) = j\text{-th largest suffix.}$
- To find the left endpoint of the interval, find the minimum j such that $T(j) \geq P$.

- a 8 ippi\$
 5 issippi\$
 2 ississippi\$
 11 i\$
 1 mississippi\$
- c 10 pi\$
 9 ppi\$
 7 sippi\$
 4 sissippi\$
 6 ssippi\$
 3 ssissippi\$
- b 12 \$

שומרים את מערך הסיפות ולכן ניתן להשוות ולקבל תשובה.

$T = \text{mississippi\$}$, $P = \text{iss}$.

8	5	2	11	1	10	9	7	4	6	3	12
---	---	---	----	---	----	---	---	---	---	---	----

- Denote $T(j) = j\text{-th largest suffix}$.
- To find the left endpoint of the interval, find the minimum j such that $T(j) \geq P$.

a 8 ippi\$
 5 issippi\$
 c 2 ississippi\$
 11 i\$
 1 mississippi\$
 b 10 pi\$
 9 ppi\$
 7 sippi\$
 4 sissippi\$
 6 ssippi\$
 3 ssissippi\$
 12 \$



15

אלגוריתם חיפוש ביןארי פשוט, $|T|, m = n$ ולכן יש $n \log n$ איטרציות, בכל אחת להשוות בין 2 מחוזות
ולכן מקבל $O(m \log n)$

אנו רוצים משהו יותר עיל, נשתמש בכך שלא נctrar להשוות כל פעם מההתחלתה.

— הרישא המשותפת הארוכה ביותר עbor 2 מחוזות. LCP

- $LCP(A, B) = \text{length of longest common prefix of } A \& B$.
- The alg. maintains $k = LCP(P, T')$ where $T' = \text{last suffix of } T \text{ compared with } P$.

Example

T($n/2$)	<u>a</u> bdbb . . .	k=2
P	<u>a</u> bcd _e	

In this example, $P < T(n/2)$, so the new search interval is $[1, n/2]$, and the next comparison is P with $T(n/4)$.

השוואה הראשונה קבילהנו 2 ע"י השוואת תווים, מכשלו קטע החיפוש יהיה $2/n$. באיטרציה הבאה האמצע $n/4$ ושוב נבצע השוואת.

$$\text{נסמן } l = \text{LCP}(T\left(\frac{n}{4}\right), T\left(\frac{n}{2}\right))$$

Example

T($n/4$)	<u>a</u> bdba . . .	l=4
T($n/2$)	<u>a</u> bdbb . . .	k=2
P	abcde	

Let $l = \text{LCP}(T(n/4), T(n/2))$.

There are 3 cases: (1) $l > k$ (2) $l < k$ (3) $l = k$.

אם $l < k$ אז אנו יודעים שא' התווים הראשונים זהים לפחות ולכון אנו יודעים מה תהיה תוצאה ההשוואה עבור $T\left(\frac{n}{4}\right)$, ולכן אם $l < k$ הזמן הוא קבוע.

אם $k < l$ אז שוב יודעים שא' התווים הראשונים זהים ולכון עד התו ה' יש שווין אחריו אין ולכן שוב ($n/4$) T לא שום השוואת.

Example

T($n/4$)	<u>a</u> aahg . . .	$l=1$
T($n/2$)	<u>a</u> bdbb . . .	$k=2$
P	abcde	

If $l < k$ then $P > T(n/4)$ as

- $P[1..l] = T(n/2)[1..l] = T(n/4)[1..l]$
- $P[l+1] = T(n/2)[l+1] > T(n/4)[l+1]$

המקרה $k=l$ ולכן נבצע השוואה רק מהתו $k+1$, ואחריו נראה את ההבדל וכן נעדכן את k .

Example

T($n/4$)	<u>a</u> bcd . . .	$l=2$
T($n/2$)	<u>a</u> bdbb . . .	$k=2$
P	abcde	

If $l = k$, compare P and $T(n/4)$ starting from the $(k+1)$ -th character, and update the value of k .

אנו רוצים לספור את כל השוואות במהלך כל האלגוריתם, נספור כמה השוואות בין $T(n/4)$ ל- P לתו ב- $T(n/4)$. כל פעם שביצעים השוואה מגדילים את k ב-1.

Example

T($n/4$)	<u>a</u> bcd . . .	$l=2$
T($n/2$)	<u>a</u> bdbb . . .	$k=2$ $k=4$
P	abcde	

If $l = k$, compare P and $T(n/4)$ starting from the $(k+1)$ -th character, and update the value of k .

מספר הפעמים שיכולים להיות השוואות מוצלחות לכל היותר m ומש' ההשואות הלא מוצלחות ככל היותר כמו החיפוש הבינארי. וכך סה"כ $m + \lceil \log_2 n \rceil$

- The value of k is non-decreasing during the algorithm.
- When case 3 occurs, each character comparison increases the value of k by 1, except the last comparison.
- Total number of character comparisons $\leq m + \lceil \log_2 n \rceil$.

- The value of k is non-decreasing during the algorithm.
- When case 3 occurs, each character comparison increases the value of k by 1, except the last comparison.
- Total number of character comparisons $\leq m + \lceil \log_2 n \rceil$.
- Time complexity: $\Theta(m + \log n)$.

השאלה איך מצאים את k ?

The algorithm needs to compute the $LCP(T(i), T(j))$ values in $\Theta(1)$ time.

אנו רצים לשמר מערך LCP עבור j ,

The LCP Lemma

For $i < j$,

$$LCP(T(i), T(j)) = \min \{LCP(T(k), T(k+1)) : i \leq k < j\}.$$

עבור i התווים הראשונים שווים, עבור $i+1$ אם LCP גדול ממש אז יש שוויון, לפחות פעמי אחד ש- $S=LCP$ ולכן התו גדול לפחות ב-1 וכן LCP הוא בדיק S . כיוון שהמחוזות בסדר לקסיקוגרפי התו הבא חייב להיות גדול.

****28****

נשמר מערך עזר עבור LCP –

- If we can compute $\text{LCP}(T(i), T(j))$ in constant time, then suffix array searching takes $\Theta(m + \log n)$ time.
- Store an array LCP with $\text{LCP}[k] = \text{LCP}(T(k), T(k + 1))$.
- $\text{LCP}(T(i), T(j)) = \min\{\text{LCP}[i], \text{LCP}[i + 1], \dots, \text{LCP}[j - 1]\}$.
- This is the RMQ problem, so after preprocessing that takes $\Theta(n)$ time and space, $\text{LCP}(T(i), T(j))$ can be computed in constant time.

נעשה זאת בדיק כמו שעשינו עם RMQ ובהינתן 2 אינדקסים נוכל למצוא את המינימום בזמן קבוע.

אם אנו שומרים את מערך LCP אז אנו מגדילים את גודל המבנה, لكن שמיירת עוד α תא זיכרון וכן מאיצ' את תהליך החיפוש.

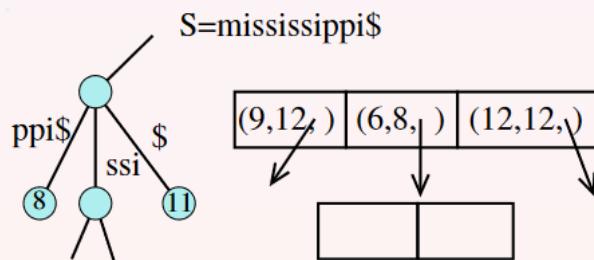
השוואה בין המבנים :

Summary

$$n = |T|, m = |P|.$$

	Query time	Construction Time	Space
Suffix array	$\Theta(m + \log n)$	$\Theta(n)$	$\Theta(n)$
Suffix tree ¹	$\Theta(m \log \Sigma)$	$\Theta(n)$	$\Theta(n)$
Suffix tree ²	$\Theta(m)$	$\Theta(n \Sigma)$	$\Theta(n \Sigma)$
Suffix tree ³	$\Theta(m)$	$\Theta(n)$ expected	$\Theta(n)$
Suffix tray	$\Theta(m + \log \Sigma)$	$\Theta(n)$	$\Theta(n)$

1: Each node store its children in a table of size $\deg(v)$.



העץ הראשון מהשיעור הקודם.

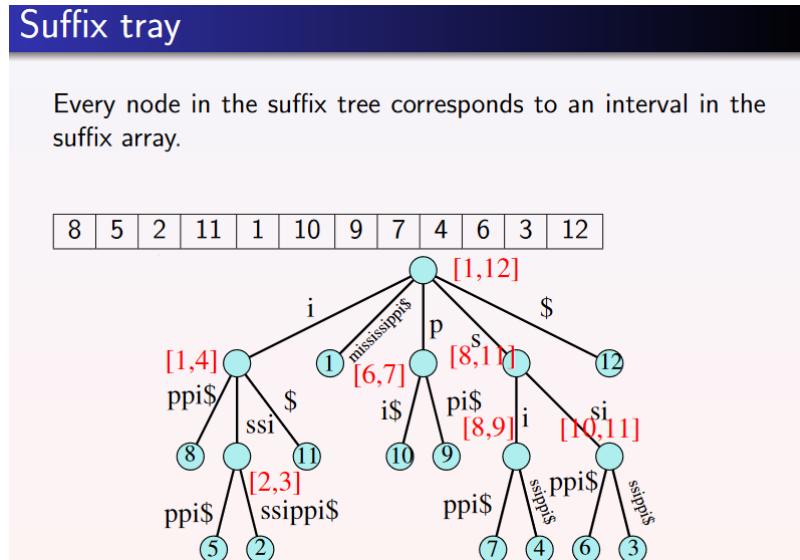
2- ניתן לשמר את מערך בגודל הא"ב ולכן בכל פעם יקח זמן קבוע אך הזיכרון יקח יותר מקום.

אלגוריתמים למחוזות / סיכום מאות אורי שביט

3- טבלה *hash* כאשר המפתח הוא האות הראשונה בקשות הבנים ולן זמן החיפוש יהיה ($O(m)$, זמן החיפוש הוא צפוי). המיקום נשאר (a) וכך . טבלה *hash* כאשר הא"ב הוא המפתח ומגיע לשולש של אינדקסים ומצבייע לצומת בעץ.

אם הא"ב גדול השימוש בטבלאות *hash* נראה הכי טוב.

array tree – שילוב של *suffix tray* – 37

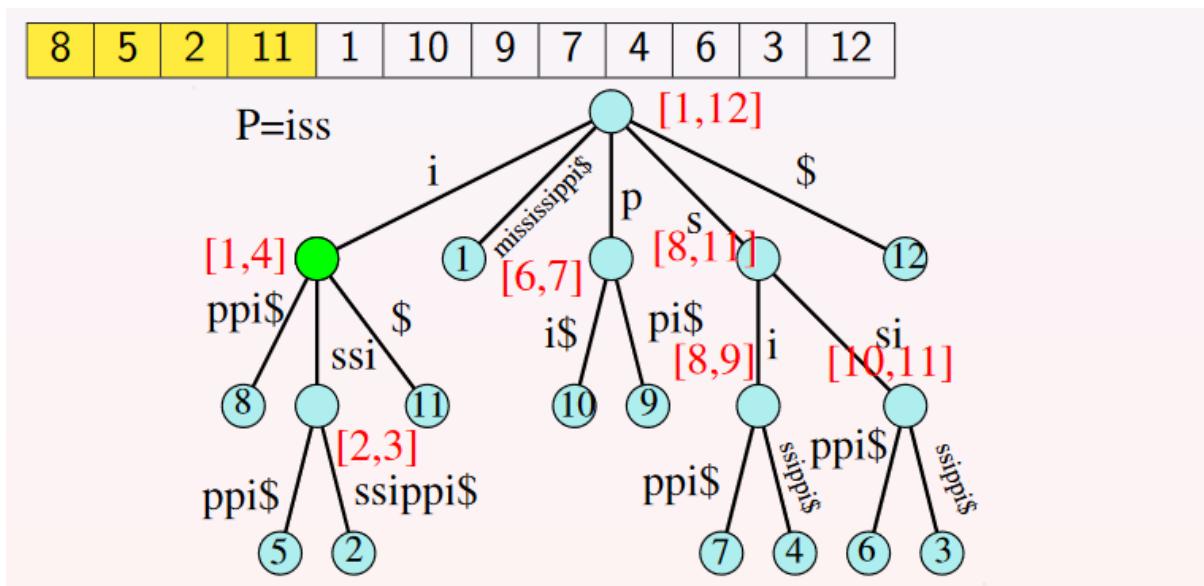


שמירת גם מערך סיפות וגם עז סיפות, כל העליים היצאים יוצרים קטע.

אינדקס התחלת וסיום במערך הסיפות – עליה מספר שהוא שנמצא במקומ מסוים במערך הסיפות.

הרעין יהיה – נחפש מחוזת מסוימת למושל, נתחל את החיפוש בעז הסיפות, בשלב מסוים נעבור מחיפוש בעץ למערך. נוכל להשתמש במידע שיש לנו כדי לCKER את החיפוש הבנאי – לנצל את אינדקס שהגענו אליו. למשל עבור *sis* – נרד ל'ז ונעשה חיפוש על 11'8 אם מספיק קטן החיפוש יהיה מאד מהיר. מספר האיטרציות הוא $c \log n$ גודל הקטע.

בדוגמא *iss* – יורדים עם ? וממשיכים את החיפוש



אם קטע במערך הסיפות יהיה בגודל הא"ב

סיה"כ יהיו 2 שלבים – חיפוש בעץ ($O(\log m)$) ואז מערך הסיפות – ($O(\log \text{size})$) גודל החיפוש

- Keep both suffix tree and a suffix array.
- Go down in the suffix tree until reaching a node with at most $\text{poly}(|\Sigma|)$ leaves in its subtree.
- Search the corresponding interval I in the suffix array.
- Time:
 - Tree: $\Theta(m)$.
 - Array: $\Theta(m + \log |I|) = \Theta(m + \log |\Sigma|)$.

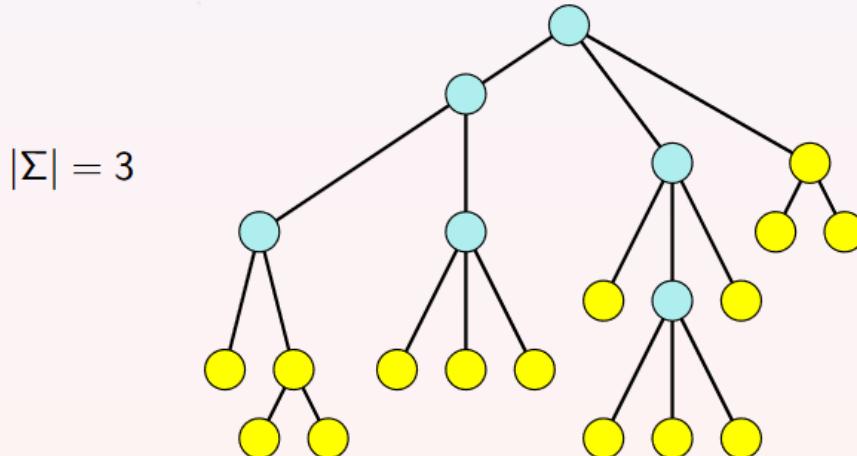
א"ב העץ – צומת בעץ מסווג 3 אם מוס' העלים הצאצאים שלו קטן מגודל הא"ב

Σ -tree

Definition

A node v in the suffix tree is **type 3** if it has less than $|\Sigma|$ leaves in its subtree.

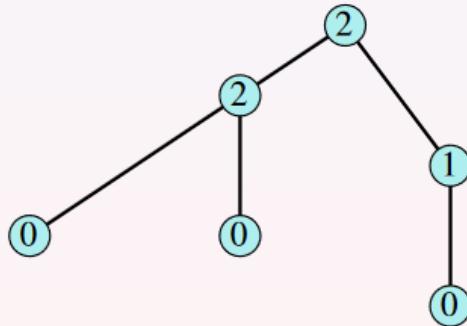
The **Σ -tree** is the tree obtained by removing type 3 nodes from the suffix tree.



נקח את העץ ונוריד את הצלמיים הצהובים . נחלקם ל-3 סוגים לפי כמות הבנים.

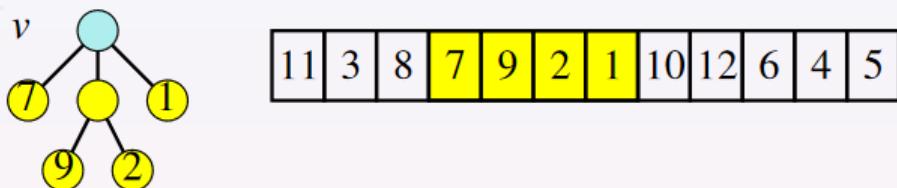
Definition

A node v in the suffix tree is **type 0/1/2** if it is not type 3, and it has $0/1/\geq 2$ children in the Σ -tree.



כל צומת נטפל בו לפחות סוג שלו, סוג 3 הוא כל הצמתים הצהובים – אם מספר העלים בתת העץ שלו קטן ממש מגודל הא"ב – פחות מ-3.

Type 0 nodes



Fact

*A type 0 node has at most $|\Sigma|^2$ leaves in its subtree.
(it has at most $|\Sigma|$ children, each with less than $|\Sigma|$ leaves)*

אם יש לנו צומת מסווג 0 – עלה בעץ הכהול – מספר העלים הצעאים שלו לא יותר מהא"ב בריבוע, כל הצעאים של Σ מספור הבנים שלו לכל היותר הא"ב ולכן יש לו לפחות כפול הא"ב מכאן קיבלנו זאת.

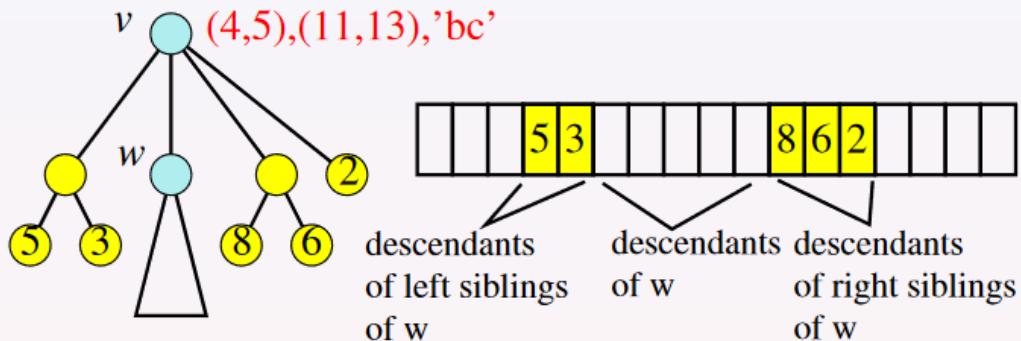
זאת אומרת שם הגענו אליו ניתן לבצע חיפוש בינארי במערך ולכן זמן החיפוש $O(\log |\Sigma|)$

החיפוש הבינארי במערך הסיפות יקח

**להסיף 47

מקרה ב – סוג 1 – יש לנו את כל הבנים הצהובים לפני ואחריו, נשמר את התוויות של הקשת לשא, משווים את התו לטו הקשת רואים שאותו דבר ולכן ניתן לרדת לכיוונו.

Type 1 nodes



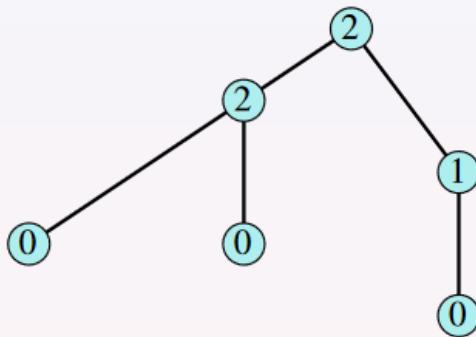
- The descendants leaves of the left (right) siblings of w form interval I_1 (I_2) of length at most $|\Sigma|^2$ in the suffix array.
- v stores the label of the edge (v, w) and the start/end positions of the intervals I_1 and I_2 .
- Time complexity: $\Theta(1)$ if we go down to w , $O(m + \log |\Sigma|)$ otherwise.

נשמר את המצביע, הקטע שמתאים לעליים הצאצאים משמאל ומימין לו.

גודל כל קטע – חסום על גודל הא"ב בריבוע, ולכן אם צריך לרדת לאחד מהם ולפניהם יקח קודם.

כלומר אם רואים שהתו בקשת גדול ממה שיש לנו כעת ב \mathcal{P} נלך לצאצאים השמאליים ונמשיך מכאן ובהמשך.

כעת נשאר לנו צמתים מסוג 2 – יש לנו מעט צמתים כאלה – נטען כי יש \mathcal{S}/n כאלה.



Fact

There are at most $n/|\Sigma|$ type 2 nodes.

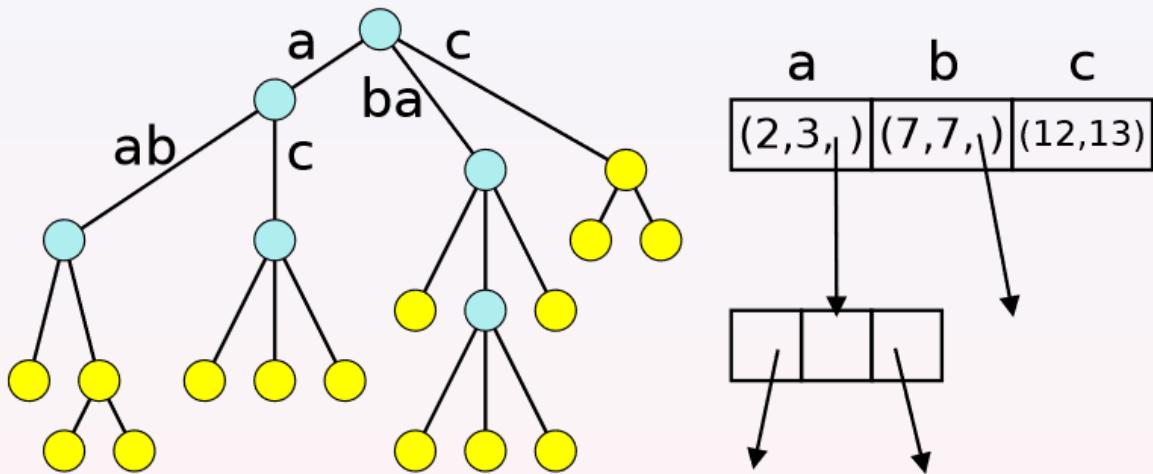
Proof

- A type 0 node has at least $|\Sigma|$ leaves in its subtree (o/w the node would be type 3).
- \Rightarrow There are at most $n/|\Sigma|$ type 0 nodes.
- \Rightarrow There are at most $n/|\Sigma| - 1$ type 2 nodes.

צמתים מסוג 0 – כל העלים מסוג 0 – יש בדיק n עליים,

נמתק צמתים עם בן בודד – קל להראות שסכום הצמתים הפנימיים קטן שווה $1-n$, אך שלמו

Type 2 nodes



Fact

There are at most $n/|\Sigma|$ type 2 nodes.

- For each type 2 node, store pointers to its children using an array of size $|\Sigma|$. Total space used is $\Theta(n)$.
- Time complexity: $\Theta(1)$.

זמן קבוע למציאת הבן!

כשאנו שומרים את הבנים של צומת מסווג 2 – שומרים מצביע לבן אם כחול ואם צהוב מספיק לשומר את הקטע שמתחאים לו שבגדל הא"ב ולכן ניתן לעבור ולחפש במערך הסימפטו.

- For a type 3 child of a type 2 node, store the interval of the child (the size of the interval is $< |\Sigma|$).

סיכום

- Going down the suffix tree takes $O(m)$ time.
- When searching the suffix array, the search interval has size $\leq |\Sigma|^2$ so the search takes $O(m + \log |\Sigma|)$ time.
- Therefore, the total time is $\Theta(m + \log |\Sigma|)$.

סוג 2 – הבנים בתוך מערך בזמן קבוע

הטיפו בטו מוסיים לvoke זמן קבוע ולכן זמן הירידה בעז לוקח $O(m)$ החיפוש במערך לוקח m וגודלו החיפוש ולכן צמצמנו אותו ולכן סה"כ זמן החיפוש יהיה $|S| \log O(m + \log |\Sigma|)$

מקום – כגודל המילה לפי הטענות יש לכל היוצר (n) צמתים מסוג 2,0,1,2 ששמורים אותם!

Type	Space per node	number of nodes
0	$\Theta(1)$	$\leq n$
1	$\Theta(1)$	$\leq n$
2	$\Theta(\Sigma)$	$\leq n/\Sigma$

- Total space is $\Theta(n)$.

mississippi\$	ippi\$
ississippi\$	issippi\$
ssissippi\$	ississippi\$
sissippi\$	i\$
issippi\$	mississippi\$
ssippi\$	pi\$
sippi\$	ppi\$
ippi\$	sippi\$
ppi\$	sissippi\$
pi\$	ssippi\$
i\$	ssissippi\$
\$	\$



הרצאה 4 - Construction of Suffix Arrays

זיו 0522882481 – פרויקט מיני דקן

הא שימצא עם טעות אחת, כל התת מחוזות שמתחלות בז' גם.

בහינתן מחוזת S מעל א"ב Σ אנו רוצים לסדר את כל הסופיות של S לפי סדר לקסיקורוגרפי. נראה אלגוריתם לבניית מערך סיפות, יש יתרון כאשר רץ בזמן לינארי גם אם א"ב לא קבוע.

מטרה- לסדר את המחרוזות לפי סדר א"ב, באלגוריתם של *mergesort* יודעים לעשות זאת אך הבעיה היא ההשוואה בין 2 מחרוזות בכל שלב שיכול להקחת ($n \cdot \log n$). (המחרוזת $aaaaaaa...aaaaa$ ביחס ל- $aaaaaaa$).

- פונקציה שומרת סדר - לכל $y < x$ מתקיים $f(y) < f(x)$.

עובדיה – אם f שומרת סדר ונתונה קבוצה S במקור של f אזי סידור הקבוצה זהה לשידור $\{f(x) : x \in S\}$:
לדוגמא:

$f(x) = x + 1$. Sorting $S = \{4,1,3\}$ is equivalent to sorting of $\{5,2,4\}$.

נקטין את גודל הא"ב :

- Σ_s – סדרה ממינית של כל סוג האותיות ב- S . $\Sigma_s = \{a \in \Sigma : a \text{ appears in } S\}$.
- $r_s(a)$ – מספר סוג האותיות המופיעות לפני a ב- Σ_s - מיקום האות בקבוצה כאשר מסדרים את האותיות בסדר עולה. $r_s(a) = |\{b \in \Sigma_s : b < a\}|$
- $r_s(S) = r_s(S_1) \dots r_s(S_n)$ – מחליפים כל תו במחרוזת במקומם שלו בקבוצה.

$S = mississippi$$.

$\Sigma_s = \{i, m, p, s, \$\}$. $r(i) = 0, r(m) = 1, r(p) = 2, r(s) = 3, r(\$) = 4$

$r(S) = 103303302204$

Sorting the suffixes of S is equivalent to sorting the suffixes of $r(S)$.

החלפנו את האותיות במספרים לפי הסדר ולכן שוקל למיון, לדוגמא :

Sorting the suffixes abad,bad,ad,d of abad is equivalent to sorting the suffixes
0102,102,02,2 of 0102.

המחרצת המחרוזת ע"ב הטרנספורמציה, אנו למשה רוצים לצמצם את הא"ב לפי הבעיה.

העברת סידור הסופיות לפי סדר לקסיקורוגרפי ולפי סדר לקסיקורוגרפי שהגדכנו לעיל זהות כפיה שניתן לראות :

mississippi\$	ippi\$	103303302204	02204
ississippi\$	issippi\$	03303302204	03302204
ssissippi\$	ississippi\$	3303302204	03303302204
sissippi\$	i\$	303302204	04
issippi\$	mississippi\$	03302204	103303302204
ssippi\$	pi\$	3302204	204
sippi\$	ppi\$	302204	2204
ippi\$	sippi\$	02204	302204
ppi\$	sissippi\$	2204	303302204
pi\$	ssippi\$	204	3302204
i\$	ssissippi\$	04	3303302204
\$	\$	4	4

יצרים מערך בגודל המחרוזת ובו רושמים את מיקום האות בפונקציה $r(S)$ אז במקומות של i נרשם 0 . הזמן לוקח זמן לינארי לפי radixsort של המערך.

In order to compute $r(S)$ efficiently:

- ① Create an array C with $C[i] = s_i$.
- ② Sort C .
- ③ Compute the ranks of the characters.

Example

mississippi\$ \Rightarrow

m	i	s	s	i	s	s	i	p	p	i	\$
---	---	---	---	---	---	---	---	---	---	---	----

 \Rightarrow

i	i	i	i	m	p	p	s	s	s	\$
---	---	---	---	---	---	---	---	---	---	----

 $\Rightarrow r_S(i) = 0, r_S(m) = 1, \dots$

Time complexity is $\Theta(n \log n)$.

If the alphabet of S is $\{0, \dots, n^c\}$ then time is $\Theta(n)$.

כעת נתאר את האלגוריתם : דומה מאד לאלגוריתם המיוון של mergesort – לוקחים את הקבוצה ומחלקים אותה ל 2 קבוצות בגודל שווה וברקורסיה ממינים כל חלק.

1. אם מחלקים את הסיפות לפי מיקום התחלת זוגי ואי זוגי של האינדקס במחוזות

2. ממינים ברקורסיה את 2 הקבוצות בנפרד

3. מיזוג הקבוצות

שימוש ברקורסיה – בעיית המיוון של השורות לא ציר לעובוד ע"מ שנוכל לעובוד ברקורסיה, ייצור מחרוזת חדשה.

- ➊ Sort the odd suffixes.
- ➋ Sort the even suffixes.
- ➌ Merge the odd and even suffixes.

103303302204	103303302204	03302204	02204
03303302204	3303302204	04	03302204
3303302204	03302204	103303302204	03303302204
303302204	302204	2204	04
03302204	2204	302204	103303302204
3302204	04	3303302204	204
302204	03303302204	02204	2204
02204	303302204	03303302204	302204
2204	3302204	204	303302204
204	02204	303302204	3302204
04	204	3302204	3303302204
4	4	4	4

המיון מנוקדת מבט של מערך לפי אות ההתחלה :

5	11	1	9	7	3	8	2	10	4	6	12
8	5	2	11	1	10	9	7	4	6	3	12

The algorithm works on arrays that contain indices of suffixes.

ההשוואה לפי איזו סיפא יותר גדולה בסדר לקסיקורגי.

הקלט עבור האלגוריתם הוא מחוזות באורך n מעל א"ב $\Sigma = \{0, 1, \dots, n-1\}$.

For $a, b \in \Sigma, c(a, b) = a \cdot n + b$.

For a string $S = s_1s_2 \dots s_n$ over Σ , let $c(S) = c(s_1, s_2)c(s_3, s_4) \dots c(s_{n-1}, s_n)$.

לכן כעת $(S) c$ היא מחוזת באורך $\frac{n}{2}$ מעל א"ב $\Sigma' = \{0, \dots, n^2 - 1\}$.

מה שנעשה ע"מ לוקחים 2 אותיות במחוזת המקורית, לוקחים 2 אותיות רצופות ומסתכמים על מספר בסיסי n , עבור מחוזת S לוקחים את $(S) c$ לדוגמא :

.For $S = 1021 (\Sigma = \{0, 1, 2, 3\})$, $c(S) = 49 (= 1 * 4 + 0 * 1) (= 2 * 4 + 1 * 1) = 49$

הצעד הרקורסיבי:

עבור S מחוזת מעל $\{1, \dots, n-1\} = \Sigma$, עברו המרוזות האזויות של S גבנה את $(S) c$

$$S=103303302204 \quad n=12$$

$$c(S)=12\ 39\ 03\ 36\ 26\ 04$$

$$S = 10\ 33\ 03\ 30\ 22\ 04 = 12 * 1 + 0 * 1, 3 * 12 + 3 * 1 \dots = 12\ 39\ 03 \dots$$

אם ניקח את הסיפא של $(s)c$ מתאימה ל S , אם ניקח את הסיפא הראשונה מתאימה לסיפא שמתחליה באינדקס הבא. מuin הסיפות האזויות של S זהה למuin הסיפות של $(s)c$.

$S = 103303302204$	$c(S) = 12\ 39\ 03\ 36\ 26\ 04$		
103303302204	12 39 03 36 26 04	03 36 26 04	03302204
3303302204	39 03 36 26 04	04	04
03302204	03 36 26 04	12 39 03 36 26 04	103303302204
302204	36 26 04	26 04	2204
2204	26 04	36 26 04	302204
04	04	39 03 36 26 04	3303302204

כלומר מuin $(s)c$ שקיים למuin s כיון שהפונקציה שומרת סדר.

ניתן לפתור את הבעיה ברקורסיה כי ייצרנו מחוזת חדשה שהאורך שלה אחר וניתן למuin את הסיפות של $(s)c$ ובכך מuin את s כיון שהפונקציה שומרת סדר.

בשוב ישנה בעיה – ההנחה באלגוריתם הוא שהקלט הוא מחוזת באורך n מעל א"ב מסוים, כרגע זה לא מתקיים כיון שהוא א"ב הוא מספרים בין $1 - n^2$, נוכל להפעיל את הרדוקציה של $((s)c)r$

- Let S be a string of length n over $\Sigma = \{0, \dots, n-1\}$.
- The odd suffixes of S correspond to all suffixes of $c(S)$.
- Sorting the odd suffix of S is equivalent to sorting all suffixes of $c(S)$.
- Problem: $c(S)$ is a string of length $n/2$ over $\Sigma' = \{0, \dots, n^2-1\}$.
- Solution: $r(c(S))$ is a string of length $n/2$ over $\Sigma'' = \{0, \dots, n/2-1\}$.

כיון שכמות המספרים סופית שוב נוכל למצוא את המרחב כיון שהמספרים גדולים אף יותר הרבה מאשר חזרנו לגדל א"ב שרצינו. נמיין את $((s)c)r$ (בבסיס $radix$ כיון שיש כמהות סופית של מספרים)

$$\{03, 04, 12, 26, 36, 29\} - r \rightarrow \{0, 1, 2, 3, 4, 5\}$$

$S = 103303302204$	$c(S) = 12\ 39\ 03\ 36\ 26\ 04$	$r(c(S)) = 250431$
103303302204	12 39 03 36 26 04	250431 0431
3303302204	39 03 36 26 04	50431 1
03302204	03 36 26 04	0431 250431
302204	36 26 04	431 31
2204	26 04	31 431
04	04	1 50431

הראנו כיצד ניתן למין את הסיפות השחורות בעזרת רקורסיה(תקף לאדומות), נניח שגם יודעים לעשות את

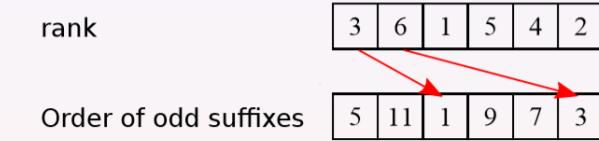
$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n) = O(n \log n)$$

ולכן נרצה למין את הסיפות האדומות בלי רקורסיה.

נגדיר מערך $[i:j]$ שיכיל את סדר הסיפות האי זוגיות, עוברים על הסיפות האי זוגיות לפי סדר עולה ולכל אחת נרשום ביצה אינדקס היא מופעה בסדר לקסיקורגי.

- $i:j$ - האינדקס (מתחל מ-1) של הסיפה ה- $i+1 \dots j$ (ו מתחילה מ-0).

03303302204	0	3303302204
303302204	3	03302204
3302204	3	302204
02204	0	2204
204	2	04
4	4	ε



נמיין את הסיפות האדומות, אם נוריד את התו הראשון מהאדומות נקבל סיפה שחורה.

2 סיפות שחחורות אנו יודעים כבר למין. נחליף כל סיפה אדומה בזוג – אות התחלתה $rank_i$. אם נמיין אותן

בסדר לקסיקורגי זה יהיה שקול למין הסיפות השחורות. נמיין לפי radix sort

כל איטרציה שלו עושים בעזרת counting sort ולבן סה"כ הזמן יהיה לינארי.

03303302204	(0,6)	(4,0)	(0,4)	02204
303302204	(3,1)	(3,1)	(0,6)	03303302204
3302204	(3,5)	(2,2)	(2,2)	204
02204	→ (0,4)	→ (0,4)	→ (3,1)	→ 303302204
204	(2,2)	(3,5)	(3,5)	3302204
4	(4,0)	(0,6)	(4,0)	4
	Sort by 2nd coordinate	Sort by 1st coordinate		
rank	3 6 1 5 4 2			

מ민ים ראשית לפי ערך ה- y ואז לפי ערך ה- x .

שלב המיזוג נותר פתוח, הוא קשה לעשייה. נעשה חלוקה חדשה ע"מ ליצור מיזוג יותר פשוט.

האלגוריתם

- מין הסופיות האי זוגיות ע"י $r(c(s))$
- מין הסופיות הזוגיות ע"ב התוצאה של האי זוגיות radix sort
- מיזוג הסופיות הזוגיות והאי זוגיות.(זמן לינארי)

- ① Sort the odd suffixes by sorting suffixes of $r(c(S)) = r(c(s_1, s_2)c(s_3, s_4) \dots c(s_{n-1}, s_n))$.
- ② Sort the even suffixes. (radix sort)
- ③ Merge the odd and even suffixes.

Step 3 can be done in linear time, but the algorithm is quite complicated.

Simpler algorithm:

- ① Sort all suffixes that begin at position 1 or 2 modulo 3.
- ② Sort remaining suffixes.
- ③ Merge the two lists.

אלגוריתם פשוט יותר:

נמיין לפי i ב 0 מודולו 3. חילוק לשני שלישים.

נמיין את הסיפיות השחורות ברקורסיה ואז את האדומות radix ולבסוף מיזוג.

Definition

An *i-suffix* is a suffix that starts at position $i \pmod{3}$.

103303302204	103303302204	02204	02204
03303302204	03303302204	03302204	03302204
3303302204	303302204	03303302204	03303302204
303302204	03302204	04	04
03302204	302204	103303302204	103303302204
3302204 →	02204 →	204 →	204
302204	204	302204	2204
02204	04	303302204	302204
2204	3303302204	2204	303302204
204	3302204	3302204	3302204
04	2204	3303302204	3303302204
4	4	4	4

ניקח בכל פעם את התווים ונחלק לשולשות, עכשו כל שלשה היא מספר בסיס n עם 3 ספרות.

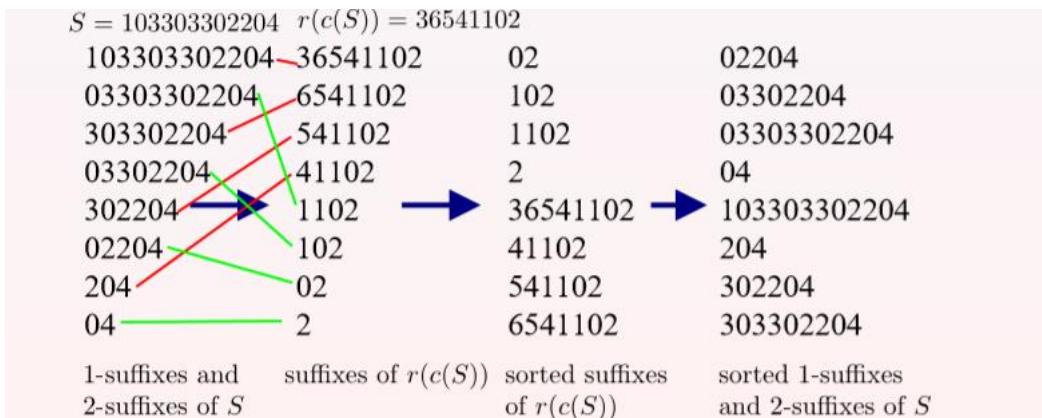
- $c(a,b,c) = a^*n^2 + b^*n + c [= "abc"]$
- $c(S) = c(S_1, S_2, S_3) \dots c(S_{n-2}, S_{n-1}, S_n)$

$S=103303302204$ $n=12$

$c(S)=147 \boxed{435} \ 434 \ 292 \ 39 \ 39 \ 26 \ 48$

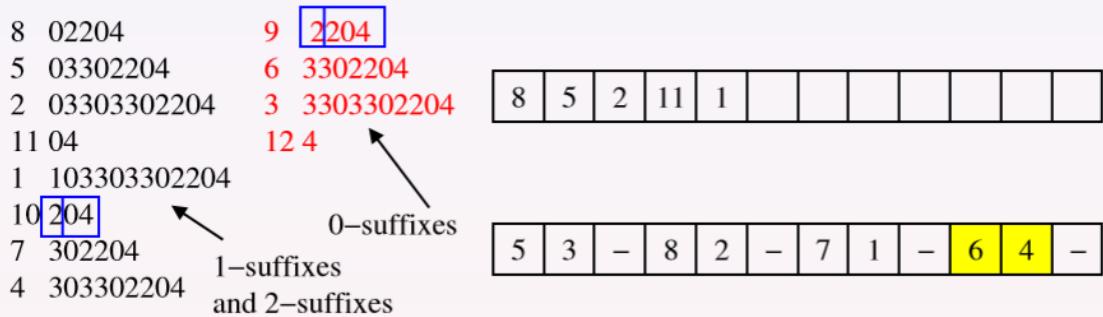
Sorting the 1-suffixes and 2-suffixes

נמיין את כל הסופיות לפי שיטת $(c(S))$



.*radix sort* היא סיפת 0-suffix-1 עם שרשור של אותו, لكن נוכל למין את 0-suffixsort בזמן לינארי ע"י בכל שלב נשווה את המחרוזות בראשית כל רשיימה.

- At each step, compare the strings at the top of the two lists.



- To compare a 1-suffix T_1 with a 0-suffix T_2 :
 - $T_1 = a_1 S_1$ with $a_1 \in \Sigma$, and S_1 is a 2-suffix.
 - $T_2 = a_2 S_2$ with $a_2 \in \Sigma$, and S_2 is a 1-suffix.
- The comparison of T_1 and T_2 takes $\Theta(1)$ time. We use an array that stores for each 1-suffix and 2-suffix its location in the sorted order of the 1-suffixes and 2-suffixes.
- To compare a 2-suffix T_1 with a 0-suffix T_2 .
 - $T_1 = a_1 b_1 S_1$ with $a_1, b_1 \in \Sigma$, and S_1 is a 1-suffix.
 - $T_2 = a_2 b_2 S_2$ with $a_2, b_2 \in \Sigma$, and S_2 is a 2-suffix.

השוואה בין 1-suffix-2 ל-2-suffix-1 נעשית בזמן קבוע ע"י מערך המכיל את הסופיות בסדר המומין של הסופיות בהתאם. (כפי שתיארנו לעיל ע"י $((S/c)/r)$ עברון)

ניתוח זמן ריצה:

Let $T(n)$ be the time complexity of sorting the suffixes of a string of length n over $\Sigma = \{0, \dots, n-1\}$.

- The recursive call takes $T(\frac{2}{3}n)$ time.
- The rest of the steps take $\Theta(n)$ time.
- We have $T(n) = T(\frac{2}{3}n) + \Theta(n) \implies T(n) = \Theta(n)$.

נגיד $T(n)$ סיבוכיות מין מחוזות באורך n מעל א"ב $\{0, \dots, n-1\}$. ישן ($\frac{2}{3}$) $T(n)$ קוריאות רקורסיביות עבור סיפות 2,1,2, שאר השלבים לוקחים זמן לינארי. לכן נקבל $O(n) = O(n) = O(n)$

i=1...n - מערך שבתא ה-i מכיל את גודל ה-LCP של התא ה-i והטא ה-(i+1)...(1) עבור -(n-i).

Example

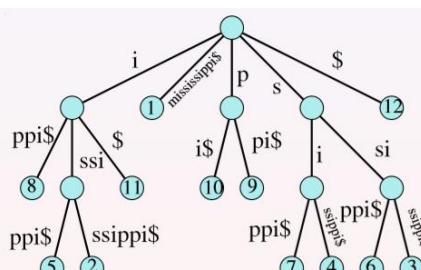
1	mississippi\$	8	ippi\$
2	ississippi\$	5	issippi\$
3	ssissippi\$	2	ississippi\$
4	sissippi\$	11	i\$
5	issippi\$	1	mississippi\$
6	ssi\$	10	pi\$
7	sippi\$	9	ppi\$
8	ippi\$	7	sippi\$
9	ppi\$	4	sissippi\$
10	pi\$	6	ssippi\$
11	i\$	3	ssissippi\$
12	\$	12	\$

The suffix array: [8 | 5 | 2 | 11 | 1 | 10 | 9 | 7 | 4 | 6 | 3 | 12]

LCP array: [1 | 4 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 3 | 0]

האלגוריתם לעץ יכול להיות שונה לייצרת מערך LCP ללא שינוי בזמן הריצה. ניתן לייצר מערך LCP מערך suffix array ב- $\Theta(n)$.

בاهינתן עץ סיפות נבנה ב- $\Theta(n^2)$ מערך סיפות וסימני העלים



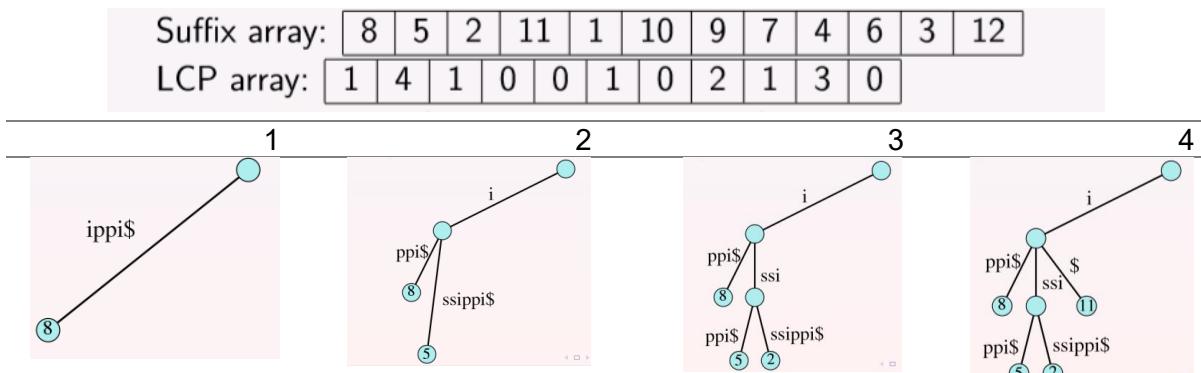
Suffix array: [8 | 5 | 2 | 11 | 1 | 10 | 9 | 7 | 4 | 6 | 3 | 12]

LCP array: [1 | 4 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 3 | 0]

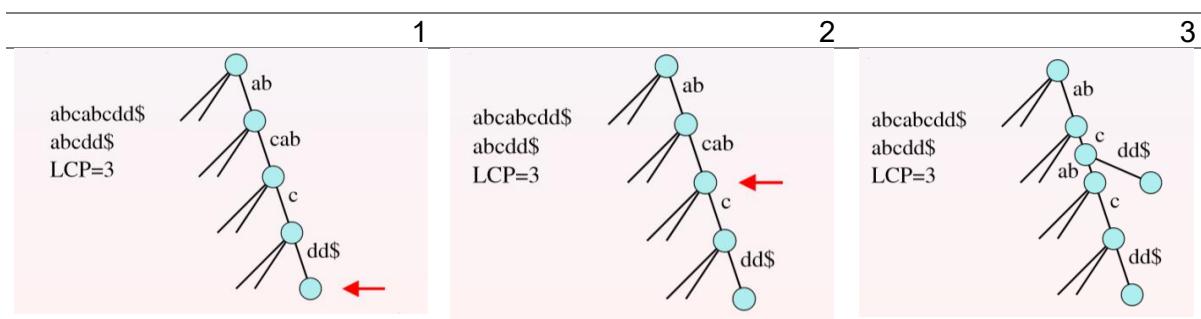
אלגוריתמים למחוזות / סיכום מאות אורי שביט

1. נסיף את הסיפות לעץ לפי סדר לקסיקוגרפי

2. נעזר במערך *LCP* למציאת המיקום שבו נשם עליה חדש



בכל איטרציה החל מהעליה האחרון שהוכנס עלה למעלה כל עוד העומק גדול יותר מב *LCP* של הסופית החדשה והסופית הבאה. נעבור על כל קשת בדיק פעם אחת :



לסיכום : עבור מחוזת *S* באורך *n* מעל א"ב $\{1 - n, \dots, 0\}$

1. ניתן לבנות מערך הסיפות וכן עץ סיפות $\Theta(n)$

- 3. בניית מערך *suffix array* ע"י: סורקים את העץ ושמרים את ערכי העלים לפי סדר ההגעה.
- 4. בניית *suffix tree* ע"י: נסיף את הסיפות לפי סדר לקסיקוגרפי בעדרת ה-*suffix array* ונמצא את המיקום להוספה העלה בעדרת ה-*LCP array* - נתחיל מהעליה האחרון שנוסף, ונעלם כל עוד העומק גדול מה-*LCP* של הסיפה הנוכחית והקדמתה.

הרצאה 5 - Succinct Data Structures

אנו מוחשים מבנה נתונים סטטי אשר יש לו סיבוכיות מקום אופטימלית, הסיבוכיות נמדדת ב- $bits$. נגדיר OPT הוא המינימום מקום הנדרש לאחסן המידע, מבנה הנתונים נקרא:

- (OPT=n) - מבנה השומר את המידע ב-(1)OPT+O מקום (نمץ ב- $bits \leq n$).
- Succinct d.s. - מבנה השומר את המידע ב-(OPT+o) מקום. o - גדל לאט
- Compact d.s. - מבנה השומר את המידע ב-(OPT) O מקום.

מבחן מחוץ S מארון מחוץ A rank-select-access data structure

- rank_c(S,i) - מספר המופיעים של c ברישא בגודל i של S.
- select_c(S,i) - האינדקס של המופיע ה-i של c ב-S.
- access(S,i) - האות במקום ה-i של S.

נניח כי המחרוזת S מופיעה בייצוג בינארי, למשל:

$$S = 010010001$$

$$rank_1(S,6) = 2, select_1(S,2) = 5$$

בבעה זאת האופטימום הוא 7 כיון שצורך ביטים לשימרת המחרוזת.

המטרה היא למשם עם זמן קבוע עבור פעולה ומינימום מקום. נתחיל בפתרון נאייבי :

אם שומרים את התשובות לכל השאלות פתרנו את הבעיה, מערך שבכל תא תשובה לשאלת $rank$ – n תאים, בכל תא מספר שייצגו בגודל $log n$ ולכן $log n$ ביטים לשימרה.

הרעין תמורה מאד RMQ – קלונר חלוקה לבlocks בגודל מסוים: ניקח את המחרוזת ונחלק לבlocks בגודל 7, בדוגמא 3. בונים 2 מערכות A – מכיל שורה לכל מחוץ אפשרית בגודל 7. בכל שורה יש תשובה לשאלת $rank$ עבור בלוק. לכל מחוץ נשמר את כל תשבות $rank$.

Store arrays A and B :

$$A[S', i] = rank_1(S', i) \text{ for every } S' \text{ of length } L$$

$$B[i] = \#\text{1's in blocks } 1, \dots, i - 1$$

S	0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1
---	---

A	000 0 0 0 001 0 0 1 010 0 1 1 011 0 1 2 100 1 1 1 101 1 1 2 110 1 2 2 111 1 2 3	B	0 1 3 4 7 9
			$L = 3$

מערך נוסף B – מכיל תא לכל בלוק קלומר 6 תאים ובכל תא מסתכלים על הבלוק המתאים ווסףים כמה תווים 1 יש משמאלי לכל בלוק. בלוק ראשון 0, בלוק שלישי 3 תווים אחד לפני

לענות על $-rank$

To answer a $rank_1(S, i)$ query, compute the vector S' of the block containing i , and return $B[\lceil i/L \rceil] + A[S', i \bmod L]$.
 (assume here that $i \bmod L = L$ if i is divisible by L)

			i=14																
S			<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1
0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1				
A	000	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>2</td><td>2</td><td>3</td><td>4</td><td>7</td><td>9</td></tr> </table>	0	0	0	0	0	1	1	1	1	2	2	3	4	7	9		
0	0	0	0	0	1	1	1	1	2	2	3	4	7	9					
001																			
010																			
011																			
100																			
101																			
110																			
111																			

$$\lceil i/L \rceil = 5$$

$$i \bmod L = 2$$

$$L = 3$$

נסתכל על הבלוק המתאים של כמות ואז נספור בתוך הבלוק עד שנגיע לשלונטי. אנו שומרים את S לפי מילוי זיכרון כלומר ניתן לגשת לביטים בקבוצות של 64 ביט. אנו רוצים בלוק בגודל לא בהכרח $3 \cdot 64 = 192$ ביטים $w = 8$

Computing S' is done by fetching one or two words of S (1 word = w bits) and performing constant number of operations (AND/SHIFT/OR).

A slightly more complicated case is when the bits of S' appear in two words.

			i=14																
S			<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1
0	1	0	0	1	1	1	0	0	1	1	1	1	0	1	1				
			w bits																

נשלוף w ביטים רלוונטיים ואז נעשה *shift* ע"מ להגעה לבלוק הרלוונטי. למשל נשלוף – 01111010 ואז /

נקבל 01111010 וنعשה *and* עם 00001111 כדי לקבל את הבלוק הרלוונטי. ניתן:

זמן השאלה – קבוע, *shift*, נבחר $2^{\log n}$.

מערך B – מספר התאים היא תא עבור כל בלוק כולם L/n ולכון \log ביטים עבור כל תא עבור ספירת מספר האחדות, $\log(n)$. ולכן הזיכרון הוא (n) ביטים עם זמן שאלתא קבוע, יותר טוב מהזיכרון הקבוע.

Choose $L = \frac{1}{2} \log n$. Recall that

$$A[S', i] = \text{rank}_1(S', i) \text{ for every } S' \text{ of length } L$$

$$B[i] = \#\text{1's in blocks } 1, \dots, i - 1$$

Query time: $O(1)$.

Space complexity: $O(n)$ bits.

- Array A : $2^L L \cdot \log L = O(\sqrt{n} \log n \log \log n)$
(there are $2^L L$ entries in the table, each taking $\log L$ bits).
- Array B : $n/L \cdot \log n = O(n)$
(there are n/L entries in the array, each taking $\log n$ bits)

נחלק את המחרוזת S לבLOCKים ב-2 רמות: בלוקים ומאקרו בלוקים L ו- L' . חלוקה לבLOCKים כמפורט ובלוקים בגודל 9, 3 מעריצים A – כמקודם, B – דומה לקודם אך כעת שומר לכל תא בערך בלוק, כמו אחדים יש משMAL לבלוק באותו מאקרו בלוק – כלומר הבלוק החמישי נמצא במאקרו בלוק השני. כמו כן מערך C המכיל תא עבור כל מאקרו בלוק ולכל תא כמה אחידות יש משMAL למאקרו בלוק.

Partition S into **blocks** of size L , and to **macro-blocks** of size L' .

Store arrays A, B, C :

$A[j, i] = \text{rank}_1(S_j, i)$, where S_j is the j -th string of length L

$B[i] = \#\text{1's in the blocks before block } i \text{ that are}$
in the same macro-block of block i

$C[i] = \#\text{1's in macro-blocks } 1, \dots, i - 1$

S	0 1 0 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1
---	---

A	<table border="1" style="border-collapse: collapse; width: 100px;"> <tr><td>000</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>001</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>010</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>011</td><td>0</td><td>1</td><td>2</td></tr> <tr><td>100</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>101</td><td>1</td><td>1</td><td>2</td></tr> <tr><td>110</td><td>1</td><td>2</td><td>2</td></tr> <tr><td>111</td><td>1</td><td>2</td><td>3</td></tr> </table>	000	0	0	0	001	0	0	1	010	0	1	1	011	0	1	2	100	1	1	1	101	1	1	2	110	1	2	2	111	1	2	3	B	0 1 3 0 3 5
000	0	0	0																																
001	0	0	1																																
010	0	1	1																																
011	0	1	2																																
100	1	1	1																																
101	1	1	2																																
110	1	2	2																																
111	1	2	3																																
		C	0 4																																

$$L = 3$$

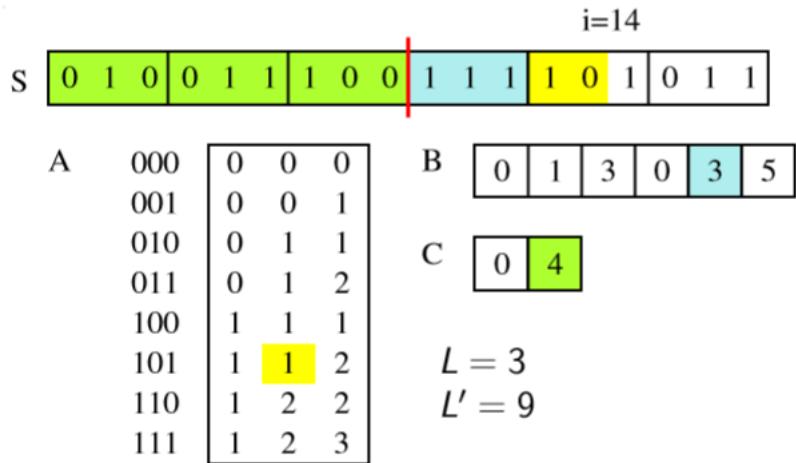
$$L' = 9$$

מענה על שאלות:

1. מאקרו בלוקים שבתוור השאלתית
2. בלוקים שלמים בתוור המאקרו בלוק
3. חילוק בבלוק הבא

נספור בנפרד כמה אחידים יש בכל החלקים, ראשית ניגש למערך C ואז למערך B ואז למערך A

To answer a $\text{rank}_1(S, i)$ query, compute the vector S' of the block containing i , and return
 $C[\lceil i/L' \rceil] + B[\lceil i/L \rceil] + A[S', i \bmod L]$.
(recall that $i \bmod t = t$ if i is divisible by L)



קודם כל מקרו בлок לפני הבלוק הנוכחי – מערך C

הmacro בлок הנוכחי – B

בתוך הבלוק המקורי ועד השאלתא – מערך A

ניתוח:

Choose $L = \frac{1}{2} \log n$, $L' = \log^2 n$. Recall that

- $A[j, i] = \text{rank}_1(S_j, i)$, where S_j is the j -th string of length L
- $B[i] = \#1's$ in the blocks before block i that are
 - in the same macro-block of block i
- $C[i] = \#1's$ in macro-blocks $1, \dots, i - 1$

Query time: $O(1)$.

נבחר את גודל הבלוק $2/\log n = L$ וגודל מקרו $n^2 = L' \cdot \log n$ גודל מקרו B יתפוא פחות ביטים כיון

שהמספרה יותר קטנה. כל איבר ב- B כעת הוא מספר בין 0 ל- $n^2 = L'^2 = \log^2 n$

C – מספר התאים L'/n בכל אחד $\log n$ ביטים לייצוג מספר. סה"כ

Space complexity: $n + O(n \log \log n / \log n) = n + o(n)$ bits.

- Array A : $2^L L \cdot \log L = O(\sqrt{n} \log n \log \log n)$.
- Array B : $n/L \cdot \log L' = O(n \log \log n / \log n)$.
- Array C : $n/L' \cdot \log n = O(n / \log n)$.

עבור שאלתא של 0 : $rank_0(S, i) = i - rank_1(S, i)$

שאלתא *select* : מוחירה את המיקום של ה-1 ה-

נחלק לבlokים כאשר בכל בלוק יש t אחדים:

To answer a $select_1(S, i)$ query, return

$$B[\lceil i/t \rceil] + A[\lceil i/t \rceil, i \bmod t].$$

(recall that $i \bmod t = t$ if i is divisible by t)

$i=8$																									
S	<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse; width: fit-content;"> <tr> <td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td> </tr> </table>	0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1	1	0						
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1	1	0								
A	<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse; width: fit-content;"> <tr> <td>2</td><td>5</td><td>6</td><td>7</td><td></td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>2</td><td>3</td><td>5</td><td></td><td></td><td></td><td></td> </tr> <tr> <td>1</td><td>2</td><td>-</td><td>-</td><td></td><td></td><td></td><td></td> </tr> </table>	2	5	6	7					1	2	3	5					1	2	-	-				
2	5	6	7																						
1	2	3	5																						
1	2	-	-																						
B	<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse; width: fit-content;"> <tr> <td>0</td><td>9</td><td>15</td> </tr> </table>	0	9	15																					
0	9	15																							

$t = 4$

A – בכל שורה שומרים את האינדקסים של 1 בכל בלוק, מיקום בתוך הבלוק

B – כמה תווים יש משאל לבlok.

ניקח $\frac{index}{t}$ – התשובה בבלוק המספר שקיבלנו.

לכן האחד שאנו מחפשים הוא האחד הרביעי בבלוק שלו כי אנו יודעים איך לחשב כמה היו לפניו.

מערך A ניכל לקבל את האינדקס היחסי של ה-1 הראשון ולן נוכל לחבר את B בבלוק הרלוונטי ולקבל $9 + \frac{5}{4}$ שהוא התשובה שказינו.

הבעיה היא שהמקום הוא *login* – נניח שמספר האחדים הוא כפולה של t ולן יש n תאים המערך A וכל מספר שאנו צריכים לשמר יכול להיות גדול. עבור המערך B $\frac{n}{t} * \log n$

Recall that each block contains t ones, and

$$A[i, j] = \text{select}_1(S', j) \text{ where } S' \text{ is the } i\text{th block}$$

$$B[i] = \#\text{bits in blocks } 1, \dots, i-1$$

Query time: $O(1)$.

Space complexity: $O(n \log n)$ bits.

- Array A: $n \cdot \log n$
- Array B: $n/t \cdot \log n$

כעת נראה פתרון מורכב יותר – נחלק את S למקרו בבלוקים בגודל $n/t = \log^2 n$ וכן גודלים $n = L'$ ונקבל 3 מערכיהם: A, B, B' . *short, long*.

A – דומה למה שהיא, ההבדל הוא שלא שומרים שורה עבהר כל מקרו בлок אלא רק עbor הארכבים.

B – אותו דבר כמו מה שהיא

B' – לכל מאקרו בлок שומר את האינדקס אם הוא מאקרו בлок ארוך. גם עבור קוצר נשמר 0

- Partition S into **macro-blocks** each containing $t' = \log^2 n$ ones.
- A macro-block is **long** if it has at least $L' = \log^4 n$ bits.
- Store arrays A, B, B'

$A[i, j] = \text{select}_1(S', j)$ where S' is the i th long macro-block

$B[i] = \text{length of macro-blocks } 1, \dots, i - 1$

$B'[i] = j$ if macro-block i is the j th long-macro-block

- We need structures for handling queries on short macro-blocks.

S	0 1 0 0 1 1 1 0 0	1 1 1 0 1 0	1 1 0
---	-------------------	-------------	-------

A	2 5 6 7	B	0 9 15	B'	1 2 0
	1 2 3 5		$t' = 4, L' = 6$		

עבור $i=8$, נחפש אינדקס מקודם $\frac{\text{index}}{t}$, אם B' יש מספר יוכל לעשות מקודם B ואז A . לא יהי הרבה ארוכים ולכן לא נשמר הרבה זיכרון.

$A[i, j] = \text{select}_1(S', j)$ where S' is the i th long macro-block

$B[i] = \#\text{bits in macro-blocks } 1, \dots, i - 1$

Space complexity:

- Array A : $\frac{n}{L'} t' \cdot \log n = O(n/\log n)$
(there are at most n/L' long macro-blocks).
- Array B : $\frac{n}{t'} \log n = O(n/\log n)$
(there are at most n/t' macro-blocks).

חישוב זהה למקודם כל כמה בлокים ארוכים. אנו רוצים לקבל שיטה לטיפול במאקרו בлок קצר

Select — short macro-blocks

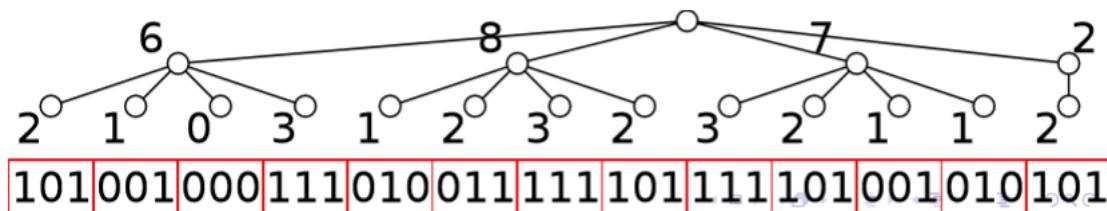
For each short macro-block S' :

- Partition S' into **blocks** of length $L = \frac{1}{2} \log n$.
- Build a complete tree with degree $d = 2\sqrt{\log n}$ whose leaves correspond to the blocks. Since the number of blocks is at most $L'/L \leq 2 \log^3 n$, the height of the tree is ≤ 6 .
- For each node v in the tree, assign a value equal to the number of ones in the blocks corresponding to the descendant leaves of v .

בاهינתן אינדקס לקבל את האינדקס החמישי בתוכו.

נחלק אותו לבLOCKים בגודל קבוע, כל BLOCK יכול 7 ביטים כאשר $L=3$, כל 3 ביטים הם בלוק.

cutת נבנה עצ' שהעלים שלו מתאים לBLOCK, לחבר את העלים באופן הבא – לכל צומת יהי בדיק d בניים (4 בדוגמא), נמשך וכך d צמתים מרמה 2 ו לחבר אותם.



את הערך d נבחר להיות $2\sqrt{\log n}$

$$\frac{L'}{L} = \frac{\log^4 n}{\frac{1}{2} \log n} = 2 \log^3 n - n$$

$$\text{רמה } 2 \text{ מספר העלים} > \frac{2 \log^3 n}{d^i} \text{ וכך הלאה ברמה } i$$

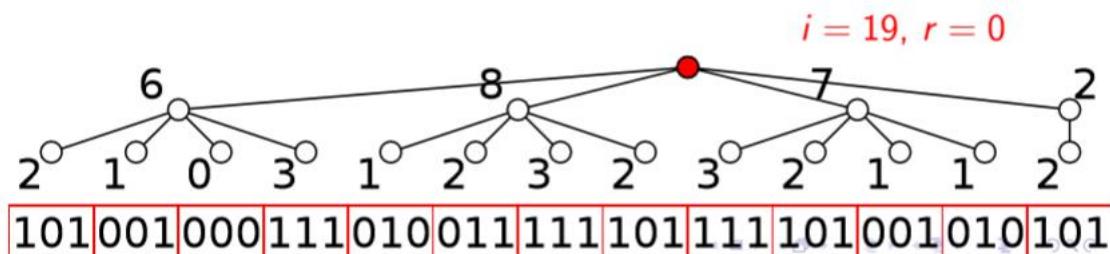
$$\text{לכן מרמה } 6 - \frac{2 \log^3 n}{d^6} \text{ וכך הלאה ברמה } 6 \text{ הוא } 6.$$

בכל צומת בעץ נשמר את כמות האחדים מתחת לצומת, תת צמות היא סכימה של הבלוקים התחתיו.

שימוש בז select :

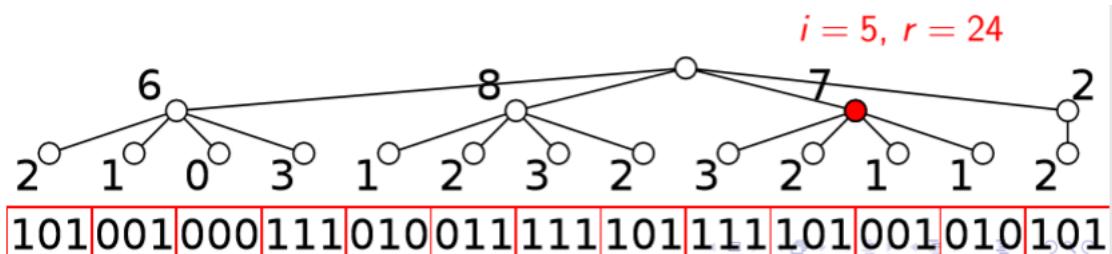
To compute $\text{select}_1(S', i)$, let $v = \text{root of the tree}$.

- ➊ Let v be the root of the tree. $r \leftarrow 0$.
- ➋ for $l = 1, \dots, 6$ do: (we assume the height of the tree = 6)
 - ➌ Let v_1, \dots, v_{d^l} be children of v , and let k_1, \dots, k_{d^l} be their values.
 - ➍ Find minimum j such that $\sum_{p=1}^j k_p \geq i$.
 - ➎ $i \leftarrow i - \sum_{p=1}^{j-1} k_p$. $r \leftarrow r + (j-1)Ld^{4-l}$. $v \leftarrow v_j$,
 - ➏ return $r + \text{select}_1(S'', i)$, where S'' is the block of v .

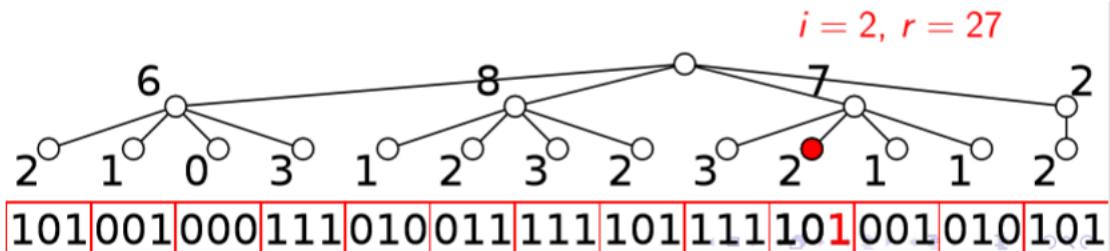


המיקום של ה-1 ה-19 בתחום המאקרו בлок. עושים זאת ע"י ירידה בעז לעלה המתאים וחישוב בתחום הבלוק.

נסתכל במספרים הרשומים בצלמתים ברמה מתחת, נסכים עד שנגיע למספר מעל או שווה לאינדקס, $7+7+6, 6+8, 6+8+8$. לכן נרד לצומת 7, כעת נעדכן את האינדקס להיות $i = 5 = 14 - 19$ ע"מ למצוא את האינדקס בתחום הבלוק. נוסיף את מספר הביטים בקטע שהתעלמו ממנו – ככמויות הבלוקים כפול כמות הביטים בכל בלוק. לכל צומת בדיקת d בניים.



לכן קיבל $24 = 3 * 8$ אינדקסים לפני הירידה ברמה.



cutet נרד שוב באותו אוף יש לנו אינדקס 5 ונוסף צומתים עד שנייגע למספר $<= 5$, נרד לתוך העץ ונוסיף כמו אינדקסים שדילגנו $= 24 + 3 = 27$. נעדכן את האינדקס $i = 2 = 3 - 5 = 2$ – ולכן נחפש בתחום הבלוק עכשו משתמש בטבלה של התשובות האפשריות וכל טבלה שומרה את התשובות האפשריות – מיקומי האחדים כמקודם.

- To answer select queries on blocks, store array $C[S'', i] = \text{select}_1(S'', i)$.

000	-	-	-
001	3	-	-
010	2	-	-
011	2	3	-
100	1	-	-
101	1	3	-
110	1	2	-
111	1	2	3

התשובה הסופית תהיה $3 + 27 = 30$

$$\log(d) = \log(2\sqrt{\log n})$$

בاهינתן סדרת בנים של צומת וערך יעד צריך למצאו את הבן המתאים. נשמר מערך עזר לטובת כר. מערך $D_{S'}$ יכול את הערכים של הצמתים בעץ S' הצמתים מסודרים לפי רמה מלמעלה למטה ומשמאלי לימין בכל רמה.

מערך E אשר

מערך F אשר מכיל את סכום k_i על לכל נקודה.

In order to descend the tree in $O(1)$ time, store:

- An array $D_{S'}$ that contains the values of the nodes in the tree of S' . The nodes are ordered by level from top to bottom, and in left to right order in each level.

D 6 8 7 2 2 1 0 3 1 2 3 2 3 2 1 1 2

- Array E in which $E[k_1, \dots, k_d, i] = \min\{j : \sum_{p=1}^j k_p \geq i\}$.
- Array F in which $F[k_1, \dots, k_d, i] = \sum_{p=1}^i k_p$.

3,2,1,1 E 1 1 1 2 2 3 4 - - F 3 5 6 7 3 5 6 8 $i = 5, r = 24$

מערך $D_{S'}$ דורש $\log t'$ ביטים כיון ש $dlogt' = O(\sqrt{\log n} \log \log n)$ כאשר ערכי d של כלILD של צומת מתאים למילה.

ניתוח: $C[S'', i] = \text{select}_1(S'', i)$ כאשר $L = \log^2 n, t' = \log^4 n, d = 2\sqrt{\log n}$ לכל S' .
מערך $L, t' = D_{S'}$ מאריך הצמתים של עצ עbor.

- Array C : $2^L L \cdot \log L = O(\sqrt{n} \log n \log \log n)$
(there are $2^L L$ entries in the array, each taking $\log L$ bits).
- Arrays $D_{S'}$: $n/L \cdot \log t' = O(n \log \log n / \log n)$
(there are $\leq n/L$ entries in all arrays, each taking $\lceil \log t' \rceil$ bits).

- Array E : $2^{(d+1)\lceil \log t' \rceil} \log d = 2^{O(\sqrt{\log n} \cdot \log \log n)} = o(\sqrt{n})$
(there are $2^{(d+1)\lceil \log t' \rceil}$ entries, each taking $\log d$ bits).
- Array F : $2^{d\lceil \log t' \rceil} d \log t' = 2^{O(\sqrt{\log n} \cdot \log \log n)} = o(\sqrt{n})$
(there are $2^{d\lceil \log t' \rceil} d$ entries, each taking $\log t'$ bits).

- בהינתן מחרוזת S מעל $\Sigma = \{1, \dots, \sigma\}$, מגדירים $\Sigma_L = \Sigma \setminus \Sigma_R = \{1, \dots, \lceil \sigma/2 \rceil\}$, ואז מmirים כל אות ב- S מ- Σ_L ל-0 ומשמר את זה בשורש. תת-העץ השמאלי "בנה באופן רקורסיבי מתוך תת-המחרוזות שמתקבלת ע"י הסרת כל האותיות שב- Σ_R מ- S . עליה נוצר כאשר כל המחרוזות בינויו מסוג אחד שלתו.

מאתהן מחרוזת S מעל א"ב $[1, 2, \dots, \sigma]$ ותומך באופרציות הבאות:

- $\text{rank}_c(S, i)$ - מספר המופיעים של c ברישא בגודל i של S .
- $\text{select}_c(S, i)$ - האינדקס של המופיע ה- i - של c ב- S .
- $\text{access}(S, i)$ - האות במקומות ה- i של S .

למשל : $\sigma = 4, S = 213224132$

$$\text{rank}_2(S, 6) = 3$$

$$\Sigma_R = \Sigma \setminus \Sigma_L = \{1, \dots, \lceil \sigma/2 \rceil\}$$

ב- S היא מחרוזת מאורך ch כאשר $1 = B[i] \in \Sigma_R$ אם $S[i] \in \Sigma_L$.

$S = 45633451242$: לדוגמה, בהתאם ל- S_L, S_R

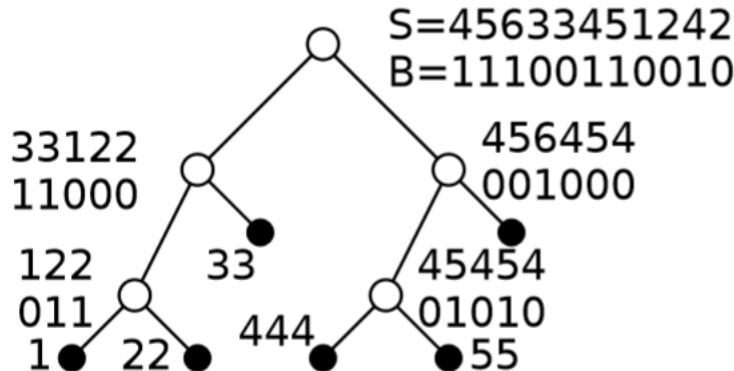
$B = 11100110010, \Sigma_L = \{1, 2, 3\}, \Sigma_R = \{4, 5, 6\}$.

$SL = 33122, SR = 456454$

בנייה עץ בינארי עבור S :

1. שורש מכיל את $\text{rank}/\text{select}$ מבנה הנתונים עבור B

2. הילד השמאלי והימני הם עצים הנבנו בצורה רקורסיבית עבור S_L, S_R



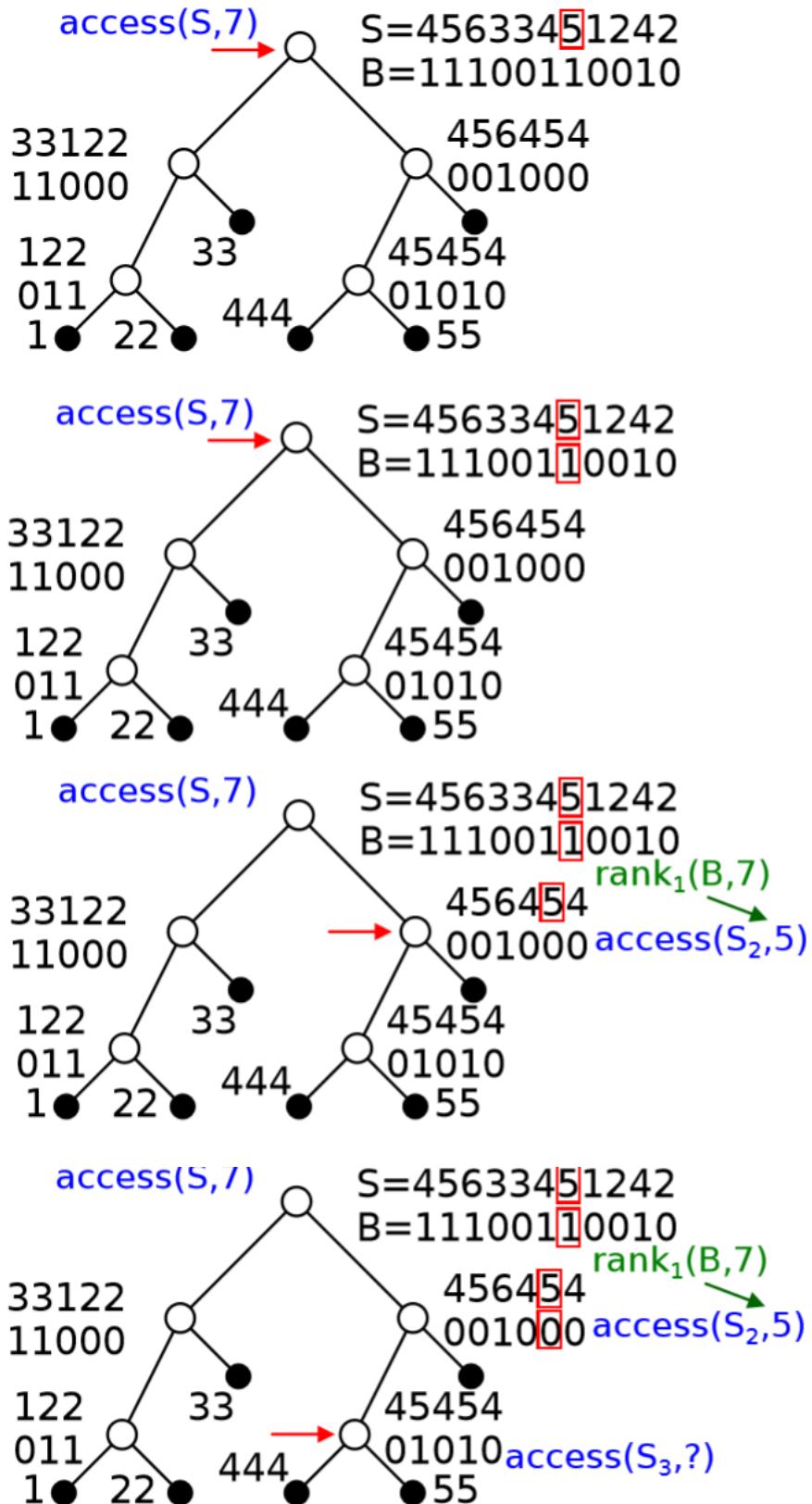
נראה איך מבצעים פעולות $\text{select}, \text{rank}, \text{access}$ במבנה זה.

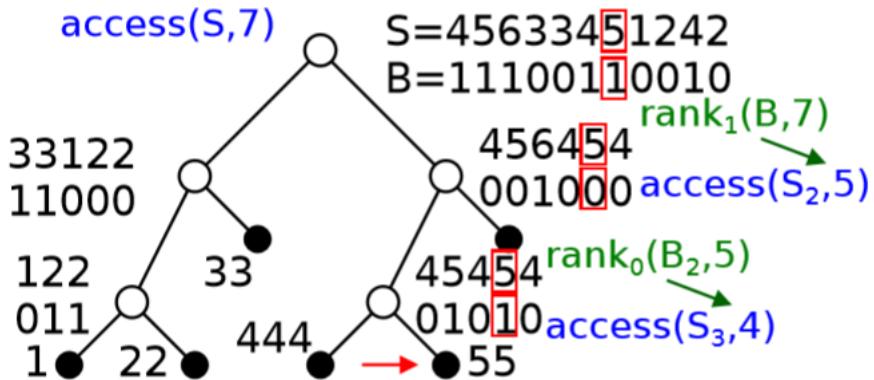
(i) $\text{access}(S, i)$ - האות במקומות ה- i של S : שקף 39 – 47

נתאר את $\text{access}(S, 7)$ בדוגמה

נתחיל מהשורש:

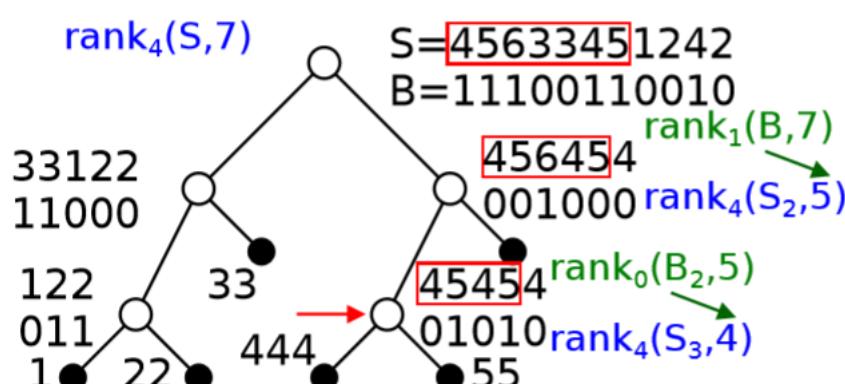
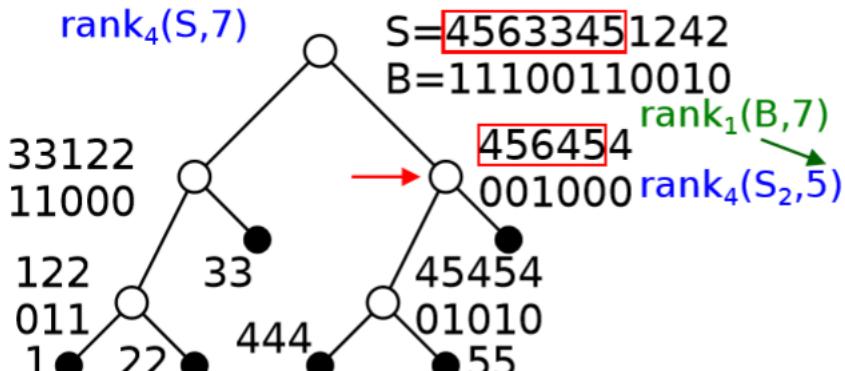
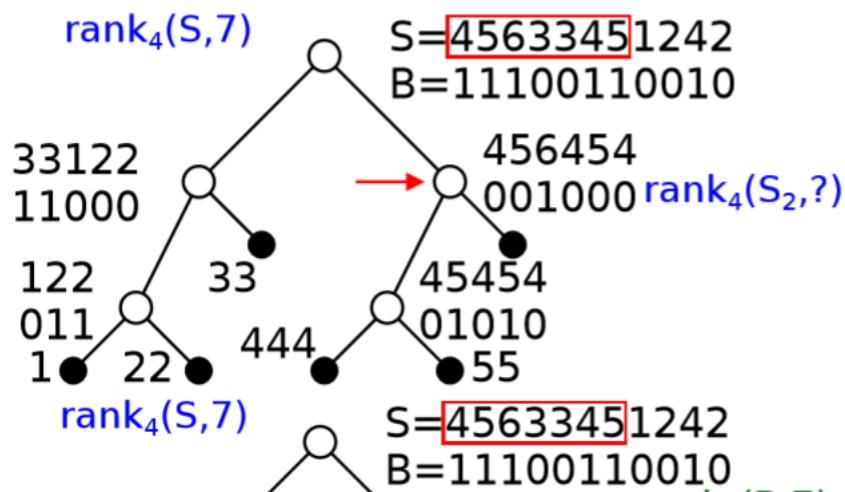
נלר' הילד השמאלי/ימני לפי ה-[i] ב- B_v ומעדכנים את $i \rightarrow (i - \text{rank}_0(B_v, i))$. נמשיך רקורסיבית עד אשר נגיע לעלה. נחזיר את מספר העלה.

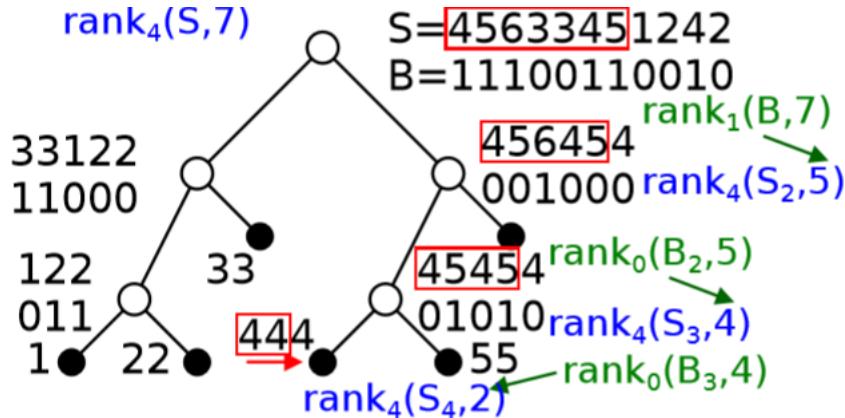




52- 48 : שקף 48 בדוגמא $rank_4(S, 7)$
נתאר את

נתחיל מהשורש:
נלק לילד השמאלי/ימני לפי c ומעדכנים את $i \rightarrow i$ or $rank_1(B_v, i) \rightarrow i$ or $rank_0(B_v, i) \rightarrow i$, נמשיר רקורסיבית עד
שנגיע לעלה, נחזר את i .

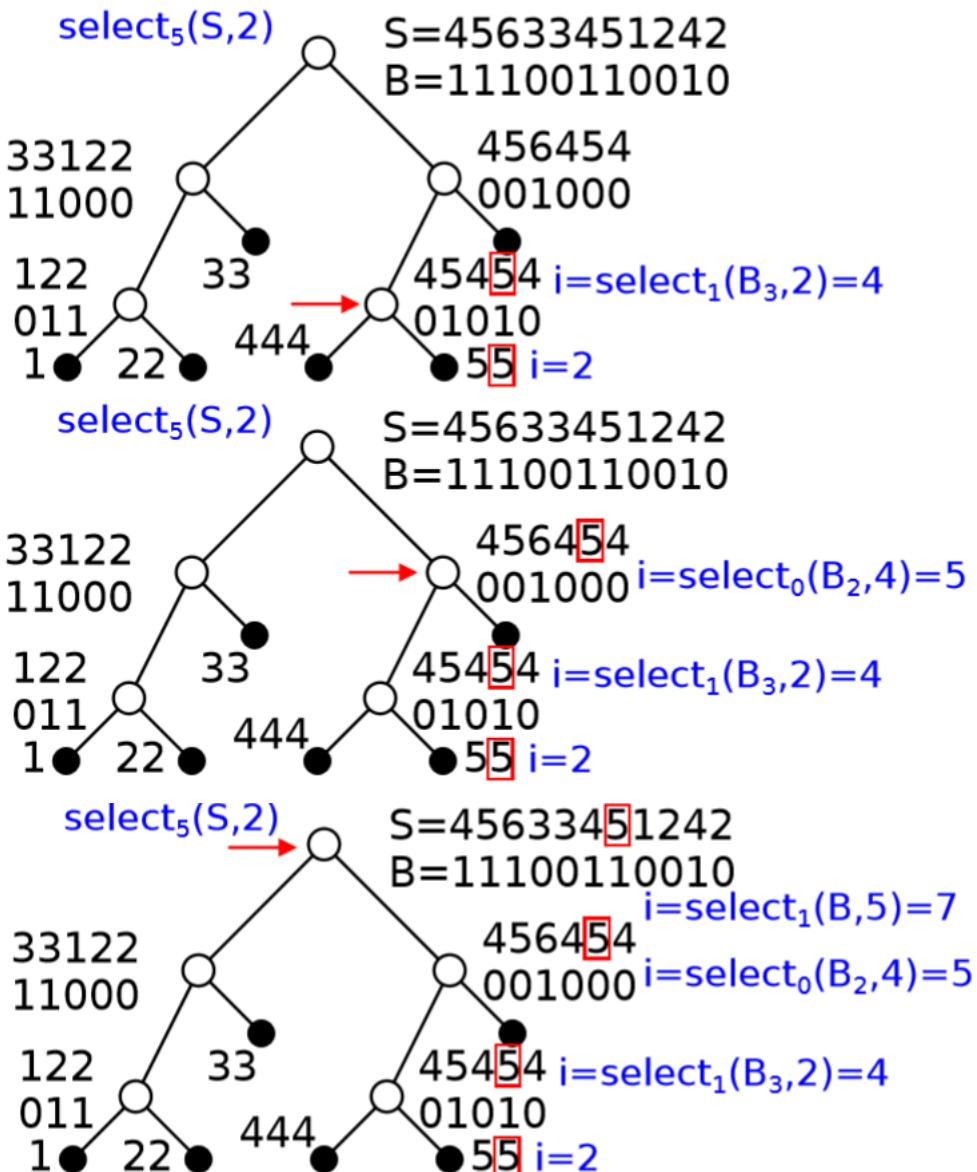




: $select_c(S, i)$ - האינדקס של המופיע ה- i של c ב- S

נתאר בדוגמא את $select_5(S, 2)$

נתחיל בעלה c ונעלמה למעלה בעץ. בכל שלב נגדיר $i \rightarrow select_c(B_v, i) \rightarrow i$ or $select_1(B_v, i) \rightarrow i$ כאשר נגיע לשורש נחזיר את i .



סיבוכיות מקומ:

הוקטור B עבור השורש מכיל n ביטים

סכום האורכים עבור כל וקטורי B_v החול מרמה מסוימת $\geq u$.

כמו הרמות היא $\log \sigma$

המקום של וקטורי $B_v = \log \sigma$

המקום עבור מבני *rank-select* עבור וקטורי B_v הוא $O(n \log \sigma)$

המקום עבור מצביעי העץ היא $O(\sigma \log n)$: לעז יש $1 - \sigma$ צמתים, כל מצביע יכול להיות מאוחסן ב($\log n$) θ ביטים לאחר שכל המבנה יכול להיות מאוחסן ב($\log n$) θ ביטים.

סה"כ המקום הדרושים : $n \log \sigma + O(\sigma \log n) + O(\log \sigma)$

לסיכום: עבור $\Sigma = \sigma$ נוכל לבנות *A rank-select-access data structure* עם $O(n \log \sigma + n \log \sigma + O(\sigma \log n))$ מקום כאשר כל שאלת זמן קבועה.

עבור כל σ אנו יכולים לבנות *A rank-select-access data structure* עם $O(n \log \sigma + O(\sigma \log n) + O(\log \sigma))$ מקום כאשר כל שאלת זמן $\log \sigma$ θ .

הרצאה 6 – FM-index

Locate - מבנה עבור השאלות הבאות על מחוזת T : Count - מספר המופיעים של מחוזת P ב-T , Extract - עבור [$i, i+1, \dots, i+k$] תווים החל מהתו ה- i).

– כל המופיעים של P ב-T, Extract – עבור [$i, i+1, \dots, i+k$] תווים החל מהתו ה- i).

– מכיל את כל הסיפות של מחוזת T מופיעות בסדר לקסיקוגרפי(הנחה – T מסת'ם ב\$) – Suffix array

1	alabar_a_la_alabarda\$	21	\$
2	labar_a_la_alabarda\$	7	_a_la_alabarda\$
3	abar_a_la_alabarda\$	12	_alabarda\$
4	bar_a_la_alabarda\$	9	_la_alabarda\$
5	ar_a_la_alabarda\$	20	a\$
6	r_a_la_alabarda\$	11	a_alabarda\$
7	_a_la_alabarda\$	8	a_la_alabarda\$
8	a_la_alabarda\$	3	abar_a_la_alabarda\$
9	_la_alabarda\$	15	abarda\$
10	la_alabarda\$	1	alabar_a_la_alabarda\$
11	a_alabarda\$	13	alabarda\$
12	_alabarda\$	5	ar_a_la_alabarda\$
13	alabarda\$	17	arda\$
14	labarda\$	4	bar_a_la_alabarda\$
15	abarda\$	16	barda\$
16	barda\$	19	da\$
17	arda\$	10	la_alabarda\$
18	rda\$	2	labar_a_la_alabarda\$
19	da\$	14	labarda\$
20	a\$	6	r_a_la_alabarda\$
21	\$	18	rda\$

suffix array – בהינתן suffix range של מחוזת P הוא אינטראול האינדקסים ב-Suffix range המתחילה ב-P .

Suffix range of 'la' is [17,19]

17	10	la_alabarda\$
18	2	labar_a_la_alabarda\$
19	14	labarda\$

$$m = |T|, |P| = n$$

בහינתן suffix array, אנו יכולים לחשב את suffix-range של P בזמן $O(m\log n)$. בעזרה מערך LCP ניתן לשפר את זמן הריצה ל $O(m + \log n)$.

suffix array משתמש בכמות גדולה של זיכרון: $n \log_2 n$ ביטים למערך, $\sigma \log n$ ביטים למין T וזכרון למערך LCP . (במקרים רבים $n \log n$ גדול בהרבה מאשר $|P|$).

Burrows-Wheeler transform (BWT)

BWT (Burrows-Wheeler transform) – בהינתן מחוזת T, BWT של T זו מחוזת כך שהאות במקומות ה- i במחוזת היא האות שלפני הסיפה ה- i , כאשר $i=1 \dots |T|$ (\$).

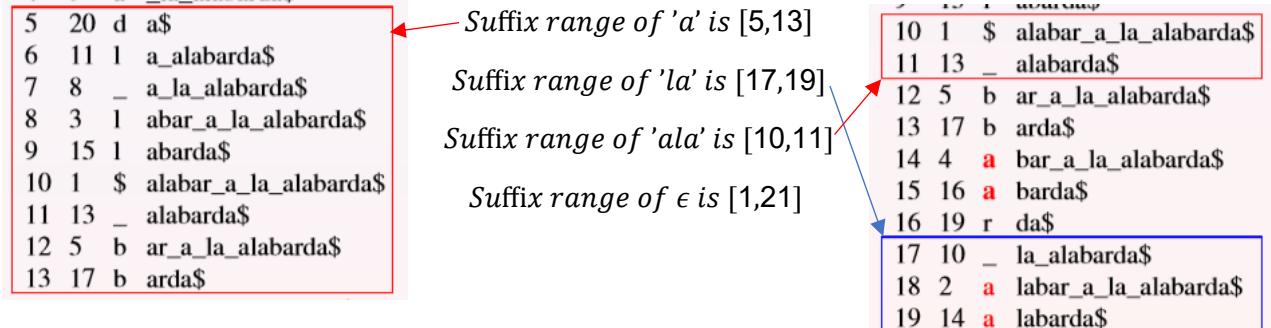
$$BWT[i] = T[SA[i] - 1] \text{ (if } SA[i] = 1, BWT[i] = T[n]).$$

1	21	a	\$
2	7	r	_a_la_alabarda\$
3	12	a	_alabarda\$
4	9	a	_la_alabarda\$
5	20	d	a\$
6	11	l	a_alabarda\$
7	8	_	a_la_alabarda\$
8	3	l	abar_a_la_alabarda\$
9	15	l	abarda\$
10	1	\$	alabar_a_la_alabarda\$
11	13	_	alabarda\$
12	5	b	ar_a_la_alabarda\$
13	17	b	arda\$
14	4	a	bar_a_la_alabarda\$
15	16	a	barda\$
16	19	r	da\$
17	10	_	la_alabarda\$
18	2	a	labar_a_la_alabarda\$
19	14	a	labarda\$
20	6	a	r_a_la_alabarda\$
21	18	a	rda\$

אלגוריתמים למחוזות / סיכום מאת אורי שביט

בהינתן מחוזת P אנו רוצים לחשב את ה *suffix range of 'ala' is [10,11]* (לעיל)

האלגוריתם מחשב את ה *suffix-ranges* של כל הסופיות של P בסדר עולה לפי אורך. נציג כי *heaps* עבור המילה הוריקה ϵ הוא טווח כל המילים - [1,21]. כיוון שכל מילה יכולה גם להתחיל במילה הוריקה.



נניח שמצאנו *suffix-range* עבור S בתחום $[r, l]$ וכן אנו רוצים למצוא את הטווח עבור S $[l', r']$.

הסופיות של T שמתחלות ב $S\alpha$ מתאימות לאינדקסים $[l, r] \in i$ עבור $\alpha = S[i]$. כאמור עבור α הסופיות של T שמתחלות ב $S\alpha$ מוגדרות באמצעות ראיון המתחילה ב $S\alpha$ אם:

סופה של R מטנה יותר מסופית ראשונה המתחילה ב $S\alpha$ אם:

1. מתחילה באות הקטנה מ α או

2. מתחילה ב α ואחריה אותן קטנה יותר בסדר לקסיקוגרפי M .

לדוגמא: $S = la, \alpha = a$

יש 4 סופיות מהסוג הראשון ו 5 מהסוג השני ולכן $C[\alpha] = 9$.

היא C מערך בגודל $|S|$, $C[\alpha]$ שווה למספר התווים ב T אשר קטנים מ α .

(i) – מספר k שבו $\alpha = S[i, k]$ – $rank_{\alpha}(X, i)$

אם $[r, l]$ הוא *suffix-range* עבור S ו $[l', r']$ הוא *suffix-range* עבור αS אז:

$$l' = C[\alpha] + rank_{\alpha}(BWT, l - 1) + 1$$

$$r' = C[\alpha] + rank_{\alpha}(BWT, r)$$

ניתן לשמר את BWT ביעילות ואת i – $rank_{\alpha}(BWT, i)$ למשל ע"י *wavelet tree*.

סיבוכיות זמן: $t_{rank} = \Theta(\log \sigma)$ – σ count query $= \Theta(m \cdot t_{rank})$

סיבוכיות מקום: אנו צריכים לשמר: מערך C – $\log \sigma$ ביטים. מבנה עבור BWT שתומך בחישוב $i[BWT, l - 1, r - \sigma]$.

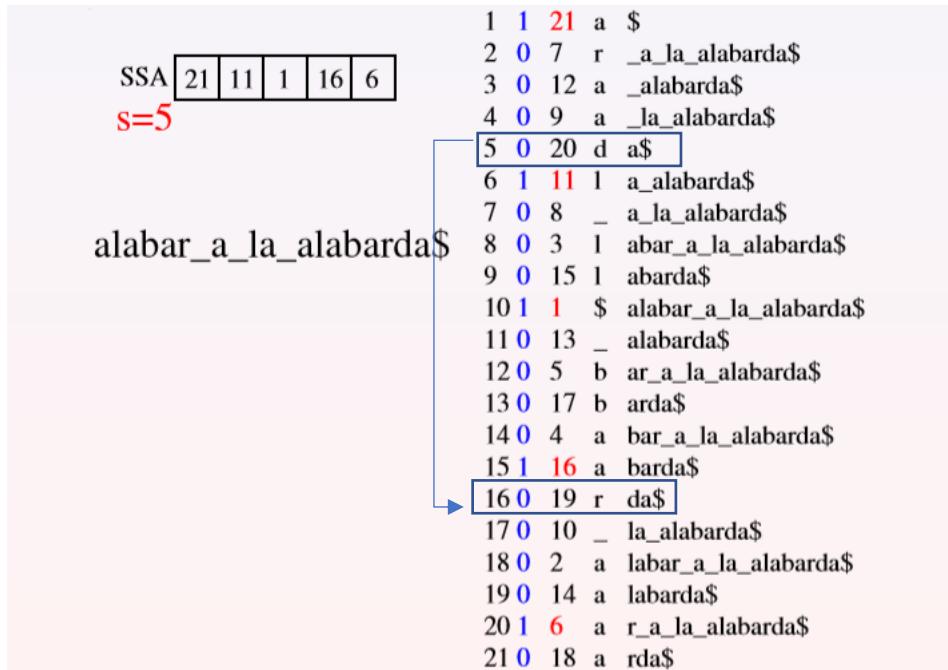
– ע"מ לתמוך בשאלתה זו אנו צריכים להיות מסוגלים לחשב $i[SA]$ עבור אינדקס i קלשא. *Locate queries*

אנו יכולים לשמר $array - suffix array$ – $array$ Ark כרוך בשמירת מידע רב. פתרון: נשמר רק את חלקיים *suffix-array* SSA (sampled suffix array) – בהינתן SSA וקבוע q , ערכי התאים ב- SSA הם ערכי $q \bmod 1$ ממוניים לפידם ב- $suffix array$.

– מערך שבתא ה- i שלו יש 1 אם ב- $array$ SSA במקומ ה- i יש ערך שנמצא ב- SSA ו-0 אחרת.

השאלה היא איך נחשב $?SA[i]$

אלגוריתמים למחוזות / סיכום מאת אורי שביט



עבור i , אם P זו המחרוזת שנמצאת במקום ה- i - ב- P (last first function) אז $LF(i)$ הוא מיקום המחרוזת P כאשר a זו האות שלפני P . $LF(5) = 16$

чисוב $LF(j)$ הוא מקרה פרטי של חיפוש לאחרור כלומר $\alpha = BWT[j]$

$$\begin{aligned} \text{If } mark[i] = 1 \text{ then } SA[i] = SSA[rank_1(mark, i)] \\ SA[15] = SSA[4] = 15 \end{aligned}$$

$$\begin{aligned} \text{If } mark[i] = 0, SA[i] = SA[LF(i)] + 1 \\ SA[21] = SA[13] + 1 \end{aligned}$$

Thus, $SA[i] = SA[LF^j(i)] + j$, where $mark[LF^j(i)] = 1$.

$$SA[21] = SA[13] + 1 = SA[15] + 2 = 18$$

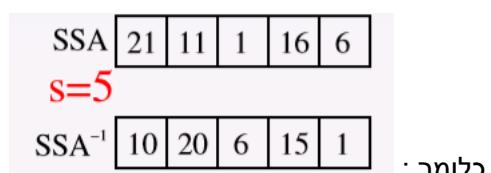
alabar_a_la_alabarda\$


8 0 3 1 abar_a_la_alabarda\$
9 0 15 1 abarda\$
10 1 1 \$ alabar_a_la_alabarda\$
11 0 13 _ alabarda\$
12 0 5 b ar_a_la_alabarda\$
13 0 17 b arda\$
14 0 4 a bar_a_la_alabarda\$
15 1 16 a barda\$
16 0 19 r da\$
17 0 10 _ la_alabarda\$
18 0 2 a labar_a_la_alabarda\$
19 0 14 a labarda\$
20 1 6 a r_a_la_alabarda\$
21 0 18 a rda\$

מערך SSA לוקח $\frac{n}{s} * logn$ ביטים.

זמן לחישוב $SA[i]$ הוא $\Theta(s \cdot (t_{rank} + t_{access}))$, אם משתמש בwavelet tree עבור BWT אז נקבל $t_{rank} + t_{access} = \omega(logn)$, $s = \log^{1+\epsilon}(n)$ אנו זוקים לנ' $t_{rank} + t_{access} = \Theta(log\sigma)$

- מערך עם המיקומים ב-SA של $s, s+1, s+2, \dots, 1, 1+s$ (המיקומים האדומים) לפי סדר זה (מיימן לשמאלי).



i	SA[i]	Suffix	
0	20	\$	
1	6	_a_la_alabarda\$	
2	11	_alabarda\$	
3	8	_la_alabarda\$	
4	19	a\$	
5	10	a_alabarda\$	
6	7	a_la_alabarda\$	
7	2	abar_a_la_alabarda\$	
8	14	abarda\$	
9	0	alabar_a_la_alabarda\$	
10	12	alabarda\$	
11	4	ar_a_la_alabarda\$	
12	16	arda\$	
13	3	bar_a_la_alabarda\$	

1	1	21	a	\$
2	0	7	r	_a_la_alabarda\$
3	0	12	a	_alabarda\$
4	0	9	a	_la_alabarda\$
5	0	20	d	a\$
6	1	11	l	a_alabarda\$
7	0	8	_	a_la_alabarda\$
8	0	3	1	abar_a_la_alabarda\$
9	0	15	1	abarda\$
10	1	1	\$	alabar_a_la_alabarda\$
11	0	13	_	alabarda\$
12	0	5	b	ar_a_la_alabarda\$
13	0	17	b	arda\$
14	0	4	a	bar_a_la_alabarda\$

<https://visualgo.net/en/suffixarray>

Extract queries

extract : נתחיל $\alpha = SSA^{-1}[ceil(r/s)+1] \dots \alpha = SSA^{-1}[ceil(r/s)]$, נחשב את i_j לכלי k , ונחזיר את $[r-(l-1) \dots r] * BWT[i_k] \dots BWT[i_1]$ (הכפל הוא שרשו).

ע"מ לחשב את $SA[i_1] \leq r + 1 \leq SA[i_2] \dots SA[i_l] = SSA^{-1}[[r/s] + 1 \dots r]$. נבדיר כי $s[r/s] = i_1$.

נרצה לחשב את $T[3..4]$ ולכן $20 = SSA^{-1}\left[\frac{4}{5} + 1\right] = SSA^{-1}[2]$ ולכן נקבל כי נתחיל מ' הראשון.

$alabar_a_la_alabarda\$$
↑
->

נחשב $1 + i_1 + i_2 + \dots + i_l = LF(i_j - 1)$ for $j = 2, \dots, k$, s.t $k = s[r/s] - l + 1$. התוצאה שתתקבל תהיה :
 $BWT[i_k] \cdot BWT[i_{k-1}] \dots BWT[i_{k-(l-r)}]$

לכן נחשב : $i_1 = 20, i_2 = 12, i_3 = 14$ ונתחיל לאוסף את האותיות הרלוונטיות למחוזת המחופשת :
 $T[3..4] = b, i_1 = 20, i_2 = 12, i_3 = 14$

$alabar_a_la_alabarda\$$ $alabar_a_la_alabarda\$$
↑↑ ↑↑
->

$T[3..4] = a \cdot b, i_1 = 20, i_2 = 12, i_3 = 14$

סיבוכיות מקום לסיכום :

1. מערך $C[\alpha]$ שווה למספר התווים ב- α אשר קטנים מ- σ ביטים

2. מבנה נתונים עבור BWT שתווסף בחישוב ($i, rank_\alpha(BWT, i)$)

Wavelet tree: $n[\log \sigma] + o(n \log \sigma) + O(\sigma \log n)$ bits

$\Theta((n/s)\log n) = o(n)$ bits : Arrays SSA and SSA^{-1} .

4. מבנה נתונים התומך בזמן קבוע : $mark, rank_1(mark, 1, i)$

סה"כ סיבוכיות מקום : $n[\log \sigma] + o(n \log \sigma) + O(\sigma \log n)$

סיבוכיות זמן:

נניח כי $m = |\mathcal{P}|$, y הוא גודל תת-המחוזות בשאלת `extract`.

הזמן t_{rank} והזמן t_{access} הם הזמן לחישוב $rank_\alpha(BWT, i)$ ו- $BWT[i]$ אזי

$\Theta(m \cdot t_{rank}) = \Theta(m \log \sigma)$ הינו `count` הינו $\Theta(m \log \sigma)$

2. הזמן לדיווח על הימצאות הוא $(n \log^{1+\epsilon} n)$

3. הזמן לשאלתה של `extract` הוא $\Theta((t_{rank} + t_{access}) \cdot (s + l))$

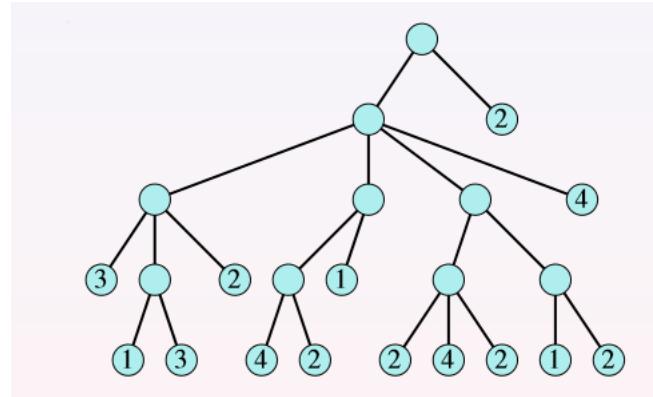
*שאילתה אחרונה

מצגת 7 – שיעור 7 - Document listing

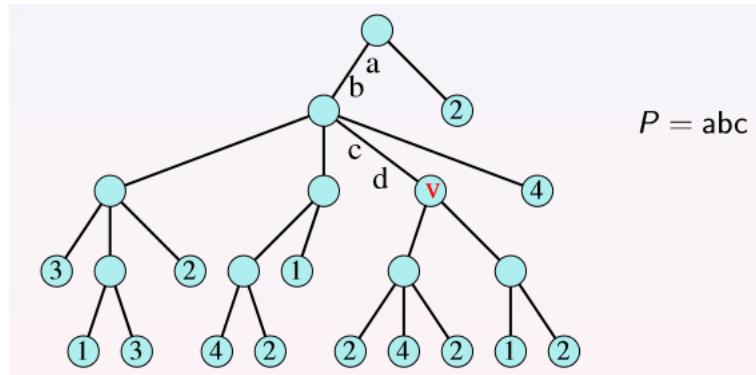
- Document listing d.s. – מבנה עבור קב' מסמכים {pd...pd} המחזיר את כל המסמכים המכילים מחוזות P. [יש גם מבנה כנ"ל עבור מציאת k המסמכים המכילים את P ה kali הרבה פעמים].

نبנה עץ סיפות מוכל שמכליל את כל המחרוזות שקיבלו, לכל מחוזת מוסיפים תו \$ ולן נוכל לבצע \$1 כאשר לכל מחוזת יהיה תו סיממת שונה.

אם נבנה עץ לדוגמה:



כעת אם יש מחוזת חיפוש abc נוכל לרדת בעט לפ' c ונדע עד שנגיע לע' locus – מקום אליו מגיעים אם יורדים לפ' מחוזת.



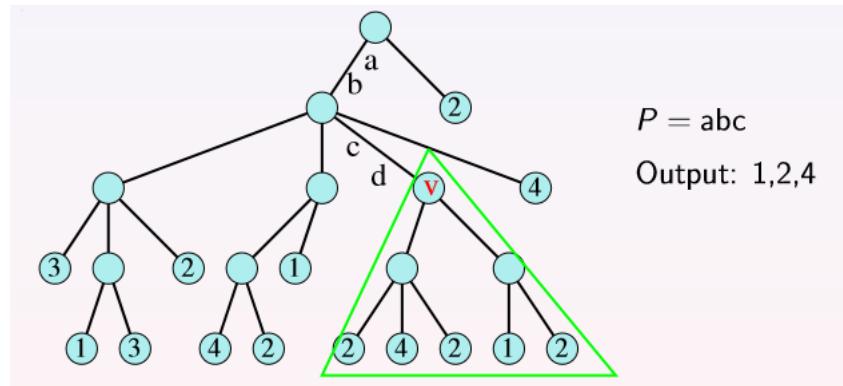
- Locus - בהינתן P , locus הוא הצומת הגבוה ביותר בו P זהה לתת-מחוזת של המחרוזת המיוצגת ע' המסלול מהשורש ל-' v .

אנו יודעים שהמחוזת מופיע ב T_1, T_2, T_4

הבעיה שאם נסרק אוטם

בכל מסך מועה מיליון פעמים ולכז אולי נצרך זמן גדול כדי לסרוק אותו כדי לדעת איפה מופיעות ויש מעת אינדקסים.

ניתן בכל צומת לשמר את התשובה כלומר לשמור את כל המספרים השונים המופיעים בתת העץ, בכל צומת פנימית נשמר את כל המספרים שופיעו בתת העץ. אם נסמן ב M את מספר המחזוזות השונות אז נצרך $O(D \log n)$. ואנו רוצים דירקט ($D \log n$) .



— נרצה לפטור בעיה על מערך A והשאילתה היא בהינתן 2 אינדקסים a, b נצרך להחזיר את המספרים המופיעים בין $[a, b]$.

- CRQ - בהינתן מערך A וקלט a, b , מבנה עבור השאלה מה קב' הערכים ב- $[a...b]$.

A	2	2	2	3	2	3	3	1
	a						b	

OUTPUT :2,3

כדי לפטור את בעיה זו נשתמש בטריק: גבנה מערך עזר C באשר C (מערך של מצביעים שמצביע על שרשרת ערכים עם אותו ערך). מסתכלים על המופיע הקודם של הטע $[i, j]$ ורושמים את האינדקס של המופיע הקודם.

A	2	2	2	3	2	3	3	1
C	0	1	2	0	3	4	6	0

איך עוזר מערך זה לבעיה?

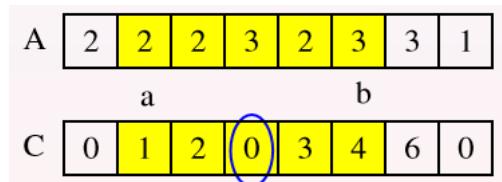
נרצה למצוא את המופיע הראשון של כל תו שונה: נשים לב שהמופיע הראשון או שיש ממשאל לו 0 ולכן נחפש את הערכים שקטנים ממנו. בדוגמה אצלונו יש 0, 1 וכאן נוכל לומר כמה שונים יש.

A	2	2	2	3	2	3	3	1
C	0	1	2	0	3	4	6	0

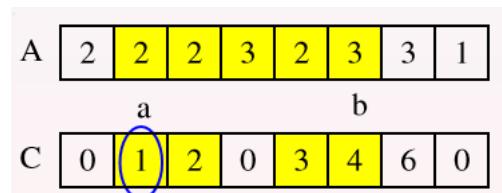
אלגוריתמים למחוזות / סיכום מאות אורי שביט

מווע לא ראשן זה אומר שהערך המתאים ב- C מציין לתא באזור הצעוב, יהי ערך גדול יותר מ- m ולכן בכל אחד מ- 3 התאים יש ערך גדול שווה מ- m

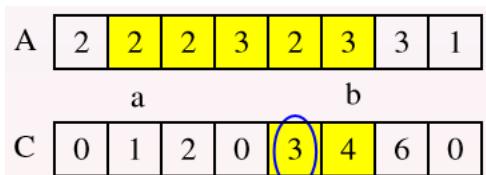
עדין צריך למצאו בצורה עילית את המספרים השונים מ- m בקטע הנ"ל: יוכל לפתור זאת באמצעות RMQ : למצאו מספר קטן ביותר. וכך יש לבצע אלגוריתם רקורסיבי, למצאו מספר קטן ביותר וכן וכך צריך לעשות להדפיים אותו, לא סימנו וכן מה שמשאל למספר ומה שמיון למספר.



לאחר מכן:



אם האיבר המינימלי גדול שווה ל m ולכן ניתן לעזור עבור המקטע כיון שכל האיברים גדולים שווים ל m .



זמן הריצה : שאלתא RMQ קבוע, השאלה כמה איטרציות יש לאלגוריתם: כל פעם הריצה מתפצלת ל 2, צומת שחור – הגענו למספר גדול m , כל פעם שנגיע לצומת לבן מצאנו מופיע חדש – מספר חדש שונה מהמספרים שהופיעו. כלומר מספר הלבנים שונה לעומת הקודסות בפלטו!

מספר הצמתים הלבנים – לכל היותר $|output|$

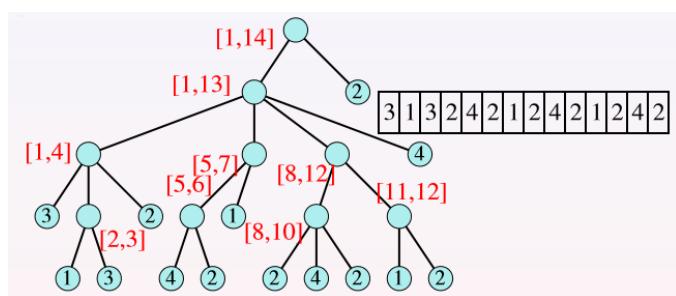
מספר הצמתים השחורים – לכל היותר |output|

ולכן מספר הצמתים הוא ($|output|$) O

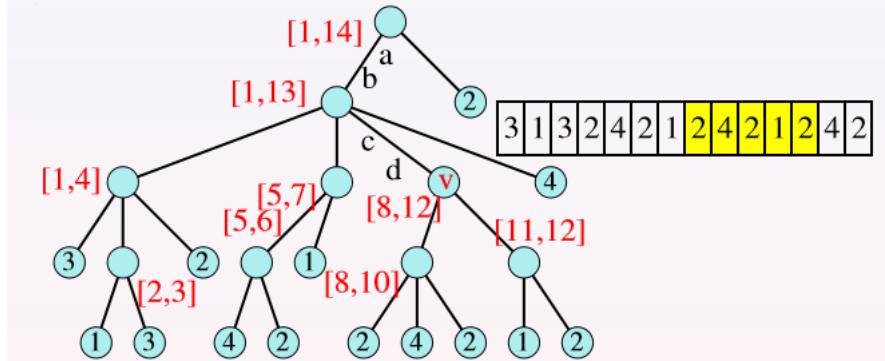
לכן זמן הריצה ($O(|output|)$)

הגענו למצות בעז ולכן נוכל להשתמש במבנה הנתונים מקודם לפתרון בעיה זו : ניצור מערך אחד שיכיל את המספרים של העלים לפי הסדר(משמאל לימין) רושמים את המספרים שנמצאים בעלים - E) ועבورو לבנייתם את ה

- CQR , בנוסף כל צומת פנימי יכול את הולה השמאלי והימני ביותר של (אב קדמון).



ירודים abc וממשיכים עד הצומת, רואים שהעלים המתאימים [12,8] על המערך והשאילתה תדפיס את כל המספרים השונים בקטע לפי CRQ .

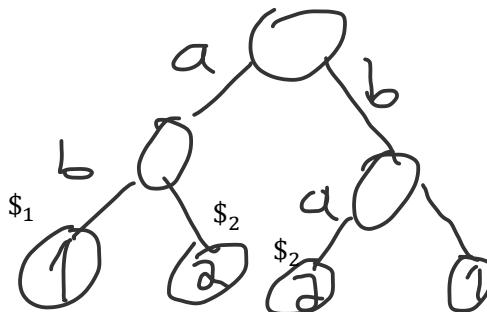


מקום של הפתרון – h סכום כדי המחרוזת, m – מספר המסמכים השונים המכילים את P .

מספר התאים במערך E – h ומבנה CRQ – n , لكن סה"כ סיבוכיות מקום (n) O .

סיבוכיות מקום - $O(m + n \cdot doc)$

דוגמה כתה - ab , bd – בונים עץ שמכיל את כל הסיפות, בכל עלה רושמים את تو המתאים לשיפא של המחרוזת. $d_1 = ab\$_1, d_2 = ba\$_2$



מנועי חיפושים:

The top-k document retrieval data structure

נרצה למצוא את המסמכים שהמחרוזת P מופיעה hei הרבה פעמים.

Example

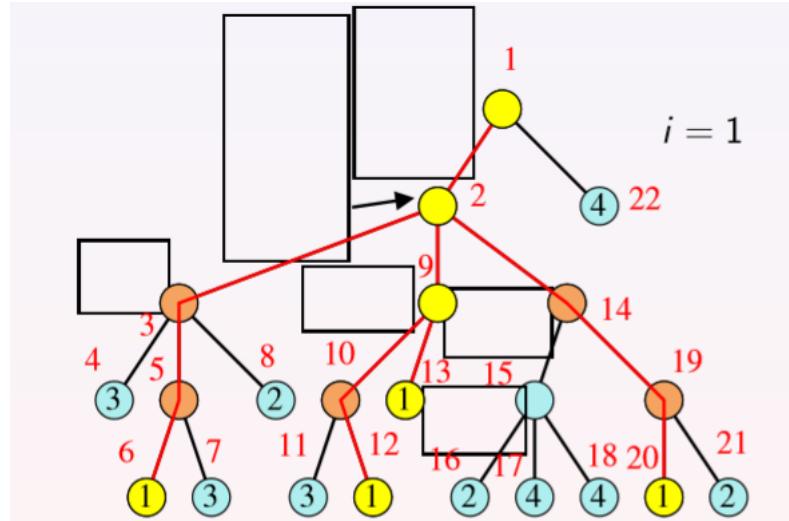
$d_1 = abab, d_2 = baba, d_3 = abcabc, d_4 = cbacba, d_5 = ccccc$.
For $P = ba, k = 2$, the output is d_2, d_4 .

נקבל k , כאשר נדפיס את המחרוזות בהן P מופיעה hei הרבה $-k$ המסמכים.

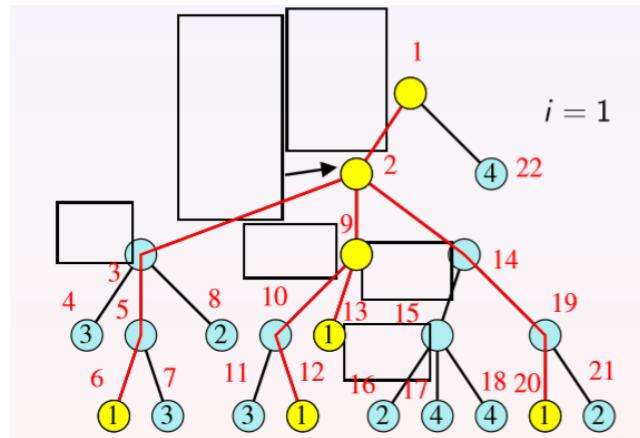
פתרון שוב מבוסס על עץ סיפות מוכפל – כל עלה בעץ מתאים לשיפא אחד ממחרוזת אחת בקבוצה, נctrך להוציא מידע בכל צומת פנימי – מערך.

נספר את הצמתים של העץ לפי *preorder*, בעת מעבור על המסמכים לפני הסדר. נטפל ב $t1$ בצהוב – עליים שמכילים 1 ואת כל הקדמוניים. הצמתים שצבענו יוצרים תת עץ.

רוצים ליציג תת עץ באורה קומפקטיבית, נכזב את הצמתים שלכל צומת יש רק בן בודד.

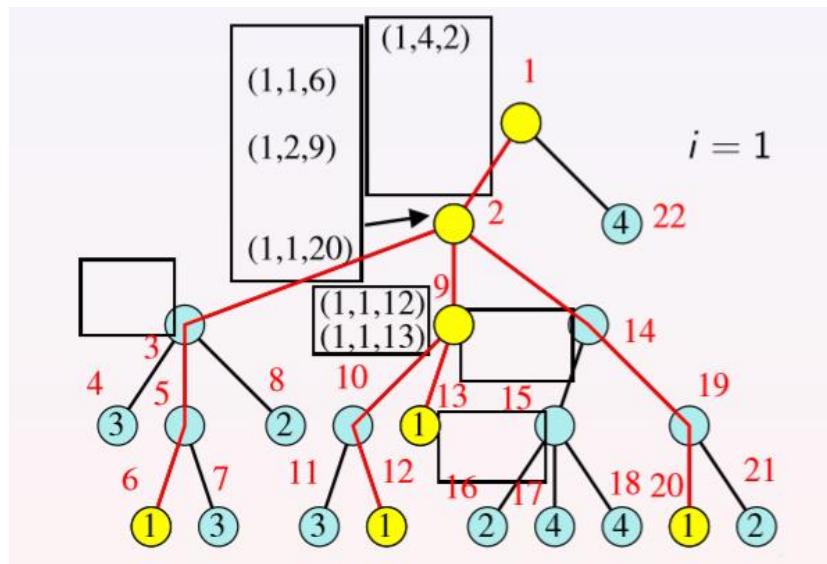


לעתה העץ המכusz נקרא T_1 בדיק עץ הסיפות של המחרוזת d , כאשר שומרים על התוויות ומשרשרים תווים לענף. קיבלנו את עץ הסיפות של d . תת עץ אדום.

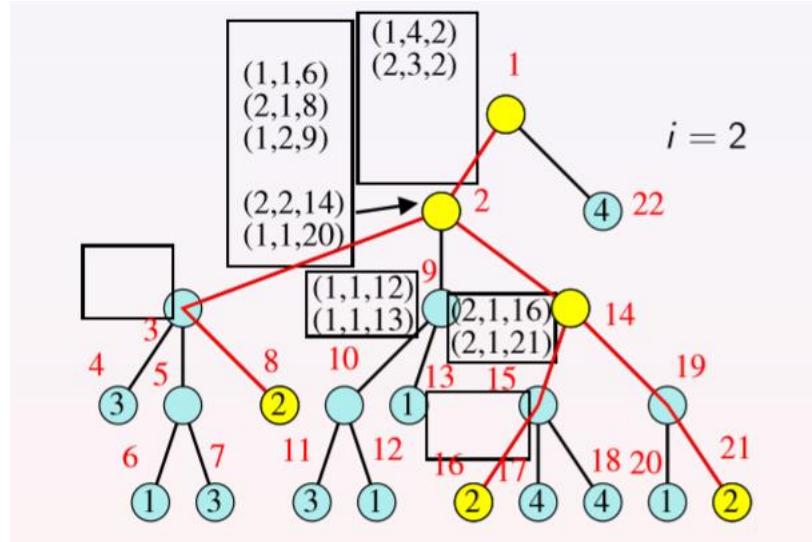


נעבור על פני כל הצמתים בתת עץ, כל צומת מסמן בא ונמן אב在他.

מוסיף ל i את השלשה – האינדקס i – מספר המסמן, האיבר האחרון בשלשה – מספר $preorder$ שכרגע מטופל, האיבר השני – מספר הפעמים שיש לנו 1 בתת העץ של הצומת $(i, \#occ(i, x), preorder(x))$.

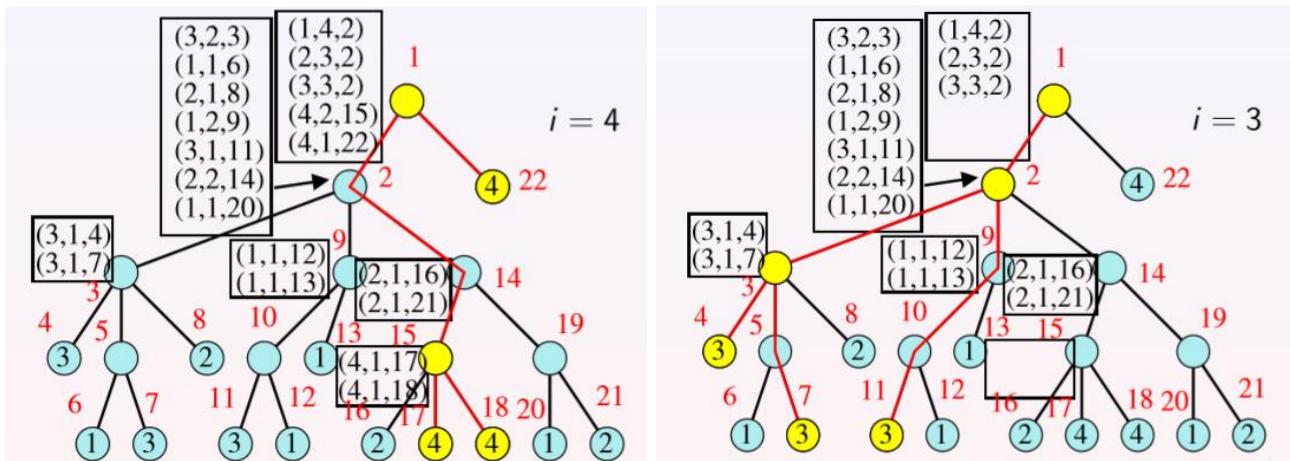


עושם את אותו דבר עבור $i = 2, 3, 4$.



עבור על כל הצמתים ונסתכל על האב ונוסף שלשה למערך של i , עבור i נוסף צומת 14 המתופלת :
האינדקס, 2 כמות לצאים שמספרם 2, 14 מיקום *preorder*.

אם נסתכל על $i=2$, האב $= 1$, ולכן 2 האינדקס, 3 כמות הצאים שהם 2 ו 21 מס' *preorder*.



כל מערך צזה נשמר ממוין ועל כל מערך צזה נעשה מערך *RMQ* ולפי הקואורדינטה השנייה, שייתן לנו את הקואורדינטה השנייה מקסימלית .

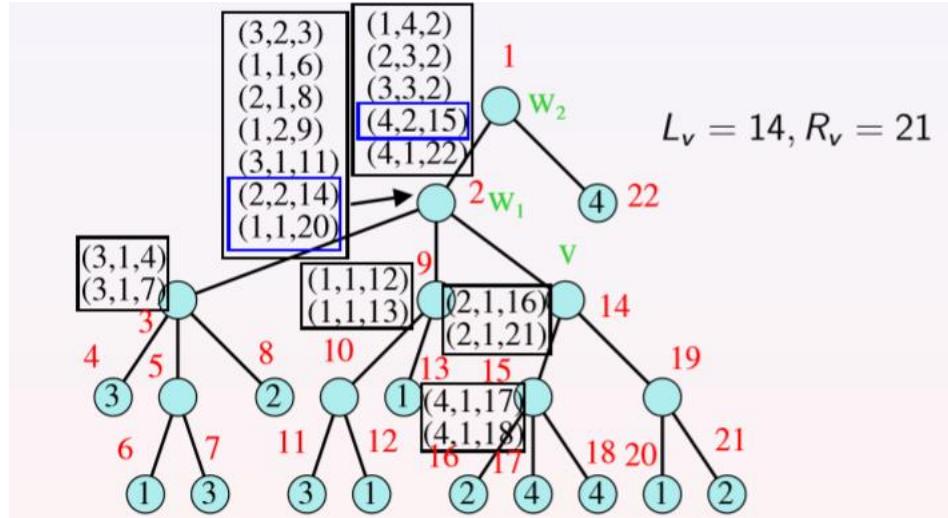
נמיין את המרכיבים לפי הראורדינטה השלישי.

ירודים בעז עד שמגיעים לצומת- נסמנה v , נסמן v *preorder num of*

R_v – *preorder(v0)where v0 is the rightmost descendant leaf of v*

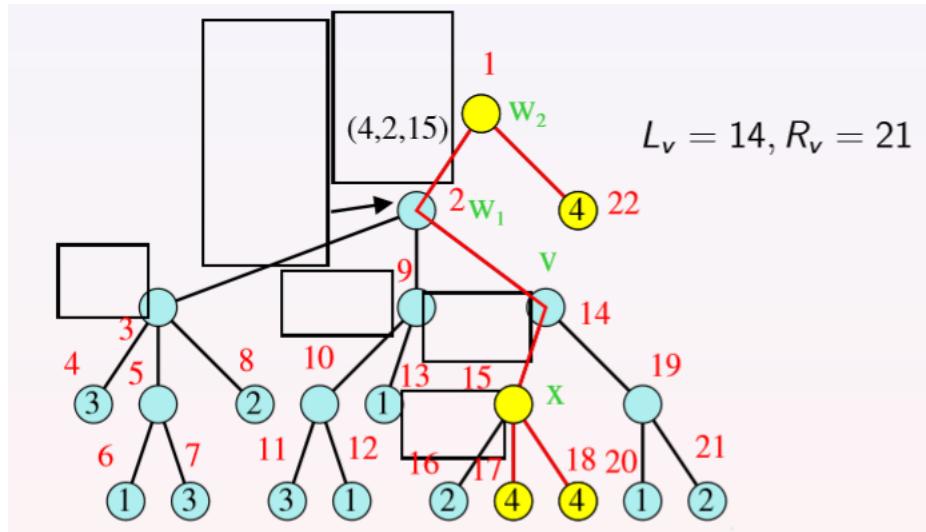
כל צומת שנמצא מתחת לעץ של v נמצא בין מס' הci גדול להci קטן.

נסתכל על האבות הקודמוניים של v – נסמנם Bw , נסתכל על המרכיבים של כל w בנפרד, מה שהאלגוריתם עושה מסתכל על השלשות שבין 21-14 . כיוון שהמערכות ממונעים איז שלשות מופיעות ברכף. אנו טוענים שהשלשות נוותניות את כל המרכיבים השונים שmorphisms בחתם העץ של v . יכולם קיבלנו בקואורדינטה הראשונה את מס' המרכיבים שבהם מופיעות המחרוזת. בקואורדינטה השנייה כמות המופיעים בכל מס' .
כלומר כעת יש לנו את המידע שהוא צריים ואני צריים להציג את K הci טובים.



איך המידע שקיבלנו זה מה שציריך?

למשל מסמך 4 :



יש לנו את תת העץ של 7 שבו מספר מופיעים של 4, כתע נסתכל על הצלמת הצהוב היכי גבוה בעץ ונקרה לו א.

לפי הגדירה כאשר בנו את המבנה ליקחנו את האב שלו וחיב להיות אב קדמון של v_i . ולכן כאשר ייצורנו את השלשה ייצורנו שלשה שנטפל בה בהכרח.

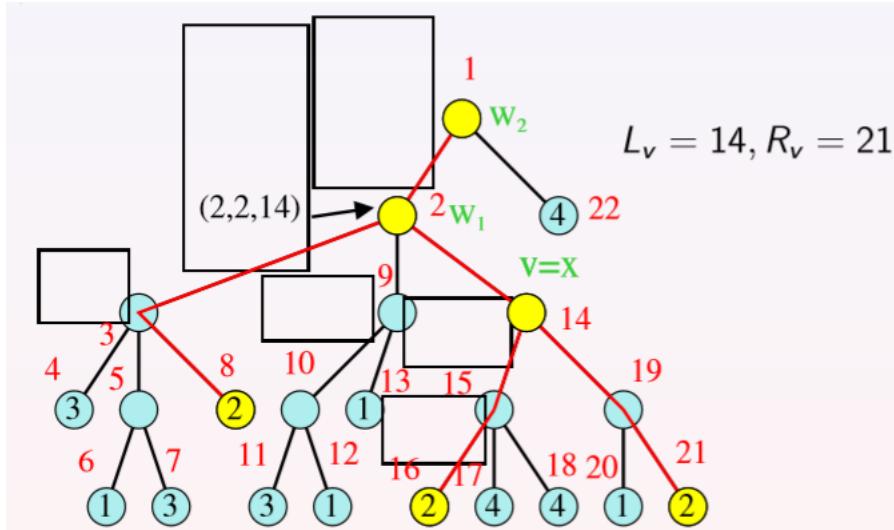
1. $v_i = x$ – האב הצהוב הוא בתוכו אב קדמון

2. $v_i \neq x$ – האב הצהוב יהיה אב קדמון של v_i

א. קודקוד הבא שהוא צאצא צהוב.

בהגדירה של המערכים, השלשה המתאימה למערך של האב, מספר מסמך , 15-סדר, מספר הצלצאים הצלובים. לכן כל מסמך שמופיע מתחת לעץ של v_i

אותה דוגמא עברו v_i .



לכן כל פעם צריך לעבור על השלשות ולמצוא את k השלשות שהקואורדינטות השניות מקסימליות.
נראה שכמעט פתרנו את הבעיה אך לא ביעילות, למשל k קטן מאוד, ומספר המופעים גדול ולכן לפחות מיליאן
כליה וצריך לבחור מתוך 3^k .

שוב נרצה להשתמש במבנה נתונים שייעזר לנו בעיה זו : RMQ !.

בנייה מערך אחד מכל המערכים: נשרשר אותם. כתת יש לנו מערך אחד ומספר מסוים של קטעים ואנו רצחים
למצא את k האברים הגדולים ביותר.

Let A be array. Given intervals $[a_1, b_1], \dots, [a_t, b_t]$ and k , we want to find the k largest elements in $A[a_1, b_1], \dots, A[a_t, b_t]$.

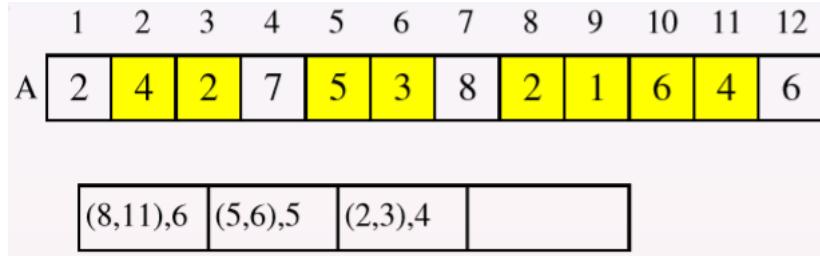
1	2	3	4	5	6	7	8	9	10	11	12	
A	2	4	2	7	5	3	8	2	1	6	4	6

עבור $k=1$ נעשה שאילתא RMQ על 2 קטעים ונעשה איחוד מקסימלי.
עבור $k > 1$ נמצא האיבר המקסימלי בכל קטע ע"י שאילתא RMQ ונבנה תור עדיפות – עבור כל קטע נחזיק את טווח הקטע והמקסימום בו.

1	2	3	4	5	6	7	8	9	10	11	12	
A	2	4	2	7	5	3	8	2	1	6	4	6
	$(2,6), 7$			$(8,11), 6$								

כעת כאשר נמצא את 7 ונלק לאינדקס שלו(לפי מצביע) ולאחר מכן הצעוב יהפוך ל 2 קטעים (ימין ושמאל) לכך
נמצא את המקסימום בקטע הצעוב בכל צד ונכנים לערמה.

(2,6), 7	(8,11), 6		
---------------------	-----------	--	--



וכך ממשיר, נוציא את האיבר המקסימלי, נדפיס אותו וכך הלאה.

מספר האיברים בערמה בהתחלה t , אם $k \leq t$ נכnis t , אם $k > t$ אזי אנו יודעים שהיימים גם איברים לא מקסימליים וכן נוכל ללקחת את k הקטועים המובילים בערמה.

לכן באיטרציה הראשונה יש לכל היותר k איברים בערמה.

בכל איטרציה אנו מוצאים איבר ומושיפים 2 לכל היותר, שכן מגדילים ב1 וכן מספר האיברים יהיה לכל היותר $2k$.

כל הכנסה לוקחת $\log k$ ולכן סה"כ מקבלים

Time: $O(t + k \log k)$ (the heap contains at most $2k$ elements).

ראינו את המבנה הנ"ל עבור k קטיעים גדולים ביותר וכאן משתמש בו עבור הבעיה שלנו, נאחד את המרכיבים למערך אחד.

הגענו לצומת v לפי רידיה בעז הסיפות ומסתכלים על האבות הקדמוניים, ומעניינים אותנו המספרים שמופיעים בעלים. נסתכל על המרכיבים של w_1, w_2, \dots, w_n ונסתכל על כל השלשות שהאיבר השלישי שלהם בין L_v, R_v .
לכן נבצע חיפוש בינארי-הערך הקטן ביותר שגודלו שווה ל14 והערך הגדל ביותר שגודלו שווה ל21 וכן קיבלנו טווחים של שלשות W_i -ים. נרצה למצוא את k השלשות הכי טובות וכן נשתמש במבנה עבור קטיעים.

זמן ריצה :

t מספר האבות הקדמוניים – לכל היותר האורך של P וכן נוכל לחסום $|m| \leq |t|$.

רידיה בעז – $O(m \log n)$ (לפי א"ב קבוע).

מציאת האינטראול $O(m \log n)$ – קטיעים בכל אחד מהאבות הקדמוניים בעזרת חיפוש בינארי. לכל היותר m מערכים ובכל אחד חיפושBINARI.

מציאת k הכי טובים – $O(m + k \log k)$

Problem	Space	Query time
Document listing	$O(n)$ words	$O(m + n \log n)$
Top-k document retrieval	$O(n)$ words	$O(m \log n + k \log k)$

הרצאה 8

A top- k retrieval data structure - מבנה עבור קב' מסמכים $\{d_1 \dots d_D\}$ עבור מציגת k המסמכים המכילים את P היכי הרובה פעמיים.

$$n = \sum_i |d_i|, D = \text{number of docs}$$

Suffix array - אם נמספר את הסיפות של מחוזת S (היכי גובה) – ה- i -suffix array של S יהיה מערך עם מספרי הסיפות מסוימים לפי סדר לקסיקוגרפי של הסיפות.

suffix range - בהינתן suffix array, ה- P הוא אינטראול האינדקסים ב- P .
המתחלים ב- P .

The **suffix range** of P in S is the interval $[L, R]$ s.t. P is a prefix of $SA[i]$ if and only if $i \in [L, R]$.

Example

$S = \text{mississippi\$}$

1	mississippi\$	8	ippi\$
2	ississippi\$	5	issippi\$
3	ssissippi\$	2	ississippi\$
4	sissippi\$	11	i\$
5	issippi\$	1	mississippi\$
6	ssi\$	10	pi\$
7	sippi\$	9	ppi\$
8	ippi\$	7	sippi\$
9	ppi\$	4	sissippi\$
10	pi\$	6	ssippi\$
11	i\$	3	ssissippi\$
12	\$	12	\$

$P = \text{iss}$

$L = 2, R = 3$

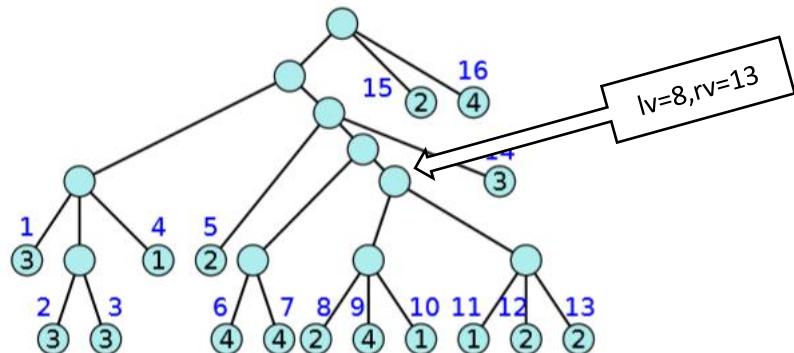
The suffix array:

8	5	2	11	1	10	9	7	4	6	3	12
---	---	---	----	---	----	---	---	---	---	---	----

suffix range CSA – מבנה עבור השאלות מהו האיבר ה- i -ב- S ומהו ה- $t_{search}(m)$ – הזמן לחישוב ה- $t_{search}(m)$ של suffix range של מחוזת P . נסמן: t_{sa} – הזמן לחישוב $SA[i]$, $t_{search}(m)$ – הזמן לחישוב ה- $t_{search}(m)$ באורך m

$t_{SA} = O(\log \sigma \cdot \log^{1+\varepsilon} n), t_{search}(m) = O(m \log \sigma)$ אשר (compressed suffix array FM – index

Succinct data structure for top- k retrieval

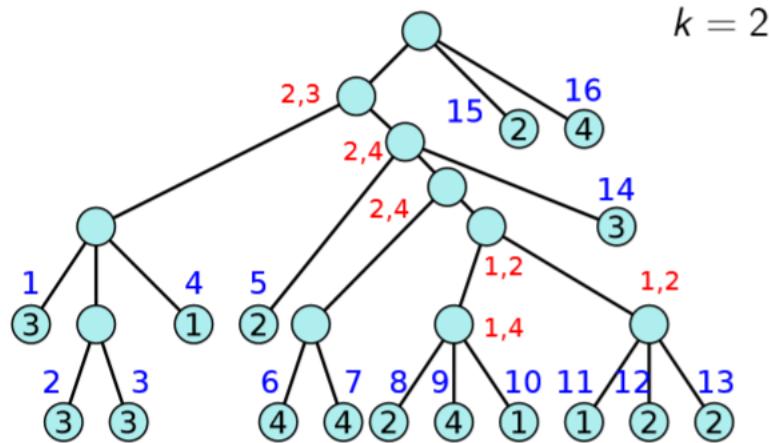


עבור $d_1 \dots d_D = S$ מבנה הנתונים המאוחסן CSA על S עם מבנים נוספים. בניית את עץ הסיפות להגדרת המבנה, לא נרצה לשמר אותו כי יקח $log n$ ביטים. התווים בערים זה איפה מתחליה הסיפה (מסמן).

תהי T סופית בעץ עבור S . עבור צומת v יהיו $[v_1, v_2]$ טווח העלים של תת-העץ של v .

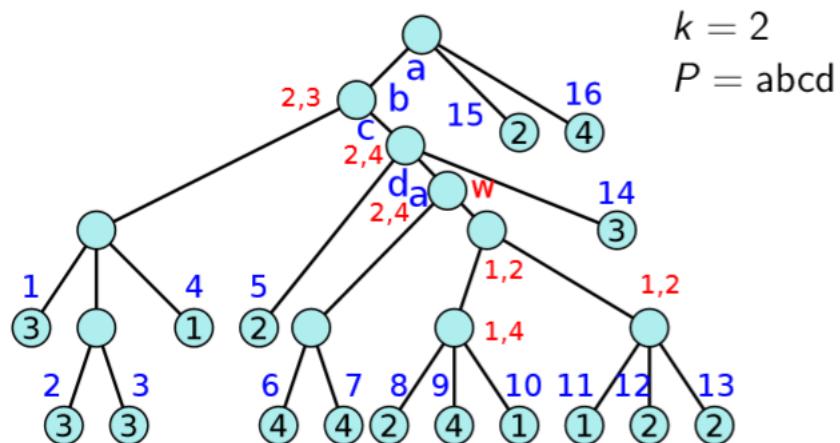
- C_v^k - קבוצת k המספרים המופיעים הici הרבה פעמים בתת-העץ המושרש ב- v .

עבור k קבוע, נניח שאנו מחשנים עבור צומת v קבוצה C_v^k המכילה את k המספרים שמופיעים הici הרבה פעמים בעלים המושרים מ- v .

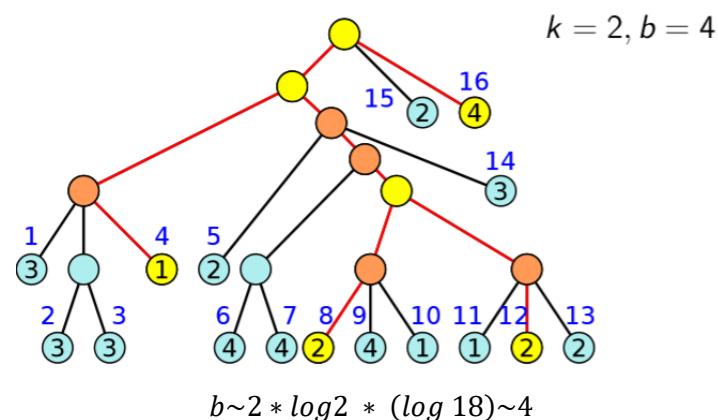


- ה-*locus* הוא הצומת הgebowa ביותר v ב- S -P generalized suffix array המיצגת ע"י המסלול מהשורש ל- v .

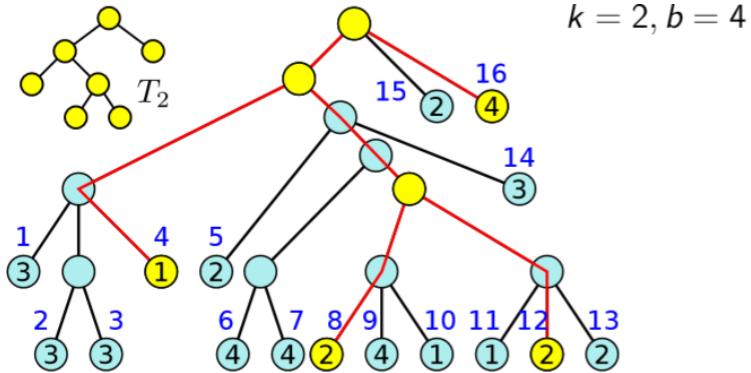
אנו יכולים לענות על שאלתא עבור k, P, w מיציאת *the locus* - v של P ולהדפיס את C_v^k
לעומת זאת לאחסן את T ואת C_v^k דרוש הרבה מאוד מאד זיכרון, הפתרון הוא להגדיר עץ קטן יותר. דוגמא:



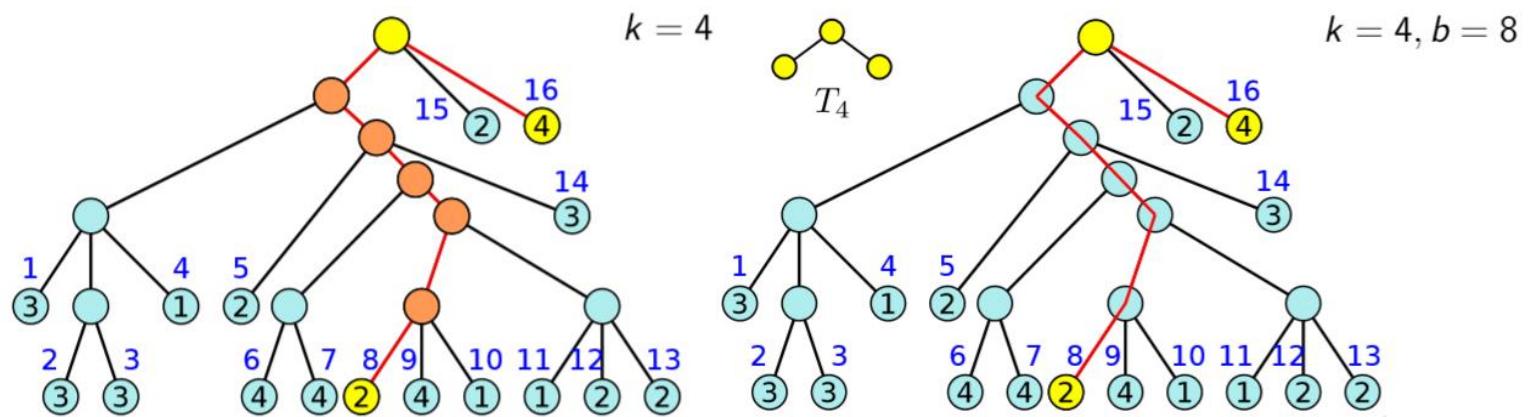
לכל $k=2,4,8,\dots$ נגידיר $*k=b$, כאשר $n \sim k * \log^{1+\epsilon} n$. נסמן כל עלה v -י ב- T ונגדיר ב- T_k את תת-העץ המכוי
המושר ע"י העלים המסומנים ואבותיהם והורדת צמתים עם בן יחיד.



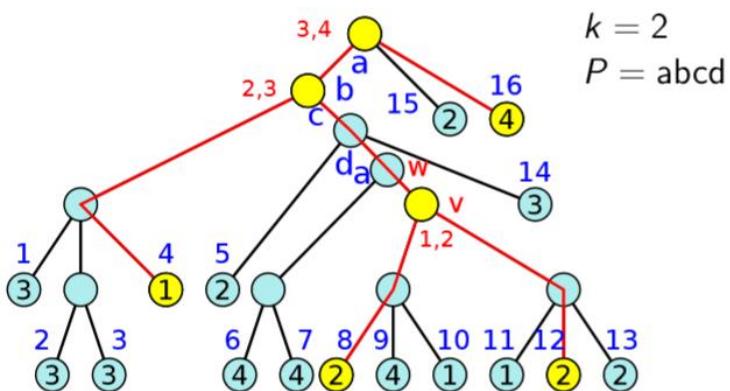
לכן נקבל :



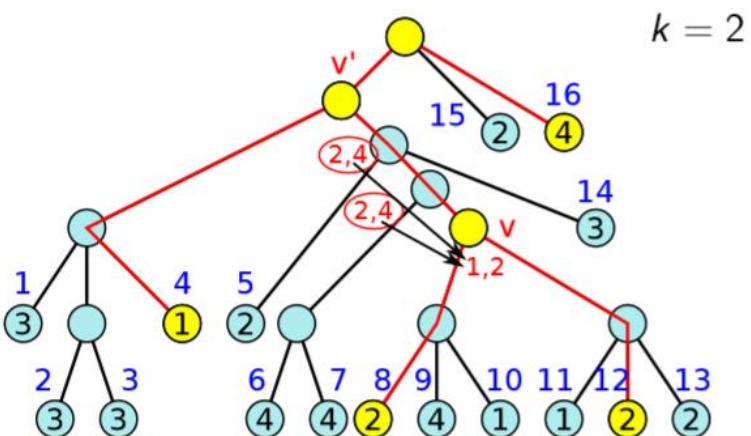
נמשר עבור $k=4$



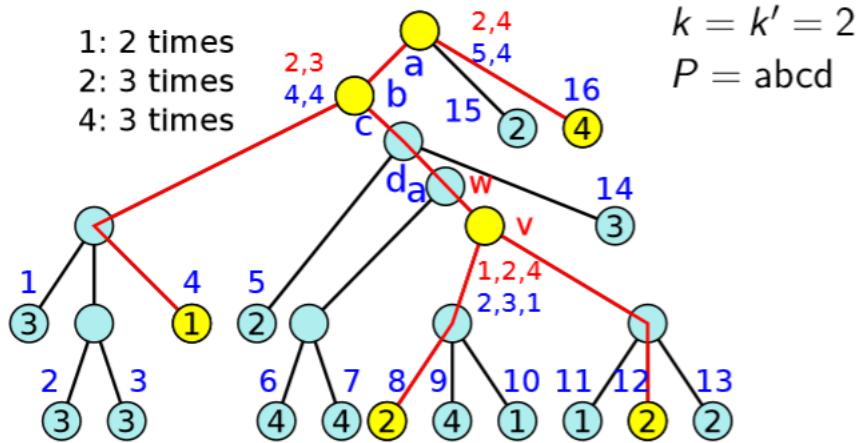
בהתנתק T_k האם נוכל לפתור את הבעיה k, P ? נראה דוגמא :



הא $locus$ של P שונה מ $locus$ של P ב- T_k שכן אנו לא יכולים לענות על השאלה.



הפתרון הוא להוסיף לקבוצה C_v^k של כל צומת v בדרכו בין v לבין בעץ T_k . ולכן גנדיר $\widehat{C}_v^k \in d$ נאחסן את התדיירות של d בתת העץ המושרש על v .
ולכן המספרים הכהולים הם התדיירות בהתאם למס' המסמך.



נרד ונגיע לו בו רשום את המסמכים והתדיירות, ואנו רוצחים מופיעים של תת העץ w . למשל בא בדוגמה 3,1,2,4: וכאן היינו רוצחים להחזיר את 4.2,4. ולכן צריך לחשב את שמתוך זה.

כעת נוכל לענות על השאלה:

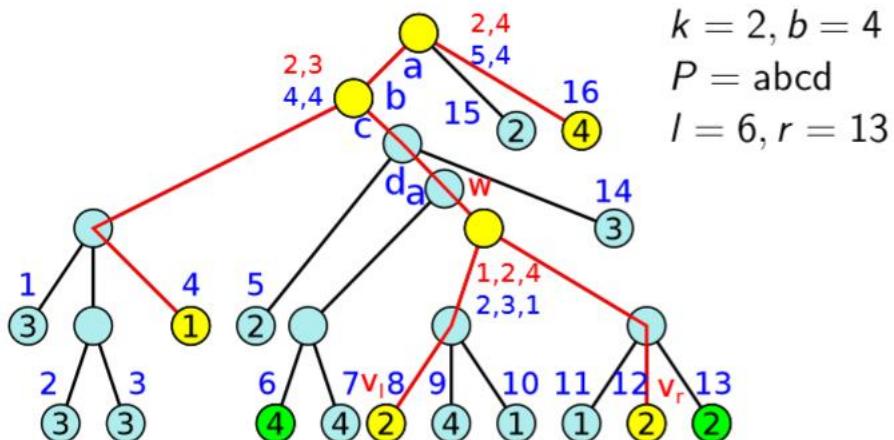
1. k = החזקה הכי קטנה של 2 שגדולה שווה ל' k'
2. נמצא את ה v locus של P ב- T_k
3. עבור כל $\widehat{C}_v^k \in d$ לחשב את התדיירות של d בעליים בתת העץ המושרש ע"י w כאשר w הוא ה v locus של P ב- T_k .
4. החזר את k' הערכים הגבוהים ביותר.

Finding the locus v of P in T_k

מה שנוטר לעשות זה לעבור על בעליים של תת העץ של w ולא בו ולהוסיף אותם לסכימה (בדומה יש 2 בעליים דלעיל)

1. משתמש ב-*CSA* לחשב את $SR[z, l]$, כאשר z, l הם מספרים של הטווח של w .
2. $**18**$

3. v הוא ה *LCA* של v_r, v_l



יש בעיה כי לא שומרים את העץ הגדול, כלומר לא יודעים מי האב הקדמון ואת העלים שלא נמצאים בעץ!

אנו יודעים את טווח הסיפות, שומרים *FMindeX* וכן את טווח הסיפות הרלוונטיות, لكن אם ניקח את עלה 13-6 יהיו העליים הרצויים של הצומת W הרצוי, אך אנו יודעים את העליים הרצויים. מציאת ש בעץ הקטן – חלק מהסיפות יהיה עליים מסוימים, כל הסיפות הללו, האב הקדמון השותף של העלה השמאלי ביותר והימני ביותר יתנו צומת מסוימת.

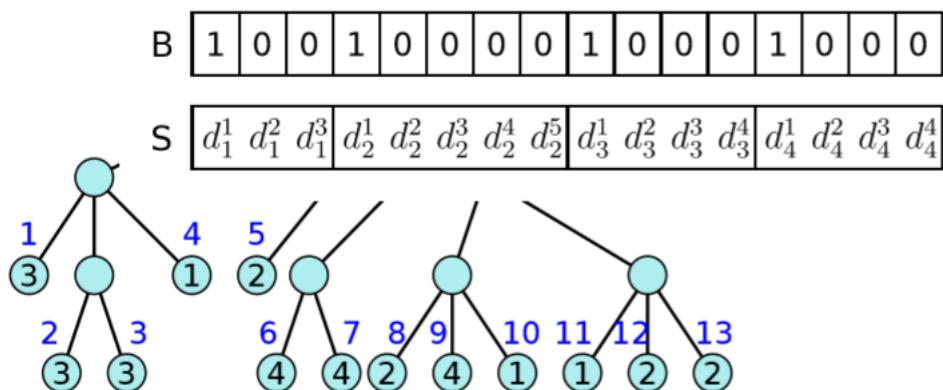
אם רוצים למצוא את הצומת τ צריך ללקח את העלה הצהוב הכי שמאלית בטווח והכי ימני בטווח ולחשב את האב הקדמון המשותף הנמוך ביותר.

ניתן לעשות את החיפוש לאחר עם *fm-index*.

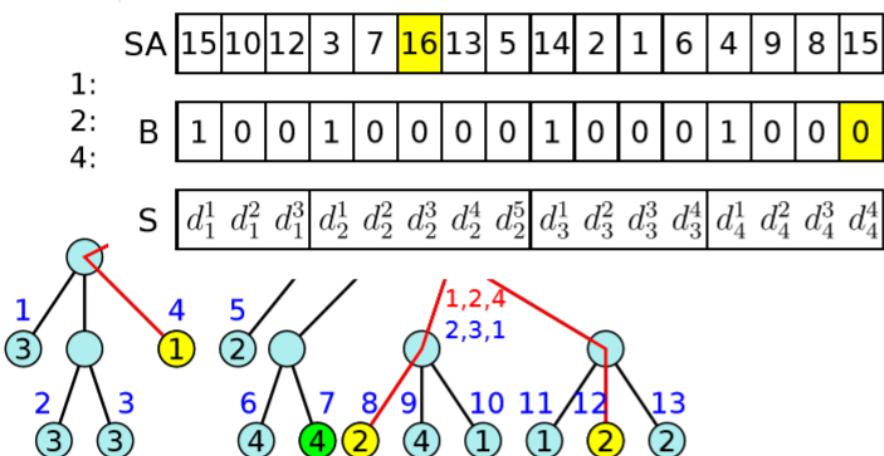
האלגוריתם צריך לעבור על כל העליים שלא נמצאים בעץ ובטווח, [6,13] ו[8,12] וכן יודעים את המספרים המעניינים. נזכיר להראות איך מחשבים את המספרים בעליים:

**

נשמר מחוזת B , אורכה שווה ל-5, בכל פעם שמתחליה מחוזת חדשה ונשメ ב- B .



כעת איך נחשב את מספר על של עלה 6? נחשב בא- SA את האיבר התא 6 (העלים מתאימים לסיפות בסדר הלקסיקוגרפי) ולאחר על *fm-index* נחשב את הערך של הסיפה בא- SA . וכך נרצה לדעת איפה התחליה ולפניהם נוכל לדעת לפני חיפוש לאחר ב- B את ה-1 הראשון שלפניו. נספור כמה פעמים 1 מופיע ולאחר הסיפה היא כמות ה-1 לפני).



סיימנו את תיאור מבנה הנתונים, כרגע ניתן לעשות אנליה לביצוע שאלות:

אלגוריתמים למחוזות / סיכום מאת אורי שביט

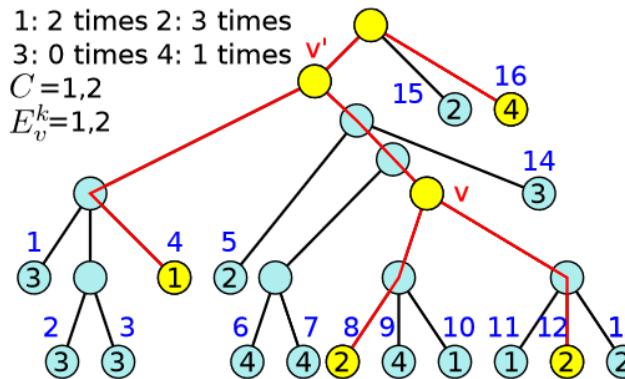
מספר העלים שנרצה לעבור עליהם הוא לכל היותר 2^k , כלומר כל העלים הרצופים שלא מסומנים ולכן יכולים להיות לכל היותר k עליים בכל צד. בכלל עלה צורך לחשב SA של אינדקס ואז rank עבורו.

זמן - $O(n \log^{1+\epsilon} k \log k \log k \log \sigma + k \log \sigma)$ FM-index ב-x-search $O(tsearch(m) + tSA)$, ובשימוש :

סיבוכיות מקומ :
אנו משתמשים במבנה index , המיקום הנוסף צריך להיות מספיק קטן : שומרם את העצים עבור $k=2,4,\dots$ עבור כל אחד מבנה LCA ושומרם את הקבוצות C_v^k וכן המיקום שבון צריכים לקבוצות הוא גדול יותר מהעץ, מספיק לחשב רק אותו.

בנוסף שומרם את המחרוזת B הפהשר פועלות rank. מחרוזת באורך X וכן המיקום שבון צריכים הוא $D \log(n/d) + O(D) + o(n)$ ונקבל סה"כ : $(n \choose x) + o(n)$

- חישוב גודל הקבוצה \widehat{C}_v^k :



נחשב את גודל הקבוצות \widehat{C}_v^k נגדיר קבוצות חדשות $E_v^k \subseteq \widehat{C}_v^k$ וניחסם את הגודל של הקבוצות E_v^k . יהי v הצעמת ב- T_k ו' v ' האב שלו ב- T_k . נתחיל כאשר $E_v^k = C_v^k$ וכן נאותחל את $C_v^k = \emptyset$. לכל צומת w בדרך מה v ' לעברו על ה策אים של העלים של w שהם לא צאאים של v ולא עברו עליהם קודם.

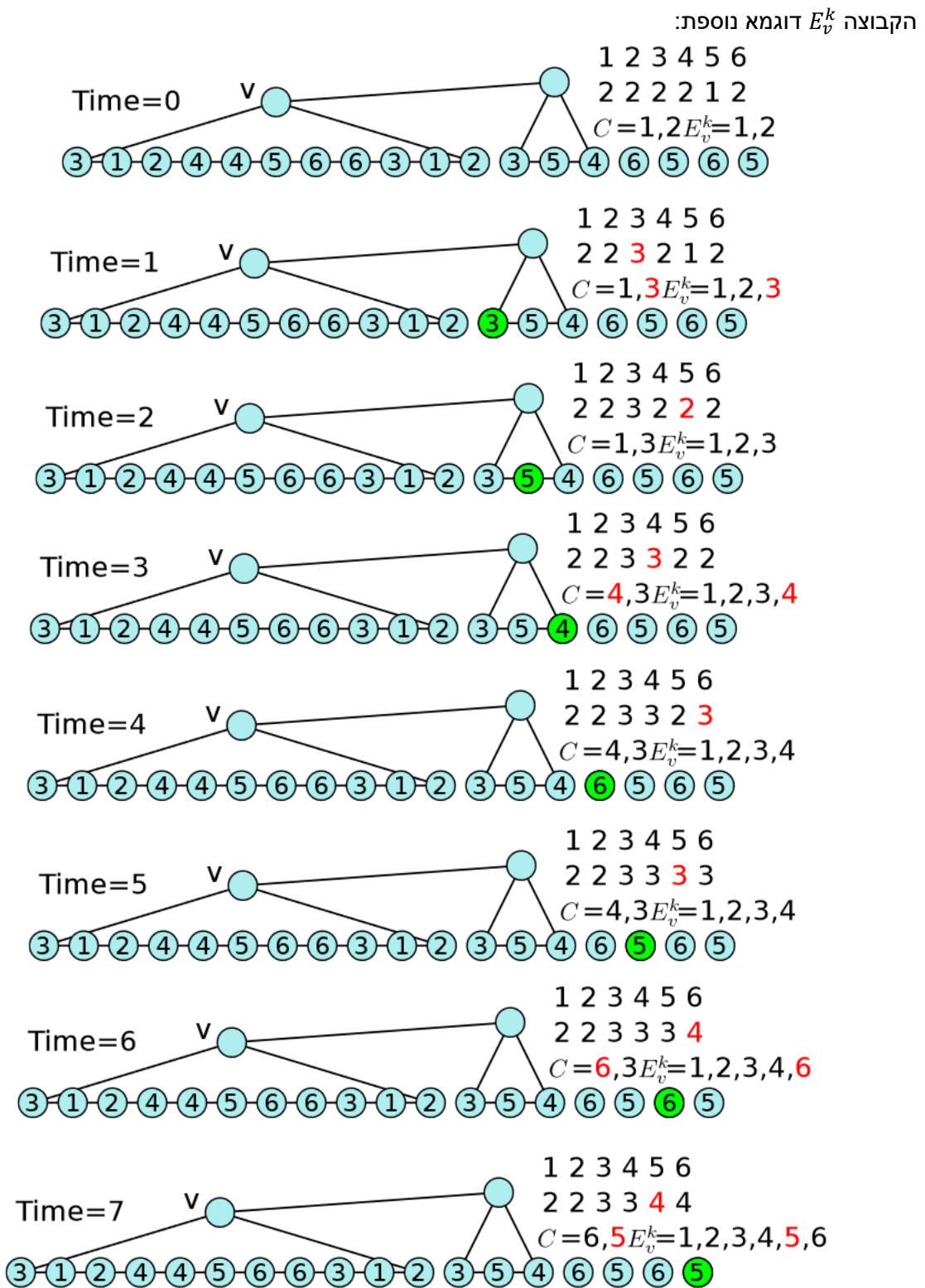
עבור כל עלה נחשב את מספר המספר p , נוסף 1 למונה התדריות של p . אם התדריות של p גדולה פחותה

מתדריות המספר $p \in C$:

$$\begin{aligned} C &\leftarrow C \cup \{d\} \setminus \{d'\} \\ E_v^k &\leftarrow E_v^k \cup \{d\} \end{aligned}$$

דוגמת הרצה :

<p>1: 2 times 2: 3 times 3: 0 times 4: 2 times $C=1,2$ $E_v^k=1,2$</p>	<p>1: 2 times 2: 3 times 3: 0 times 4: 3 times $C=4,2$ $E_v^k=1,2,4$</p>
<p>1: 2 times 2: 4 times 3: 0 times 4: 3 times $C=4,2$ $E_v^k=1,2,4$</p>	<p>1: 2 times 2: 4 times 3: 1 times 4: 3 times $C=4,2$ $E_v^k=1,2,4$</p>



חסם עבור E_v^k :

1. יהי f_0 תדירות המינימום של מסמך C בזמן 0.

2. יהי t_1 זמן המינימום שבו $|E_v^k| = 2k$ (נכיה כי קיימ t_1).

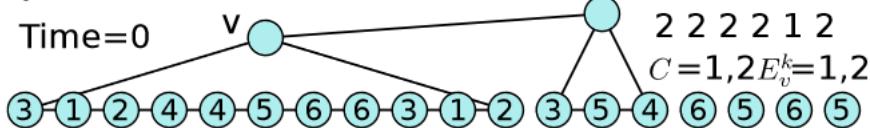
3. יהי f_1 תדירות המינימום של מסמך C בזמן t_1 .

$$f_1 \geq f_0 + 1$$

עבור מסמך d שנוסף ל E_v^k בין זמן 1 ל t_1 : התדריות של d בזמן $0 \geq f_0$, הדריות של d בזמן $t_1 \leq f_1$. $2b \geq t_1 \geq f_1 \geq f_0 + 1$. לכן d מופיע לפחות פעם אחת ב t_1 העלים שעברנו עליהם. לכן k דוגמת הריצה:

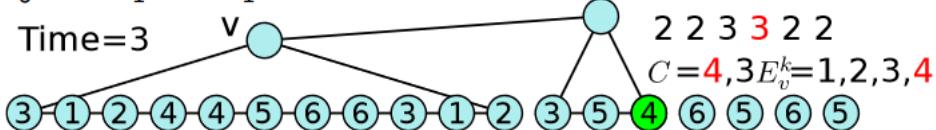
$$f_0 = 2$$

Time=0



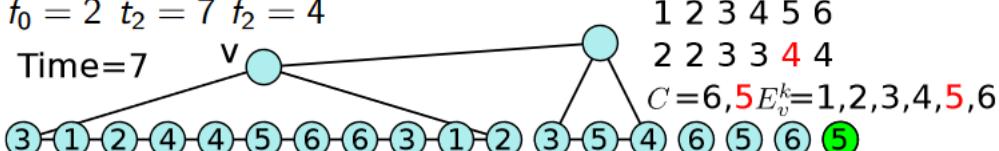
$$f_0 = 2 \quad t_1 = 3 \quad f_1 = 3$$

Time=3



$$f_0 = 2 \quad t_2 = 7 \quad f_2 = 4$$

Time=7



נכיה שזמן הסיום $|E_v^k| = sk$ for $0 \leq s' < k$

יהי t_s זמן המינימום שבו $|E_v^k| = sk$

$$2b \geq t_{s-1} \geq k + 2k + \dots + (s-1)k = \theta(s^2k)$$

$$s = O\left(\sqrt{\frac{b}{k}}\right) = O(\sqrt{l})(b = kl)$$

$$|E_v^k| \leq sk + k = O(k\sqrt{l})$$

סיבוכיות מקום (43-44 שקיף)

לכל $\widehat{C}_v^k \in d$ אנו צריכים לאחסן את d (כמספר) ואת התדריות שלו, כמוות הזיכרון לאיבר ייחד הוא $logD + logn$

ביטים. לכן סה"כ $(|\widehat{C}_v^k| logn) = O(k\sqrt{l} logn)$

$O(k\sqrt{l} \cdot logl)$ ביטים.

בעץ T_k יש нам $O(n/b)$ צמתים, כל צמות דרושת $O(klogn + k\sqrt{l} \cdot logl)$ ביטים.

לכן המקום עבור העץ T_k $= O\left(\frac{b}{n}(klogn + k\sqrt{l} \cdot logl)\right) = O\left(\frac{nlogn}{l} + \frac{nlogl}{\sqrt{l}}\right) = O\left(\frac{n}{logklog^{\epsilon/2}n}\right)$ ביטים.

המקום לכל העצים $: T_k$

$$O\left(\frac{n}{log^{\epsilon/2}n}\right) * \sum_{i=1}^{logD} \left(\frac{1}{i}\right) = O\left(\frac{n}{log^{\epsilon/2}n} loglogD\right) = o(n)$$

המקום למערך B הוא $Dlog\left(\frac{n}{D}\right) + O(D) + o(n)$

כמו כן אנו צריכים את CSA בשימוש ב-FM-index.

סה"כ מקום $\sim nlog\sigma bits$

זמן שאילתא: ובשימוש ב- σ FM-index זה $O(m log \sigma + k log k log \sigma log^{2+\epsilon} n)$

הרצאה 9 Edit Distance

להגדרות הבאות נניח כי S ו- T הן מחרוזות כך ש- $n = |T|$, $m = |S|$, $m \leq n$.

Edit distance - מספר פעולות העריכה(הוספה/מחיקה/החלפה של תו/הזהה של תת מחרוזת) המינימלי הנדרש כדי להגיע מ- S ל- T . פעולות העריכה הן החלפה, הורדה והוספה של אות. $ED(S, T)$.

דוגמה : $S = abcabc, T = abdabbc$

פעולות $abcabc \rightarrow abdabc \rightarrow abdabbc$

רץ' פשוט של פעולות עריכה להפיכת S לאפשר להיות מיוצג ע"י $\text{Alignment}(S, T)$. לדוגמה:

$\text{abca-}bc$	abcabc-
abdabbc	abdabbc

T' ו- S' הן מחרוזות של S ו- T אשר כל המקפים ('-) מ- S' ומ- T' נותנת את S ו- T . $\text{Alignment}(S, T')$ או $\text{Alignment}(T, S')$ ימינה-למטרה ולמטרה, שכן

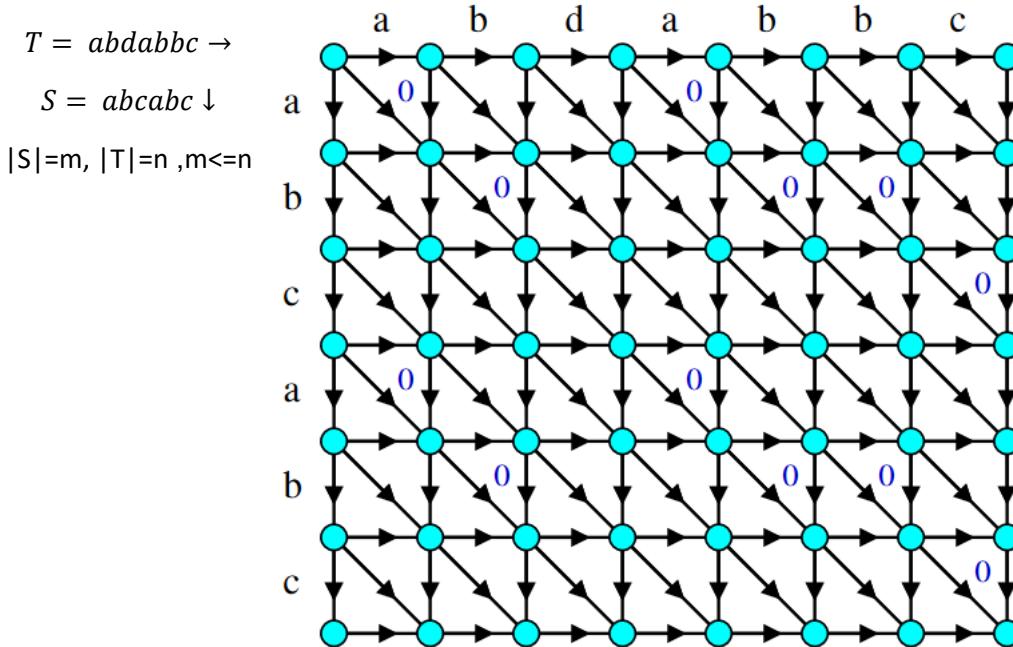
המחיר של (T', S') הוא מספר האינדקסים i עבור $T'[i] \neq S'[i]$.

$$ED(S, T) = \text{minimum cost of alignment of } S \text{ and } T$$

Alignment graph - גרף עם $(m+1) \times (n+1)$ קודקודים המוחברים בקשרות ימינה, ימינה-למטרה ולמטרה, שככל במשקל 1, למעט הקשתות האלכסוניות $((j, i), (j-1, i-1))$ אם $T[j] = S[i]$, משקלן הוא 0. (S כתובה מלמעלה למטרה ליד הקשתות שבין השורות ו- T משמאלי לימיין ליד הקשתות שבין הטורים).

בכדי לפטור את הבעיה נציג אותה ע"י בעיה על גרפים :

אם יש تو זהה יהיה משקל 0, אחרת יש משקל 1.



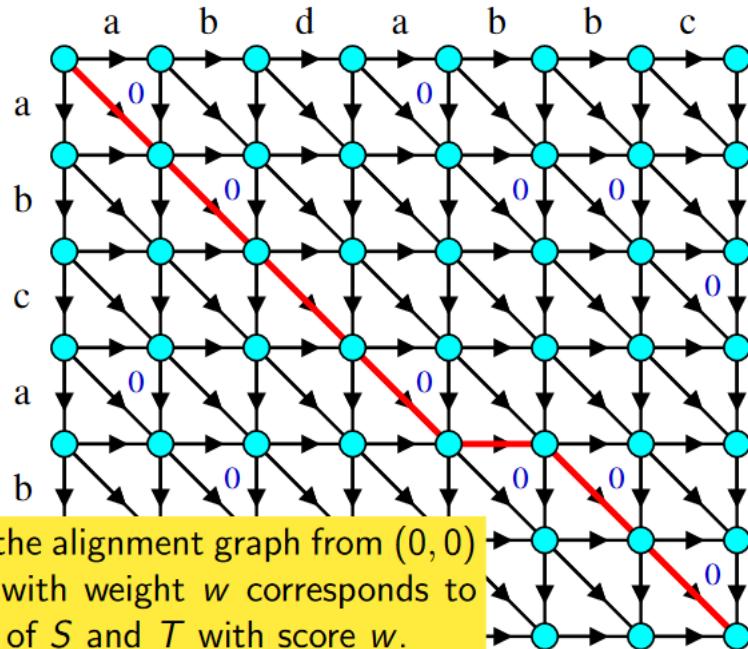
הגדרת הפורמלית Alignment graph מכיל קודקודים (i, j) לכל $0 \leq i \leq m, 0 \leq j \leq n$ עבור קשתות :

$$((i-1, j), (i, j)) \quad \forall i > 0, \forall j \geq 0 \quad (\text{weight} = 1).1$$

$$((i, j-1), (i, j)) \quad \forall i > 0, \forall j > 0 \quad (\text{weight} = 1).2$$

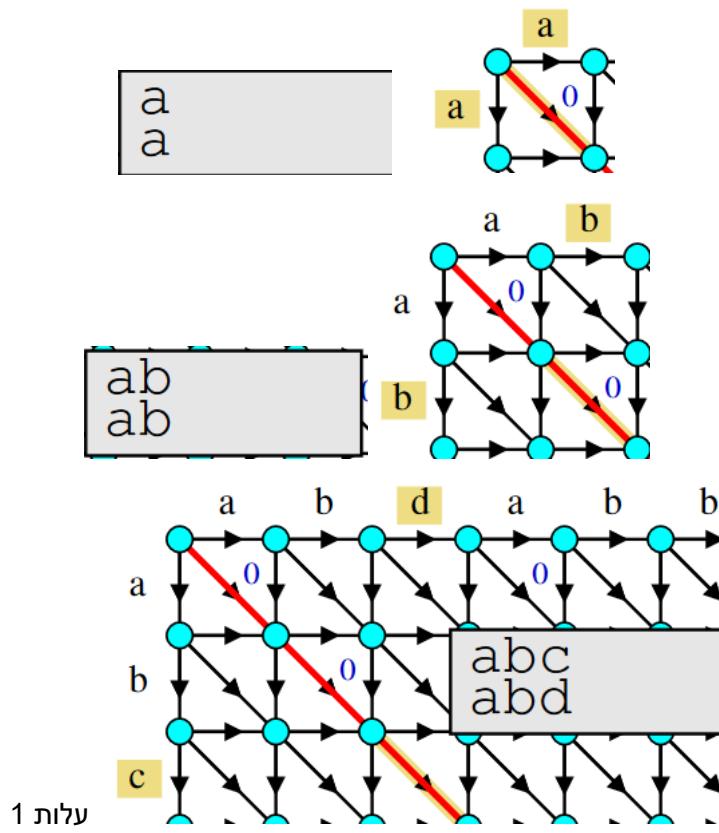
$$((i-1, j-1), (i, j)) \quad \forall i > 0, \forall j > 0 \quad (\text{weight} = 1 \text{ if } S[i] \neq T[i] \text{ and weight 0 otherwise}).3$$

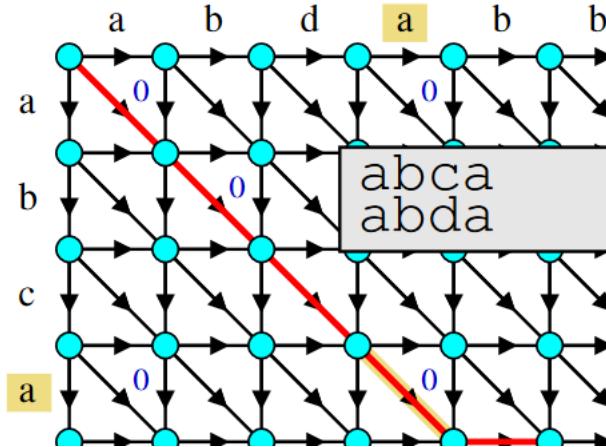
לכל מסלול m - n ($0,0$) ב- $S-T$ alignment graph יש מסלול של $S-T$ ולהיפך.



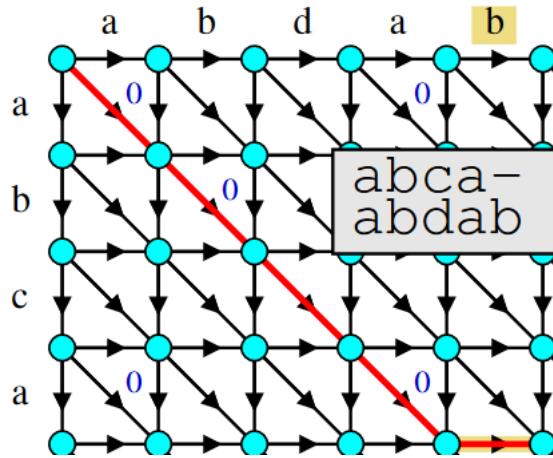
הבעיה למשא את המשקל המינימלי של מסלול בין שמאל קיצון לימין קיצון.

נסתכל על הליכה במסלול :





הופכים את המחרוזת האנכית לאופקית בהליכה במסלול!
כשעוברים על קשת אופקית נרשום – ומתחתיו נרשום את האות האופקית המתאימה.



אם יש מסלול אז יש העמדה בין S ל- T .
ע"י כך יש העמדה הפוכה לפיה הליכה הפוכה למה שעשינו, - הליכה אופקית, a, a הליכה אלכסונית ...

Finding the optimal alignment

טענה: לכל מסלול מ- $(0,0)$ ל- (n,m) ב-graph alignment יש alignment של S ו- T ולהיפך.

יהי T מחרוזות באורך n , S מחרוזות באורך m אזי: $ED(S, T) = \min_{\text{paths } P} \sum_{(i,j) \in P} w(i,j)$.

2. $ED(S[1 \dots i], T[1 \dots j]) = \min_{\text{paths } P} \sum_{(i,j) \in P} w(i,j)$.

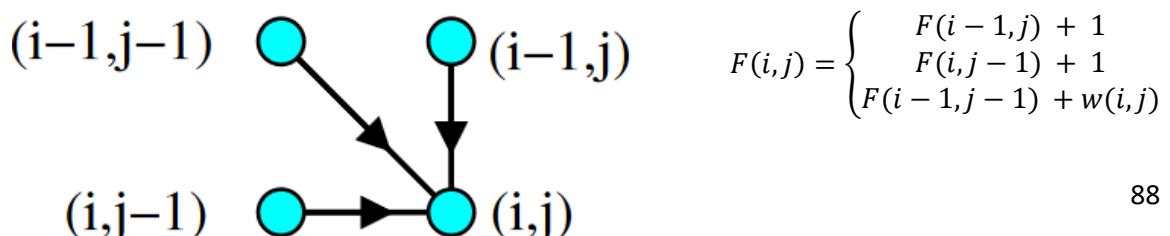
נשתמש בתכנון דינامي Dynamic programming

תכנות דינמי למציאת $ED(S, T)$ – נגדיר מערך F מסדר $n \times m$ ונמלא אותו כך:

$$w(i,j) = \text{weight of edge from } (i-1, j-1) \text{ to } (i, j) = \begin{cases} 1 & S[i] \neq T[j] \\ 0 & \text{otherwise} \end{cases}$$

. $F[0 \dots m, 0 \dots n]$ s.t $F(i, j) = \min \text{ weight of path from } (0,0) \text{ to } (i,j) (= ED(i,j))$

נמלא את F האמצעות רקורסיבית: $F(0,0) = 0$, $F(i,0) = i$, $F(0,j) = j$.



אלגוריתמים למחuzeות / סיכום מאת אורי שביט

כדי למצוא את ה-*h*-alignment נחזור מ- (n, m) לפניהם (ה)תא שלפיו מולא הערך בתא הנוכחי.

מיצרים טבלה זו מידית ועוברים על הטבלה וממלאים אותה מלמעלה למטה, דוגמא:

<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>a</td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1								b	2								c	3								a	4								b	5								c	6								<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>a</td><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0							b	2	1							c	3	2							a	4	3							b	5								c	6								<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>a</td><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>5</td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>6</td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0							b	2	1							c	3	2							a	4	3							b	5	4							c	6	5						
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1																																																																																																																																																																																																																						
b	2																																																																																																																																																																																																																						
c	3																																																																																																																																																																																																																						
a	4																																																																																																																																																																																																																						
b	5																																																																																																																																																																																																																						
c	6																																																																																																																																																																																																																						
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0																																																																																																																																																																																																																					
b	2	1																																																																																																																																																																																																																					
c	3	2																																																																																																																																																																																																																					
a	4	3																																																																																																																																																																																																																					
b	5																																																																																																																																																																																																																						
c	6																																																																																																																																																																																																																						
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0																																																																																																																																																																																																																					
b	2	1																																																																																																																																																																																																																					
c	3	2																																																																																																																																																																																																																					
a	4	3																																																																																																																																																																																																																					
b	5	4																																																																																																																																																																																																																					
c	6	5																																																																																																																																																																																																																					
<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>a</td><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>5</td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>6</td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0							b	2	1							c	3	2							a	4	3							b	5	4							c	6	5							<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td></td><td></td><td></td><td></td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td></td><td></td><td></td><td></td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td></td><td></td><td></td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td></td><td></td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>1</td><td></td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1						b	2	1	0	1					c	3	2	1	1					a	4	3	2	2	1				b	5	4	3	3	2	1			c	6	5	4	4	3	2	1		<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	0	1	2	3	4	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0																																																																																																																																																																																																																					
b	2	1																																																																																																																																																																																																																					
c	3	2																																																																																																																																																																																																																					
a	4	3																																																																																																																																																																																																																					
b	5	4																																																																																																																																																																																																																					
c	6	5																																																																																																																																																																																																																					
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1																																																																																																																																																																																																																				
b	2	1	0	1																																																																																																																																																																																																																			
c	3	2	1	1																																																																																																																																																																																																																			
a	4	3	2	2	1																																																																																																																																																																																																																		
b	5	4	3	3	2	1																																																																																																																																																																																																																	
c	6	5	4	4	3	2	1																																																																																																																																																																																																																
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	0	1	2	3	4																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															

תוצאה:

	a	b	d	a	b	b	c	
a	0	1	2	3	4	5	6	7
b	1	0	1	2	3	4	5	6
c	2	1	0	1	2	3	4	5
a	3	2	1	1	2	3	4	4
b	4	3	2	2	1	2	3	4
c	5	4	3	3	2	1	2	3
a	6	5	4	4	3	2	2	2

העודה האופטימלית יכולה להימצא החל מהתא (n, m) וכל צעד בתזוזה לתא אשר הגענו ממנו לערך הנוכחי:

<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	2	3	4	5	6	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	2	3	4	5	6	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	2	3	4	5	6	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	2	3	4	5	6																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	2	3	4	5	6																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	2	3	4	5	6																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															
<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	2	3	4	5	6	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr><th></th><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><th>a</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </thead> <tbody> <tr><td>a</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>b</td><td>2</td><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><td>c</td><td>3</td><td>2</td><td>1</td><td>1</td><td>2</td><td>3</td><td>4</td><td>4</td></tr> <tr><td>a</td><td>4</td><td>3</td><td>2</td><td>2</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> <tr><td>b</td><td>5</td><td>4</td><td>3</td><td>3</td><td>2</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>c</td><td>6</td><td>5</td><td>4</td><td>4</td><td>3</td><td>2</td><td>2</td><td>2</td></tr> </tbody> </table>		a	b	d	a	b	b	c	a	0	1	2	3	4	5	6	7	a	1	0	1	2	3	4	5	6	b	2	1	0	1	2	3	4	5	c	3	2	1	1	2	3	4	4	a	4	3	2	2	1	2	3	4	b	5	4	3	3	2	1	2	3	c	6	5	4	4	3	2	2	2																																																																								
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	2	3	4	5	6																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															
	a	b	d	a	b	b	c																																																																																																																																																																																																																
a	0	1	2	3	4	5	6	7																																																																																																																																																																																																															
a	1	0	1	2	3	4	5	6																																																																																																																																																																																																															
b	2	1	0	1	2	3	4	5																																																																																																																																																																																																															
c	3	2	1	1	2	3	4	4																																																																																																																																																																																																															
a	4	3	2	2	1	2	3	4																																																																																																																																																																																																															
b	5	4	3	3	2	1	2	3																																																																																																																																																																																																															
c	6	5	4	4	3	2	2	2																																																																																																																																																																																																															

אלגוריתמים למחוזות / סיכום מאת אורי שביט

המסלול חייב להגיע לצומת הסיום, רצים לדעת כיצד הגיעו, כדי לעשות זאת נסתכל על הערכים –

1. לא הגיעו משמאל כי היה צריך להיות 3
2. לא הגיעו מלמעלה כי המשקל היה אמר או היה 4
3. הגיעו מאלכסון כיון שהמשקל של אלכסונית הוא 1 או 0 וכן קיבלנו 2.

	a	b	d	a	b	b	c
a	0	1	2	3	4	5	6
b	1	0	1	2	3	4	5
c	2	1	0	1	2	3	4
a	3	2	1	1	2	3	4
b	4	3	2	2	1	2	3
c	5	4	3	3	2	1	2
a	6	5	4	4	3	2	2

סיבוכיות זמן : $O(mn)$

סיבוכיות מקום : $O(mn)$

בריצה הלאנו ומלאנו את הטבלה וכן קורע כגדל הטלול.

נניח שאנו רצים לחשב את מרחק ההליכה, רצים לשמר על זמן ולהקטין את הזיכרון: כל הערכים שאנו מחשבים אנו צריכים את העמודה הנוכחיית והקודמת ולכן נוכל להקטין את המקום ל(m) O .

אם אנו רצים לחשב את הערך של $(S, T) ED$ וכן זוקקים ל(m) O זיכרון, לאחר חישוב כל העמודה אנו לא זוקקים לערכים יותר של העמודה הקודמת.

<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>6</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	0							1							2							3							4							5							6							<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>5</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>6</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	0	1						1	0						2	1						3	2						4	3						5	4						6	5						<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>0</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>5</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	1	2						0	1						1	0						2	1						3	2						4	3						5	4					
a	b	d	a	b	b	c																																																																																																																																																																				
0																																																																																																																																																																										
1																																																																																																																																																																										
2																																																																																																																																																																										
3																																																																																																																																																																										
4																																																																																																																																																																										
5																																																																																																																																																																										
6																																																																																																																																																																										
a	b	d	a	b	b	c																																																																																																																																																																				
0	1																																																																																																																																																																									
1	0																																																																																																																																																																									
2	1																																																																																																																																																																									
3	2																																																																																																																																																																									
4	3																																																																																																																																																																									
5	4																																																																																																																																																																									
6	5																																																																																																																																																																									
a	b	d	a	b	b	c																																																																																																																																																																				
1	2																																																																																																																																																																									
0	1																																																																																																																																																																									
1	0																																																																																																																																																																									
2	1																																																																																																																																																																									
3	2																																																																																																																																																																									
4	3																																																																																																																																																																									
5	4																																																																																																																																																																									
<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>2</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>0</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	2	3						1	2						0	1						1	1						2	2						3	3						4	4						<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>3</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	3	4						2	3						1	2						2	1						3	2						4	3						<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>4</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>5</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>6</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	4	5						3	4						2	3						3	2						4	1						5	2						6	3												
a	b	d	a	b	b	c																																																																																																																																																																				
2	3																																																																																																																																																																									
1	2																																																																																																																																																																									
0	1																																																																																																																																																																									
1	1																																																																																																																																																																									
2	2																																																																																																																																																																									
3	3																																																																																																																																																																									
4	4																																																																																																																																																																									
a	b	d	a	b	b	c																																																																																																																																																																				
3	4																																																																																																																																																																									
2	3																																																																																																																																																																									
1	2																																																																																																																																																																									
2	1																																																																																																																																																																									
3	2																																																																																																																																																																									
4	3																																																																																																																																																																									
a	b	d	a	b	b	c																																																																																																																																																																				
4	5																																																																																																																																																																									
3	4																																																																																																																																																																									
2	3																																																																																																																																																																									
3	2																																																																																																																																																																									
4	1																																																																																																																																																																									
5	2																																																																																																																																																																									
6	3																																																																																																																																																																									
<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td>5</td><td>6</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>4</td><td>5</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>3</td><td>4</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>1</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td>2</td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c	5	6						4	5						3	4						2	3						1	2						2	2						<table border="1" style="display: inline-table; vertical-align: top; border-collapse: collapse; width: 100px; height: 100px;"> <tr><th>a</th><th>b</th><th>d</th><th>a</th><th>b</th><th>b</th><th>c</th></tr> <tr><td></td><td></td><td></td><td>6</td><td>7</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>5</td><td>6</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>4</td><td>5</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>3</td><td>4</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>2</td><td>3</td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>1</td><td>2</td><td></td><td></td></tr> </table>	a	b	d	a	b	b	c				6	7						5	6						4	5						3	4						2	3						1	2																																																																									
a	b	d	a	b	b	c																																																																																																																																																																				
5	6																																																																																																																																																																									
4	5																																																																																																																																																																									
3	4																																																																																																																																																																									
2	3																																																																																																																																																																									
1	2																																																																																																																																																																									
2	2																																																																																																																																																																									
a	b	d	a	b	b	c																																																																																																																																																																				
			6	7																																																																																																																																																																						
			5	6																																																																																																																																																																						
			4	5																																																																																																																																																																						
			3	4																																																																																																																																																																						
			2	3																																																																																																																																																																						
			1	2																																																																																																																																																																						

האם אנו יכולים למצוא את העמידה האופטימלית ב(m) O מקום?

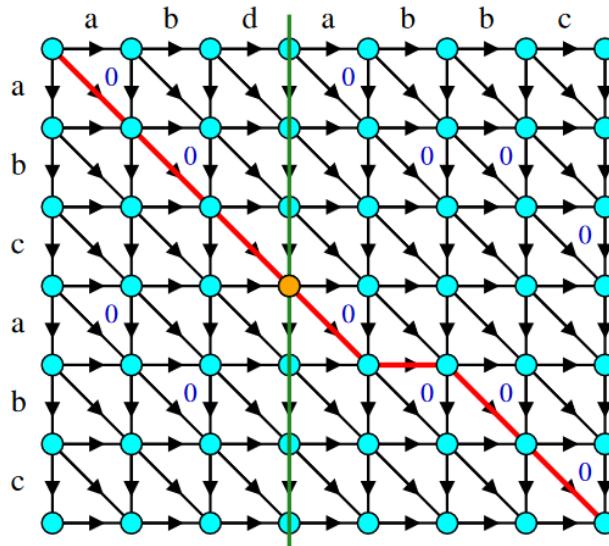
האם אנו יכולים לשמר את המסלול האופטימלי? אנו מוצאים את הערך ולא את המסלול.

סרטון הסבר – Hirschberg's algorithm

אלג' Hirschberg (תכנון דינמי עם פחות מקום) - בכל פעם שנתכן תכנון דינמי נשמר רק את ערכי ה-ED של העמודה הקודמת. נבצע תכנון דינמי לשני חצאי ה-alignment graph וنبחר מסלול שעובר דרך $(j/2, j)$ עם עבורו סכום שני ה-ED עד אליה וממנה מינימלי. ה-ED מהנקודה והלאה יוחשב בעזרת מערך F^A (ובע) עם אותה נוסחת תכנון דינמי פרט לכך שבעומדה שאיתה מתחילה יש את 0, $m-1 \dots m, 0$ (למטה) וכן שמלאים מעלה ושמאליה. לאחר שמצאנו את הנקודה $(j/2, j)$ נבצע רקורסיבית את האלג' עבור הדרכים מ- $(0,0)$ ל- $(j/2, j)$ ו- $(j/2, m)$.

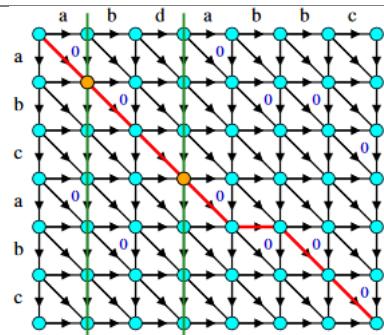
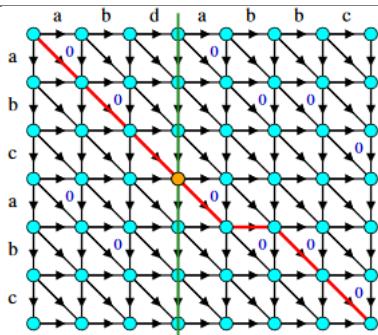
הפתרון הוא לא טריויאלי – נשתמש באלגוריתם רקורסיבי – ניקח את העמודה האמצעית (קו יירוק בגרף) כעת נרצה למצוא את הנקודה ע"ג העמודה שבה עובר המסלול. אם נדע לעשות זאת נוכל למצאו את המסלול ברקורסיה.

מסלול אופטימי $(0,0) \rightarrow (j, n)$ העובר ב $(j/2, j)$ מורכב מרכיב 2 מסלולים אופטימליים: $(0,0) \rightarrow (j, n/2) \rightarrow (m, n)$.



1. נמצא את j כך שהמסלול האופטימי עובר ב $(j/2, j)$.

2. ברקורסיה נמצא את המסלולים $(j, n/2) \rightarrow (m, n), (0,0) \rightarrow (j, n/2)$.



המטרה היא איך למצוא את הנקודה הכתום (דרכו עובר מסלול אופטימי בין $(n, m) \rightarrow (0,0)$) נחשב תחילת טבלא F כשאר נමלא עד הנקודה הכתומה, כאשר נשמר רק 2 עמודות בזיכרון בכל צעד. נחשב משקל מינימלי של כל צומת על הפס היוק (n, m) . נוכל לעשות זאת לכל צד וחיבור שלהם ייתן את המשקל המינימלי.

הגדרה: $\hat{F} = \min \text{weight of path } (i, j) \text{ to } (m, n)$

אלגוריתמים למחוזות / סיכום מאות אורי שביט

1. לכל j : חשב $F(j, n/2)$ and $F^\wedge(j, n/2)$

נתחיל מהעמודה הנטונה ונחשב את העמודה הבאה עד להגעה לנקודה $(n/2)$

	a	b	d	
a	0	1		
b	1	0		
c	2			
a	3			
b	4			
c	5			
a	6			

	a	b	d	
a	0	1		
b	1	0		
c	2	1		
a	3			
b	4			
c	5			
a	6			

	a	b	d	
a	0	1		
b	1	0		
c	2	1		
a	3	2		
b	4			
c	5			
a	6			

....

	a	b	d	
a				3
b				2
c				1
a				1
b				2
c				3
a				4

כ"ל מהכיוון השני חישוב : $F^\wedge(j, n/2)$

	a	b	b	c	6
a					5
b					4
c			2	3	
a			1	2	
b			0	1	
c			1	0	

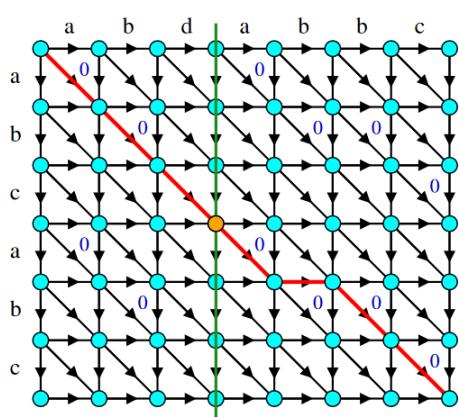
	a	b	b	c	6
a					5
b					4
c					3
a			1	2	
b			0	1	
c			1	0	

	a	b	b	c	6
a					5
b					4
c					3
a			1	2	
b			0	1	
c			1	0	

....

	a	b	b	c	
a	2				
b	3				
c	2				
a	1				
b	2				
c	3				
a	4				

לכן קיבלנו את :



				3
				2
				1
				1
				2
				3
				4

	2			
	3			
	2			
	1			
	2			
	3			
	4			

נמשיל את האלגוריתם

1. לכל j : חשב $F(j, n/2)$ and $F^\wedge(j, n/2)$
2. נמצא את j כך שהסכום $(F(j, n/2) + F^\wedge(j, n/2))$ מינימלי.
3. נdfs את $(j, n/2)$
4. בركורסיה מצא את המסלול האופטימלי עבור : $(m, n) \rightarrow (0,0) \rightarrow (j, n/2) \rightarrow (m, n)$. (חזר ל1)

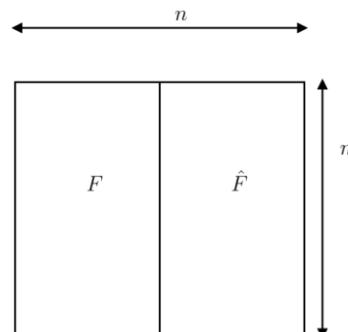
עוצרים את הרקורסיה כאשר יש רק 2 עמודות.

סיבוכיות מקום : $O(m + \log n)$
 ע"מ לנתח את סיבוכיות הזמן נctrar לבודק כמה תאים מלאו במהלך הרצפה.
 קיבלנו שהצומת היא ייחידה ומשקל המסלול הוא 2. הדבר חשוב הוא הדמיון!

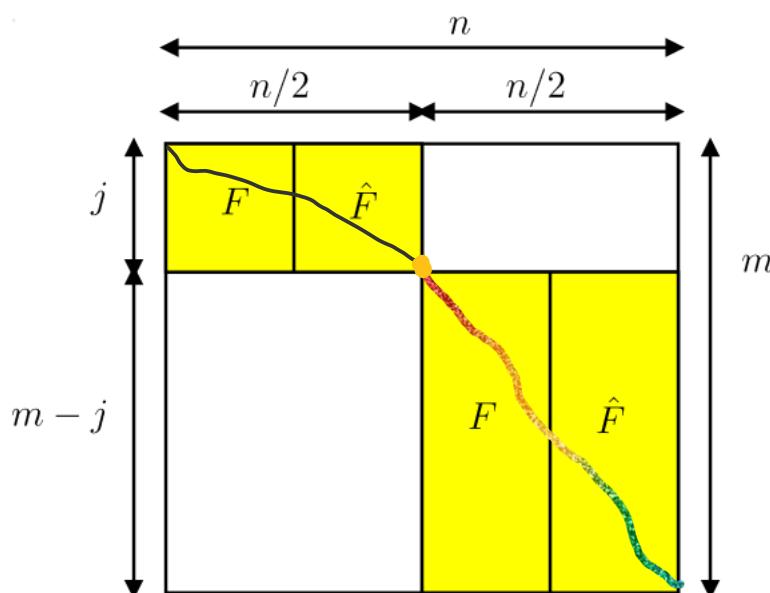
ניתוח :
 בכל שלב מחזיקים קבוע של M זיכרון עבור העמודות בכל שלב. הרקורסיה העומק שלה הוא $\log n$ ולכן צריכים לשמר עת עומק הרקורסיה המירבי הנשמר אוטומטית.

Space complexity $O(m + \log n)$.

כדי לנתח את הזמן נctrar לחשב את הגודל הכללי של תאים בטבלה שמוסלאו ע"י האלגוריתם:
 באיטרציה הראשונה יש טבלה F שמללאים את כל העמודות על העמודה אמצעית, F^\wedge ממלאים מהאמצעית החוצה. יש الوح תאים שאמנו ממלאים.



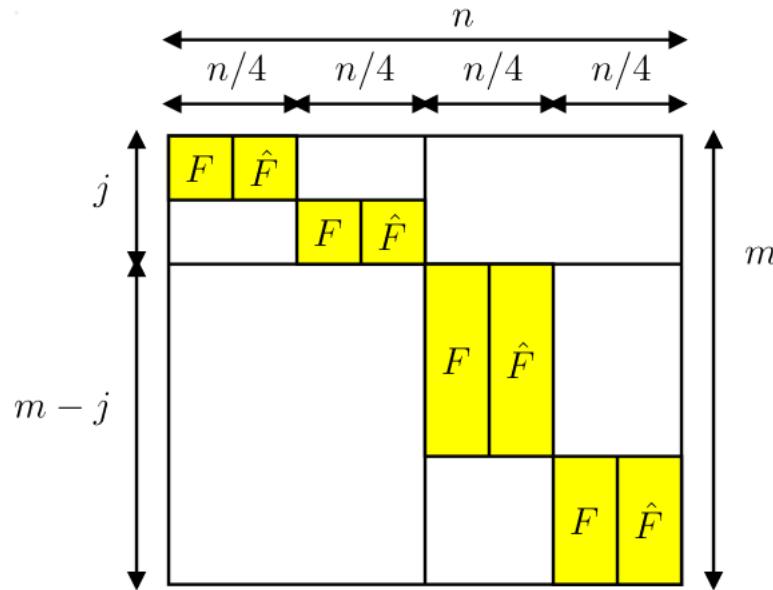
יש 2 קריאות רקורסיביות כלומר אם הסתכמנו על העמודה האמצעית לכן עושים קריאה רקורסיבית למציאת מסלול שמאל ימני. וכך כן עבור כל אחד יד לשות את התהילה הרקורסיבי עבור המסלול השמאלי – כלומר מהכתום לעלה.



זיכרון הטללה : $m * n$

לאחר מכון בקריאה הרקורסיבית הראשונה שטח הטבלאות קטן ל- $\frac{mn}{2}$:

יש לנו 4 קריאות רקורסיביות ולכן שטח הטבלאות בקריאה השנייה:



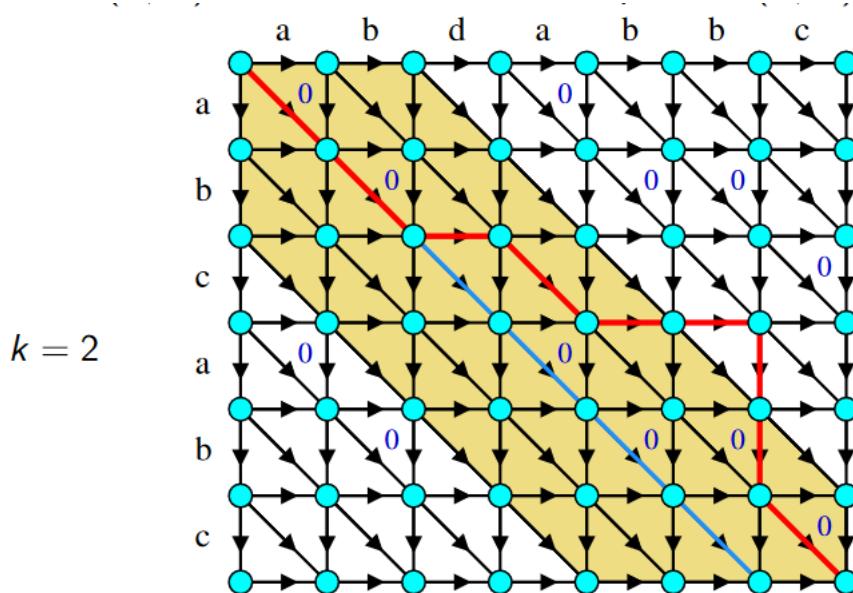
לכן קיבלנו כי יש לנו סדרה חשבונית: $mn + \frac{1}{2}mn + \frac{1}{4}mn + \dots \leq 2mn$

סיבוכיות הזמן לכך לנארית בשטח הטללה כולה!

סיבוכיות זמן $O(mn)$

Comparing similar strings

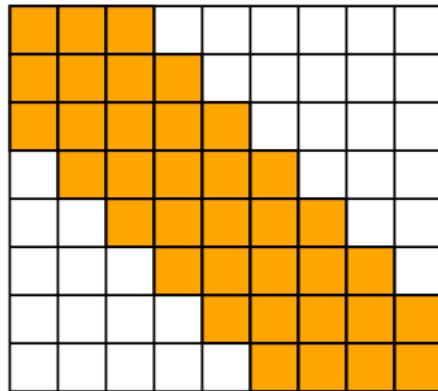
נניח שיש לנו S, T, k כאשר $k \leq k$ כאשר $ED(S, T) \leq k$, ונתנים לנו את זמן העריכה.



המסלול האופטימלי מוכל באזור הצהוב.

במנוחי הטבלה צריך רק לחשב את התאים בטבלה המתאימים לשטח הצהוב – התאים הכתומים.

- We can fill the cells of $F(i,j)$ only for i,j satisfying $|i - j| \leq k$.



זמן ריצה ($O(kn)$) בכל שלב ממלאים $2k+1$ תאים עבור ח.

אלגוריתם מציאת $\text{ED}(S,T)$

אם $k < \text{ED}(S,T)$ – תכוננו דינמי: תחיליה נבדוק אם $|m-n| > k$, ואם כן נחזר אינסופ. נמלא ורק תאים עבור $|i - j| \leq k$.

1. אם $\text{ED}_k(S,T) = \text{ED}(S,T)$ אז $\text{ED}(S,T) \leq k$

2. אם $\text{ED}_k(S,T) > \text{ED}(S,T)$ אז $\text{ED}(S,T) > k$

$\text{ED}_k(S,T)$ – הערך המוחזר מהאלגוריתם.

• $\text{ED}(S,T)$ – מינימום cost עבור alignment של S - T .

אם המרחק האופטימלי גדול מ- k אז הוא יצא מהטוווח.

An $O(dn)$ Algorithm for edit distance

Compute $d = \text{ED}(S, T)$ as follows:

- For $k = 1, 2, 4, 8, \dots$ do:
 - Compute $\text{ED}_k(S, T)$.
 - If $\text{ED}_k(S, T) \leq k$, output $\text{ED}_k(S, T)$ and stop.

באיטרציות שבהן k קטן מדי' וולכן התנאי השני לא מתקיים ולכן ממשיך לעבור להבא, ולכן האלגוריתם עוצר כאשר החזקה הראשונה של 2 גודלה מפ.

1. אם $d > k$ אז $\text{ED}_k(S,T) \geq d > k$

2. אם $d \leq k$ אז $\text{ED}_k(S,T) = d$

. $k = 2^{\log d} \leq d \leq k$

האלגוריתם עוצר כאשר $k = 2^{\log d}$.

זמן ריצה : $O(1 \cdot n + 2 \cdot n + 4 \cdot n + \dots + 2^{\log d} \cdot n) = O(dn)$

נראה אלגוריתם יותר טוב: 81-96.

Compressing F

בכל אלכסון של המטריצה F בערכים בסדר לא יורד.

	a	b	d	a	b	b	c	
a	0	1	2	3	4	5	6	7
b	1	0	1	2	3	4	5	6
c	2	1	0	1	2	3	4	5
a	3	2	1	1	2	3	4	4
b	4	3	2	2	1	2	3	4
c	5	4	3	3	2	1	2	3
a	6	5	4	4	3	2	2	2

	a	b	d	a	b	b	c	
a	0	1	2	3	4	5	6	7
b	1	0	1	2	3	4	5	6
c	2	1	0	1	2	3	4	5
a	3	2	1	1	2	3	4	4
b	4	3	2	2	1	2	3	4
c	5	4	3	3	2	1	2	3
a	6	5	4	4	3	2	2	2

אנו משתמשים בעובדה שאנו יכולים לשמר את כמות המספרים ולא את המספר, למשל: באלכסון של 0 אנו נשמר כי יש 3, 2 אחדות 2 אחד.

$$L[q, e] = \max (\{i : F[i, i + q] = e\} \cup \{-1\})$$

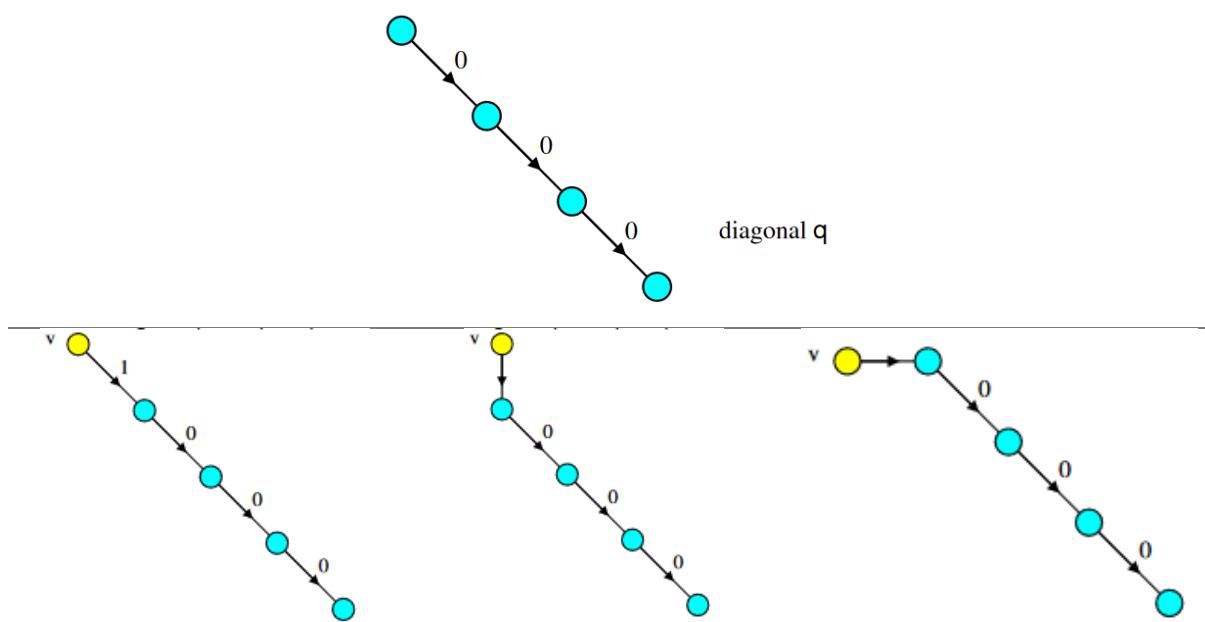
$$\text{לדוגמא: } L[0,0] = 3, L[0,1] = 6, L[0,2] = 7$$

An $O(n+m+d^2)$ algorithm

אלג' משופר למציאת $\text{ED}(S,T)$ אם $k < \text{ED}(S,T)$ נבנה generalized suffix tree עבור $S-T$ ונכין אותו לשאלות LCA . לכל $k = e <= q <= k$ נחשב את $[q, e]$ שזהו ערך השורה המלא באלכסון ה- q שבתא שלה יש e , ונחזר את e עבורו $m = m - n$. חישוב ערכי $[q, e]$ לפי הנוסחה $(n \dots m], T[y+1 \dots m], T[y+1 \dots m], \text{LCP}(S[y+q+1 \dots m], T[y+1 \dots m]) + 1$, כאשר

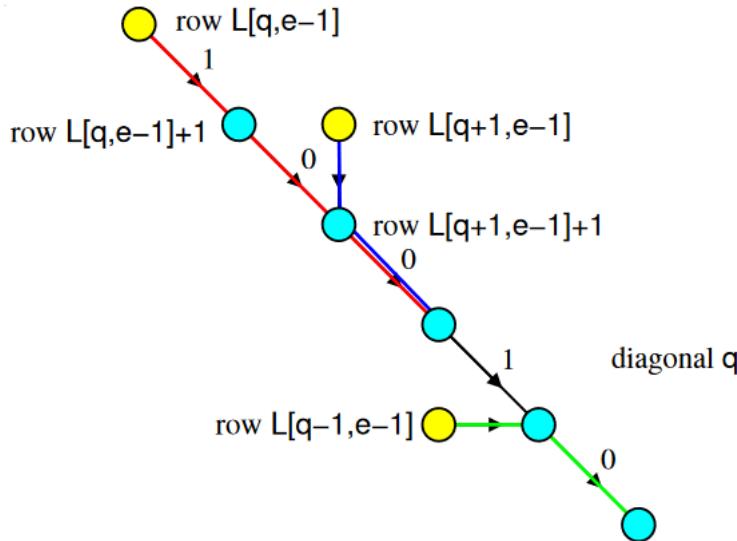
$$y = \max\{L[q, e-1]+1, L[q+1, e-1]+1, \dots, L[q-1, e-1]\}$$

נניח כי קיים מסלול ממושך מ- v המתחילה באלכסון q ובשורה $[e, L]$, יהי (w, v) הקשת האחורונה שמשקלה 1 ב المسلול. w באלכסון של q , v הוא על האלכסון $1 + q$ or $q - 1$.



נבדוק את 3 האפשרויות למיקום v :

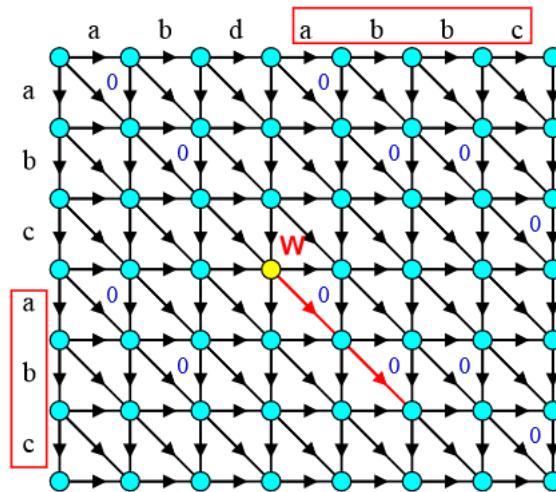
1. על האלכסון q כלומר $v = (L[q, e-1], L[q, e-1] + q)$ (האידום)
2. על האלכסון $q+1$ כלומר $v = (L[q+1, e-1], L[q+1, e-1] + q+1)$ (הכחול)
3. על האלכסון q כלומר $v = (L[q-1, e-1], L[q-1, e-1] + q-1)$ (הירוק)



לא נבדוק את הערכים האחרים עבורה.

$$s = \max\{L[q, e-1] + 1, L[q+1, e-1] + 1, L[q-1, e-1] + 1\}$$

לאחר שמצאנו את w אנו צריכים למצוא את מספר קשთות במשקל 0 הרצופות על האלכסון q לאחר w . כדי למצוא את מספר הקשთות במשקל 0 הרצופות נבצע שאלת LCP על הסופיות של S ו- T :



$$L[q, e] = s + LCP(S[s+q+1..n], T[s+1..m])$$

בהתנאי k כך ש $ED(S, T) \leq k$

1. נבנה עץ סיפות מוכלל עבור T ונהפוך אותו ל-LCA

2. נחשב $L[q, e]$ לכל $-k \leq q \leq k$ and $0 \leq e \leq k$

3. נוציא את הערך e עבורו $L[n-m, e] = m$

סיבוכיות זמן: $O(n+m+k^2)$

הרצאה 10 matrices DIST

T' - Alignment S' ו- T' הן הסרת כל המקפים ('-') מ-S' ומ-T' נותנת את S ו-T. אין זר שישי מקיף גם ב-[i] S' וגם ב-[i] T'.

תהי $R \rightarrow (\{-\} \cup \Sigma) \times (\{-\} \cup \Sigma)$: δ פונקציית תוצאה. יהי T, S 2 מחוזות כך ש' T' , S' הן העמדת S ו-T.

התוצאה(score) של T', S' הוא $\sum_{i=1}^{|S'|} (\delta(S'[i], T'[i])$ הוא סכום ערכי פונקציית הדמיון על זוגות האותיות המקבילות של S' ו-T'.

לדוגמא:

Define

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ -2 & \text{if } x = c, y = b \\ -1 & \text{otherwise} \end{cases}$$

Let $S = abcabbd$, $T = abbabb$

$\begin{array}{r} \text{abcabbd} \\ \text{abbabb-} \end{array}$	$\begin{array}{r} \text{abcabbd} \\ \text{abba-bb} \end{array}$
Score: 5 - 2 - 1	4 - 2 - 2

LCS - פונקציה המחזיר 1 עבור שתי אותיות זהות ו-0 עבור שתי אותיות שונות:

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

דוגמא: העמדה האופטימלית היא: $abbabb-$, Let $S = abcabbd, T = abbabb$

התוצאה היא 5. יש 5 אותיות זהות. המחרוזת ababb נקראת LCS (longest common subsequence) של S.

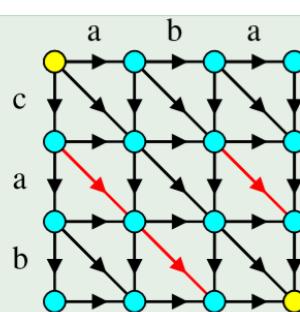
(Longest common subsequence) LCS - של S ו-T זהה תת-הסדרה הארוכה ביותר שנitin' לקבל גם מ-S ו-T ע"י מכיקת אותיות

הגדרה: תהי δ פונקציית scoring, בהינתן T, S נמצאת העמדה בין S ו-T עם מקסימום (או מנימום).

The optimal alignment problem זהה למציאת מסלול משקל מקסימלי ב-alignment graph. משקל הצלעות מ(i, j) הינו:

$$w_{i,j}^{\rightarrow} = \delta(-, T[j]) \quad w_{i,j}^{\downarrow} = \delta(S[i], -) \quad w_{i,j}^{\searrow} = \delta(S[i], T[j])$$

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ -1 & \text{otherwise} \end{cases}$$



את הבעיה ניתן לפתור בעזרת תכנון דינامي, יהיו $F(i, j) = \text{משקל מקסימלי של מסלול בין } (j, i) \rightarrow (0,0)$.

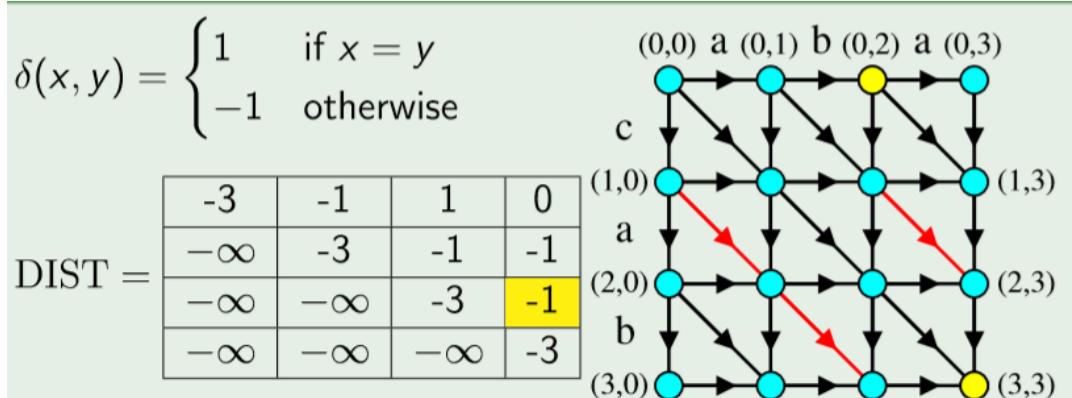
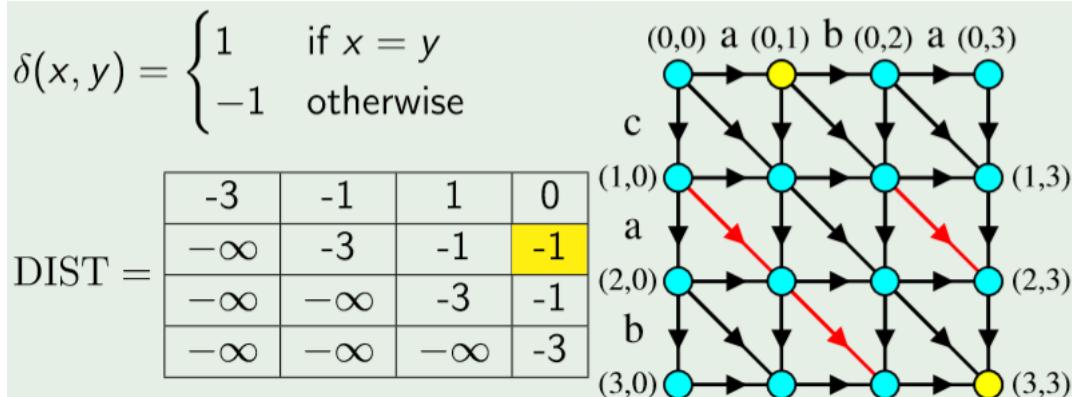
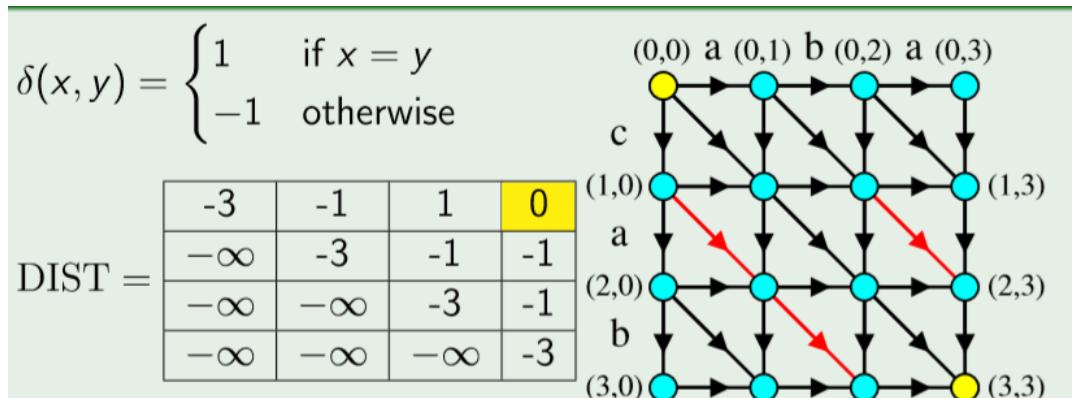
$$F(i, j) = \max \left\{ \begin{array}{l} F(i - 1, j) + w_{i,j}^{\downarrow}, \\ F(i, j - 1) + w_{i,j}^{\rightarrow}, \\ F(i - 1, j - 1) + w_{i,j}^{\searrow} \end{array} \right\}$$

. משקל מסלול מקסימלי $(n, 0) \rightarrow (m, n)$ הוא $F(m, n) \rightarrow (0, 0)$
סיבוכיות זמן $O(mn)$.

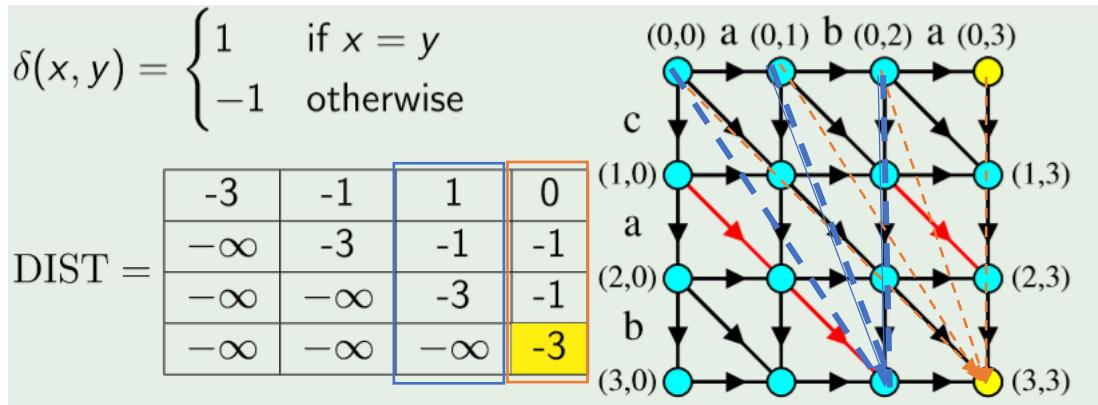
הגדרה: $DIST(i, j) = \text{משקל המקסימלי של מסלול מ}(i, 0) \text{ ל}(j, m)$.

$DIST_{ij}$ - מטריצה שבתא ה- i,j , שלה נמצא משקל הדרך המקסימלית מצומת $(i, 0)$ לצומת (j, m) .

אנו לוקחים מרחוקים כל פעם מהקודקודיםعلילונים לאחד הקודקודים התחתיים, העמודות לפי קודקודים למטה



אלגוריתמים למחוזות / סיכום מאות אורי שביט



Monge property

הגדירה: מטריצה M היא Monge אם לכל 4 תאים בה מתקיים $k_2 + k_3 \leq k_1 + k_4$

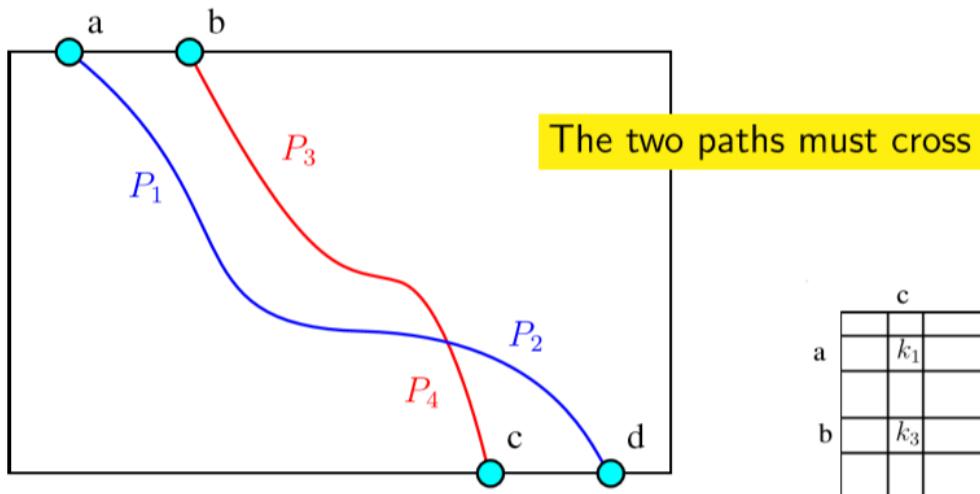
2	3	1
1	4	2
2	5	4

	c	d
a	k_1	k_2
b	k_3	k_4

דוגמא:

- מטריצה M שלכל אינדקסי שורה $a < b$ ואינדקסי عمودה $c < d$ olla מתקיים כי
 $M[a,c] + M[b,d] \geq M[a,d] + M[b,c]$

DIST is Monge



$$\text{DIST}(a, d) = w(P_1) + w(P_2), \text{DIST}(b, c) = w(P_3) + w(P_4).$$

$$\text{DIST}(a, c) \geq w(P_1) + w(P_4), \text{DIST}(b, d) \geq w(P_2) + w(P_3).$$

$$\implies \text{DIST}(a, c) + \text{DIST}(b, d) \geq \text{DIST}(a, d) + \text{DIST}(b, c).$$

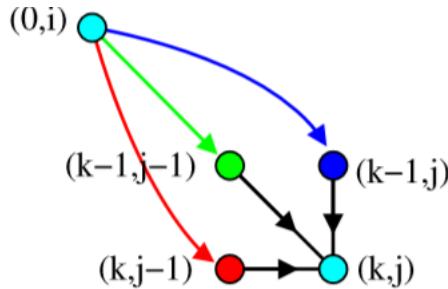
1. DIST זה המסלול המקיים ולכן \geq מתקיים

$$\text{DIST}(a, c) + \text{DIST}(b, d) \geq w(P_1) + w(P_4) + w(P_2) + w(P_3) = \text{DIST}(a, c) + \text{DIST}(b, d)$$



ולכן DIST היא Monge

הגדרה $DIST_k = \text{המשקל המינימלי של מסלול } (0,0) \text{ ל } (j,k)$. האלגוריתם ירכיב $DIST_1, \dots, DIST_m$



עובדה:

$$DIST_k(i,j) = \max \left\{ \begin{array}{l} DIST_k(i,j-1) + w_{k,j}^{\rightarrow} \\ DIST_{k-1}(i,j-1) + w_{k,j}^{\nwarrow} \\ DIST_{k-1}(i,j) + w_{k,j}^{\downarrow} \end{array} \right\}$$

where $w_{k,j}^{\rightarrow}, w_{k,j}^{\nwarrow}, w_{k,j}^{\downarrow}$ are the weights of the edges entering (k,j) .

בנייה מטריצת DIST - בניית מטריצת $DIST_k$ לכל $k=1 \dots m$ וnochzir את $DIST_m$. מתקיים:

$$DIST_k(i,j) = \max \{ DIST_k(i,j-1) + w_{k,j}^{\rightarrow}, DIST_{k-1}(i,j-1) + w_{k,j}^{\nwarrow}, DIST_{k-1}(i,j) + w_{k,j}^{\downarrow} \} :=$$

(הסימונים אחרי $\rightarrow, \nwarrow, \downarrow$ הם חיצים ימינה, ימינה-למטה ולמטה, המשך בשורה הבאה)

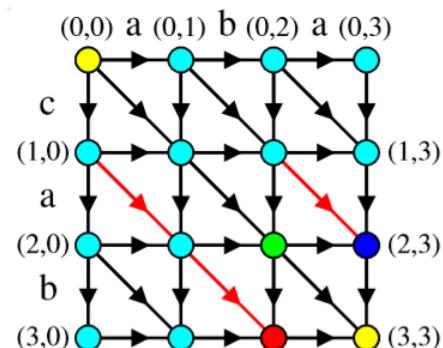
$$:= \max \{ R(i) + c_r, G(i) + c_g, B(i) + c_b \}$$

נחשב את $DIST_1$ מטענה 1 ע"י חישושים ביןאריים. כל עמודה j של $DIST_k$ נשמרת בערך $B_{j,k}$ עם ה

עילים
, וכך גם משקלן הקשחות בדרכם לעלה i יהיה $[j][i]$. יוצרים דרכים ל- $DIST_k$ מ- $DIST_{k-1}$. כל צומת j בדרכים הללו יורדים לבן שלו, ואת הבן שלו ירדנו אליו לא נבנה. אלא נקבע לעצם שכבר קיימים, והקשת אל העץ המוצבע תהיה במשקל של הדרך ל- j בעץ המקורי ועוד $c_r/c_g/c_b$. כל צומת יוכל לדרכן ממנו לעלה הימני ביותר בתת-העץ של בנו השמאלי, ואז בחישוב הבינארי של j ונחילה מהשורשים של $DIST_{k-1}$ ב- $B_{j-1,k}$ ונרדף לפיו ההשוואות. כנ"ל עבור i .

-2	0	-1	-1
$-\infty$	-2	-2	0
$-\infty$	$-\infty$	-2	0
$-\infty$	$-\infty$	$-\infty$	-2

-3	-1	1	0
$-\infty$	-3	-1	-1
$-\infty$	$-\infty$	-3	-1
$-\infty$	$-\infty$	$-\infty$	-3

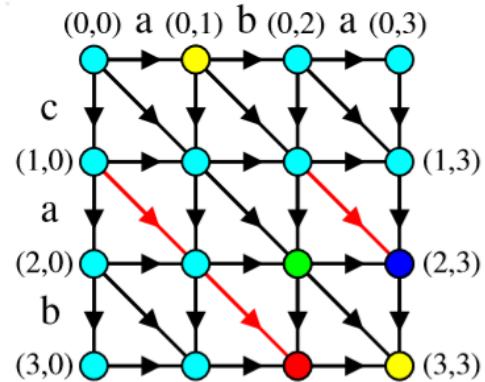


$$DIST_3(\cdot, 3) = \max \left\{ \begin{pmatrix} 1 \\ -1 \\ -3 \\ -\infty \end{pmatrix} - 1, \begin{pmatrix} -1 \\ -2 \\ -2 \\ -\infty \end{pmatrix} - 1, \begin{pmatrix} -1 \\ 0 \\ 0 \\ -2 \end{pmatrix} - 1 \right\}$$

$$= \max(R + c_r, G + c_g, B + c_b)$$

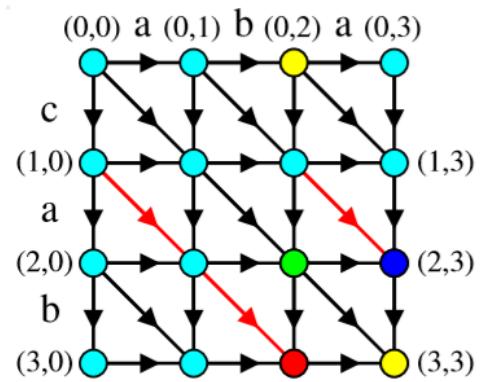
	-2	0	-1	-1
DIST ₂ =	$-\infty$	-2	-2	0
	$-\infty$	$-\infty$	-2	0
	$-\infty$	$-\infty$	$-\infty$	-2

	-3	-1	1	0
DIST ₃ =	$-\infty$	-3	-1	-1
	$-\infty$	$-\infty$	-3	-1
	$-\infty$	$-\infty$	$-\infty$	-3



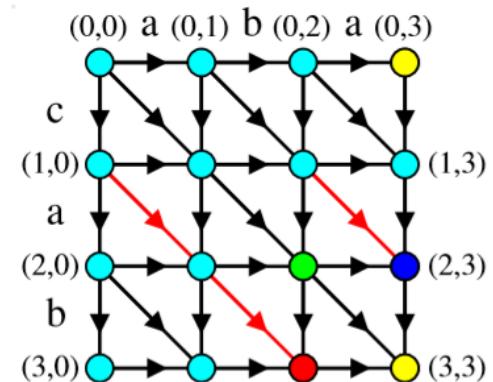
	-2	0	-1	-1
DIST ₂ =	$-\infty$	-2	-2	0
	$-\infty$	$-\infty$	-2	0
	$-\infty$	$-\infty$	$-\infty$	-2

	-3	-1	1	0
DIST ₃ =	$-\infty$	-3	-1	-1
	$-\infty$	$-\infty$	-3	-1
	$-\infty$	$-\infty$	$-\infty$	-3



	-2	0	-1	-1
DIST ₂ =	$-\infty$	-2	-2	0
	$-\infty$	$-\infty$	-2	0
	$-\infty$	$-\infty$	$-\infty$	-2

	-3	-1	1	0
DIST ₃ =	$-\infty$	-3	-1	-1
	$-\infty$	$-\infty$	-3	-1
	$-\infty$	$-\infty$	$-\infty$	-3



$$DIST_k(i,j) = \max(R(i) + c_R, G(i) + c_G, B(i) + c_B) \text{ where}$$

$$R(i) = DIST_k(i, j - 1), c_R = w_{i,k}^{\rightarrow}$$

$$G(i) = DIST_{k-1}(i, j - 1), c_G = w_{k,j}^{\searrow}$$

$$B(i) = DIST_{k-1}(i, j), c_B = w_{i,k}^{\downarrow}$$

המסלולים לעיל הם מסלולים לנקודות אדומנה, ירוקה וכחולה. כאשר $(0, i) \rightarrow (2, j)$ כאשר $i, j \in \{0, 1, 2, 3\}$ — אם אין מסלול כלומר צריך לחזור אחורה. (המטריצה והגרף הפוכים: העמודות של המטריצה מ-4 ל-0

טענה: קיימים אידקושים $i_1 \leq i_2$ כך ש:

$$DIST_k(i, j) = \begin{cases} R(i) + c_R & \text{if } i \leq i_1 \\ G(i) + c_G & \text{if } i_1 < i \leq i_2 \\ B(i) + c_B & \text{if } i_2 < i \end{cases}$$

הוכחה:

$$\begin{aligned}
DIST_k(i,j) &= \max(R(i) + c_R, G(i) + c_G, B(i) + c_B) = \\
&= \max(R(i) - G(i) + c_R, c_G, B(i) - G(i) + c_B) + G(i) \\
&= \max(\text{diff}R(i) + c_R, c_G, \text{diff}C(i) + c_B) + G(i)
\end{aligned}$$

where

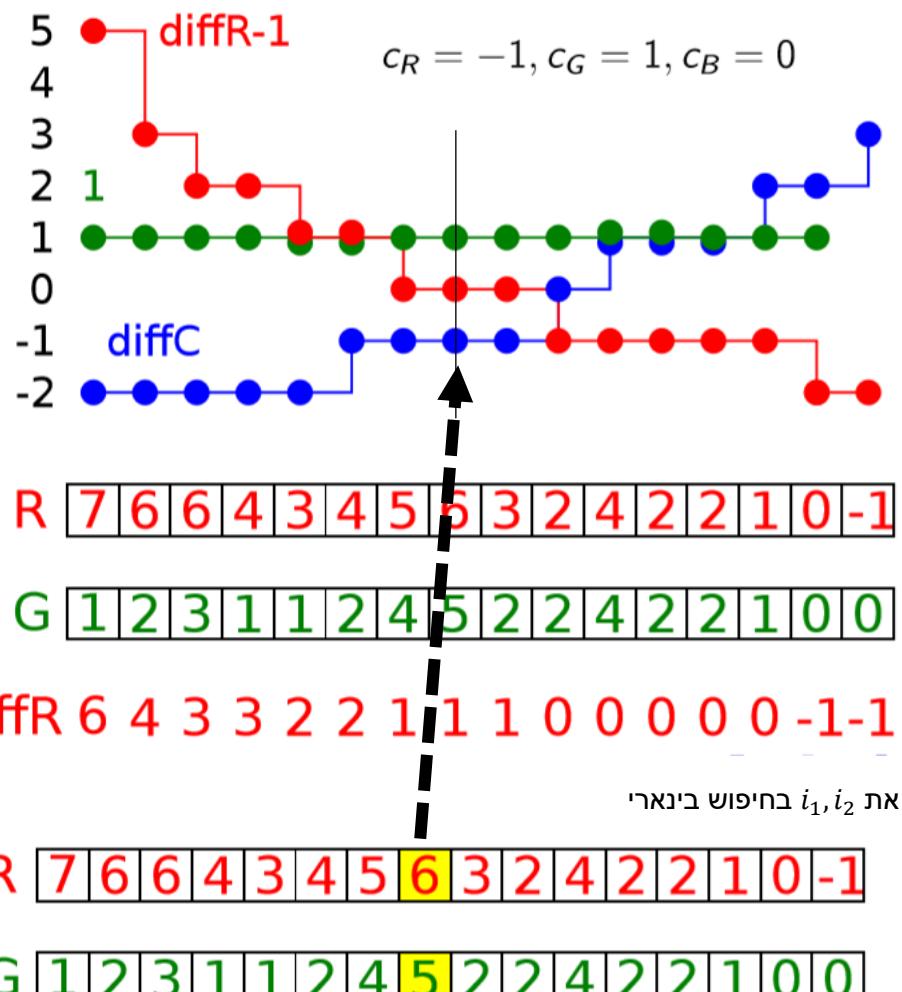
$$diffR(i) = R(i) - G(i) = DIST_k(i,j-1) - DIST_{k-1}(i,j-1)$$

$$diffC(i) = B(i) - G(i) = DIST_{k-1}(i,j) - DIST_{k-1}(i,j-1)$$

למה **1:** $diffC$ מונוטונית עולה.

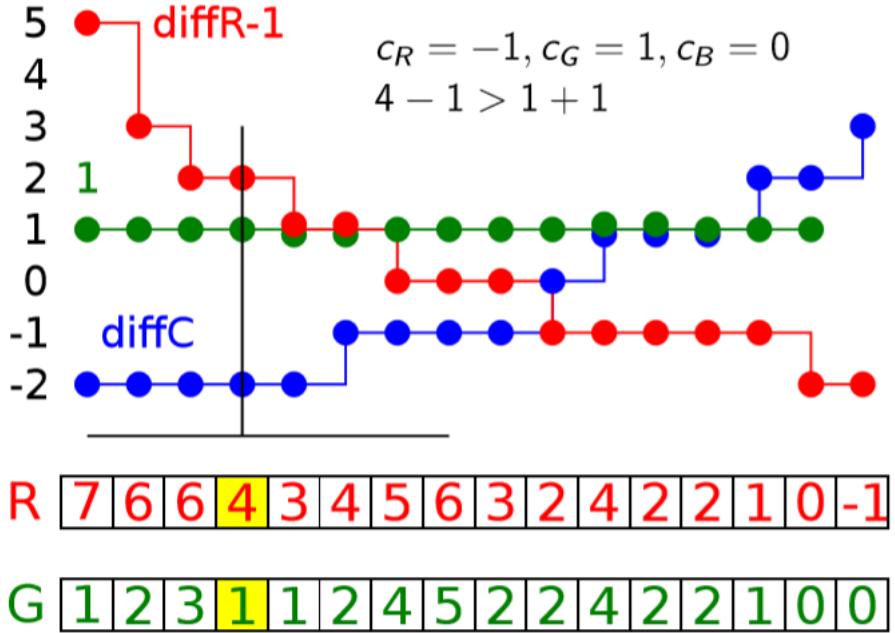
למה 2: $diffR$ מונוטונית יורדת.

דוגמאות



$$c_R = -1, c_G = 1, c_B = 0$$

$$6 - 1 < 5 + 1$$



diffR 6 4 3 3 2 2 1 1 1 0 0 0 0 -1 -1

הוכחת למה 1: נניח ש $i < j$ אנו רוצים להראות :

$$DIST_{k-1}(i, j) - DIST_{k-1}(i, j - 1) \leq DIST_{k-1}(i', j) - DIST_{k-1}(i', j - 1)$$

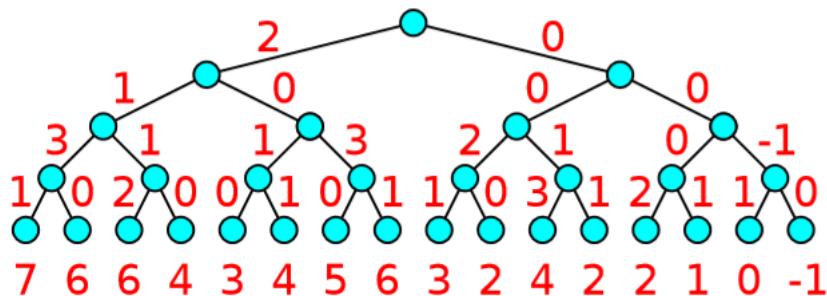
האי שווין נובע מתכונת Monge של $DIST_{k-1}$.

אחסון $DIST$

על מנת לאחסן את העמודות של $DIST_k$ באמצעות מערכים : לבנות עמודה אחת נדרש $O(n)$ זמן, לבנות את כל העמודות יקח $O(n^2)$ זמן ולבניית כל טבלת $DIST$ יקח $O(mn^2)$ זמן.

ע"י אחסון עמודות $DIST$ בעץ, הבנייה של עמודה אחת תיקח $O(\log n)$.

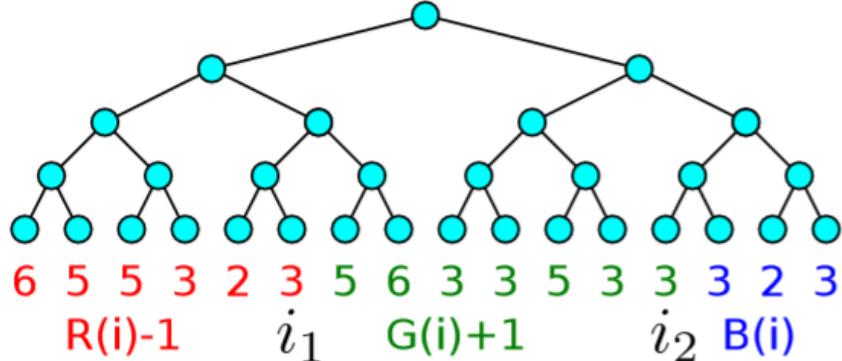
.אחסון העמודה j של $DIST_k$ בעץ $B_{j,k}$ עם n עליים. סכום כל הקשתות על מסלול מעלה i בעץ :



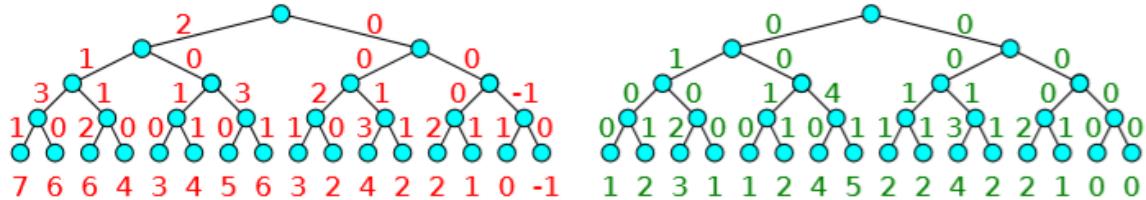
.נניח שאנו יכולים לחשב את כל העצים עבור כל העמודות של $DIST_{k-1}$ והעמודות $1, \dots, j-1$ של $DIST_k$

.אנו יכולים לבנות את העץ $B_{j,k}$ עבור העמודה j של $DIST_k$ ע"י שימוש ב $B_{j-1,k}, B_{j-1,k-1}, B_{j,k-1}$

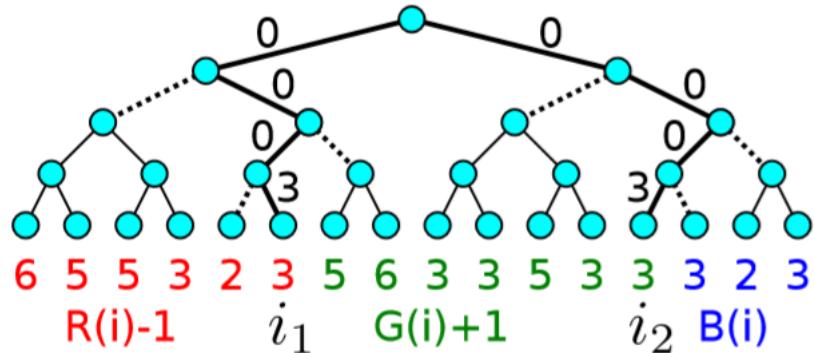
העץ עבורי $B_{j,k}$ העמודה ה j של $DIST_k$



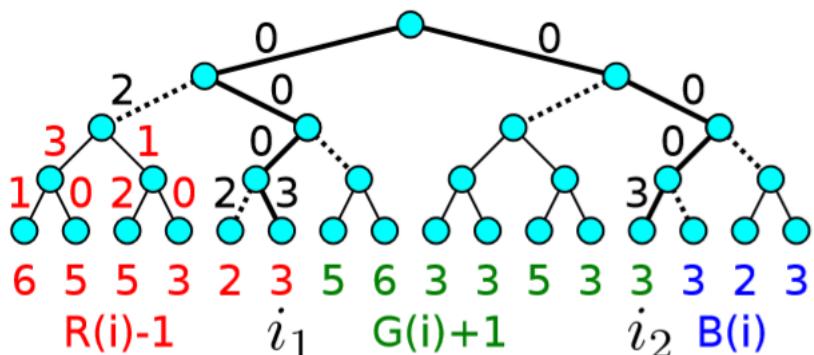
העצים עבורי R ו- G :



יצור מסלול לעלי i_2 , i_1 עם הקשתות שמשקלן 0, נצפה שעבור הקשתה האחורונה שמשקללה הוא $score$ של העלה. בן שלא נמצא על המסלול יוחלף במצביע לעץ ישן. מה משקל הקשתות הללו?



משקל של כל קשת הנכנתת לו הוא משקל של מסלול בין השורש לו בעץ המקורי $+ c_R/c_G/c_B$.



בהתוכן: בניית עצ של עמודה יחידה של $DIST_k$ לוגריתם ($\log n$), מה זמן חישוב i_1, i_2 ?

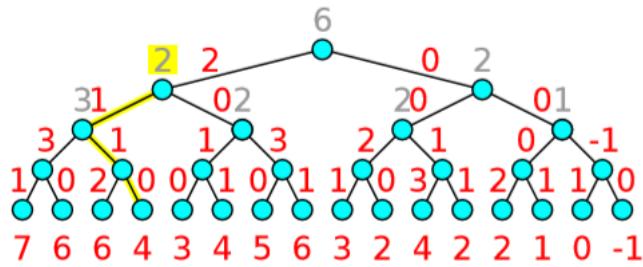
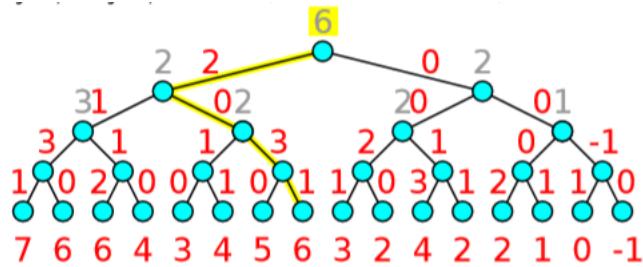
זמן הוא ($n^2 \log n$) כיון שכניסה לערך אחד של $DIST$ לוגריתם ($\log n$).

Weights of paths to middle leaf

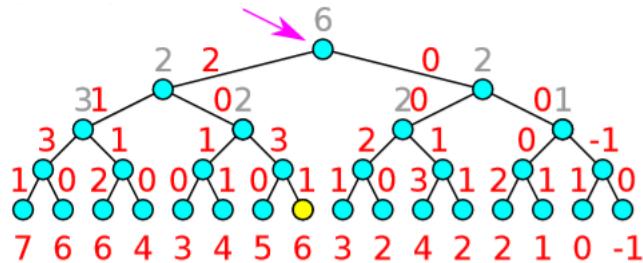
עבור כל אחד מהצמתים הפנימיים v , נאחסן את משקל המסלול מה לעלה הכי ימני בתת עץ השמאלי של v . כאשר מבצעים חישוב i החול מהשורש של העצים $B_{j-1,k}, B_{j-1,k+1}, \dots, B_j$ והולכים למטה לפי ההשוואה.

R 7 6 6 4 3 4 5 6 3 2 4 2 2 1 0-1

G 1 2 3 1 1 2 4 5 2 2 4 2 2 1 0 0



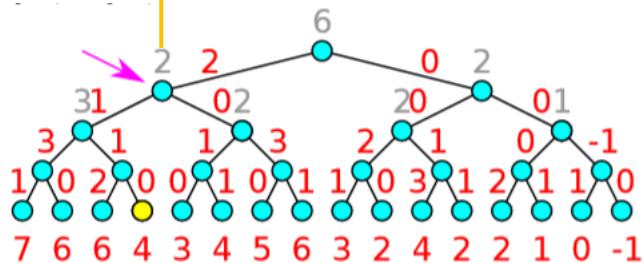
משקל המסלול מ- 6 עד עז השמאלי של ימי בתה הצעירה



R

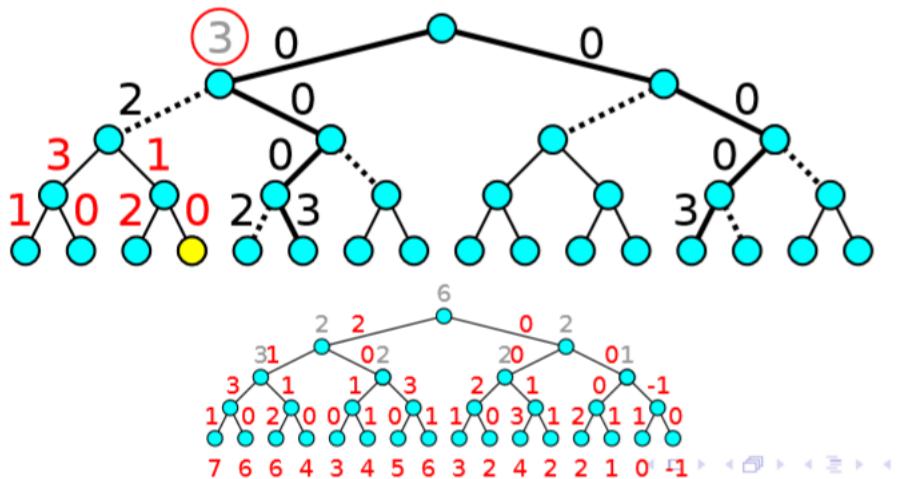
7	6	6	4	3	4	5	6	3	2	4	2	2	1	0	-1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----

G 1 2 3 4 1 1 2 4 5 2 2 4 2 2 1 0 0



עדכון משקל מסלולים:

כאשר אנו בונים עץ חדש אנו צריכים לחשב את משקל המסלולים, ניתן לעשות זאת ע"י שימוש במשקל מסלולים הקיימים בעץ. בדוגמא למטה, משקל הקשת בין A ליד השמאלי שלו הוא 1 בעץ של R ו-2 בעץ החדש משקל המסלול מושג עליה בעץ החדש הוא 3.



ניתוח סיבוכיות:

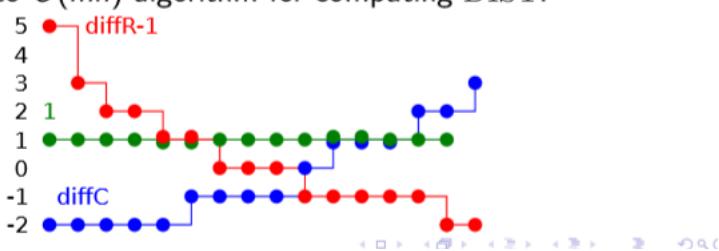
הרכבת העץ של עמודה ייחידה עבור $DIST_k$ לוקח ($\log n$)

הרכבת כל העמודות של $DIST_k$ לוקח $O(n \log n)$

הרכבת DIS כלו לוקחת $O(mn \log n)$.

Discrete scores

- Suppose that the weights of diagonal edges are from $\{0, 1, \dots, C\}$ for some constant C , and the weights of horizontal and vertical edges is 0.
 - In this case the diffR and diffC functions are integers from $\{0, \dots, C\}$.
 - Instead of storing the columns of the DIST matrices, we store the diffR and diffC functions for every edge in the graph. One function can be represented in $O(C)$ space, by storing the jump points.
 - This leads to $O(mn)$ algorithm for computing DIST.



- The DIST matrix of a grid graph can be built in $O(mn \log n)$ time.
 - For discrete scores, the DIST matrix can be built in $O(mn)$ time.

אלגוריתמים למחוזות / סיכום מאות אורי שביט

LCS - של S ו-T זהה תת-הסדרה הארוכה ביותר שניתן לקבל גם מ-S וגם מ-T ע"י מחיקת אותיות.

LCS score - פונקציה המחזיר 1 עבור שתי אותיות זהות ו-0 עבור שתי אותיות שונות.

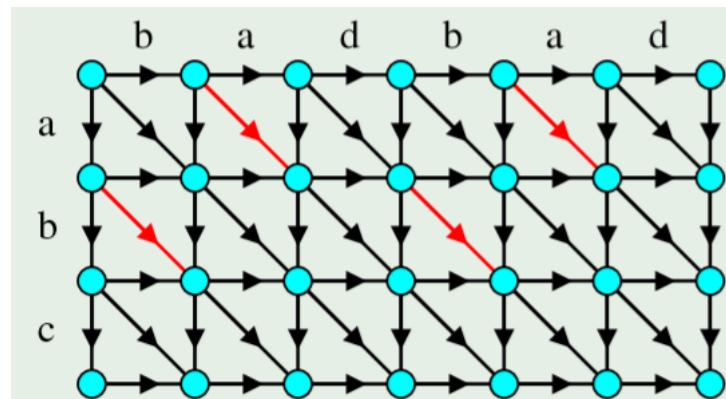
מציאת LCS של A ו-B - בניית מטריצת DIST של A ו-B (שרשור) לפי LCS score נמצא את maksimum מבין ($a+i, b+j$) DIST $[i, j]$ עבור $i=0 \dots n-1$ ו- $j=0 \dots m-1$.

הבעיה: בהינתן 2 מחוזות A ו-B נחשב את LCS shift של A עם B' של B. לדוגמה:

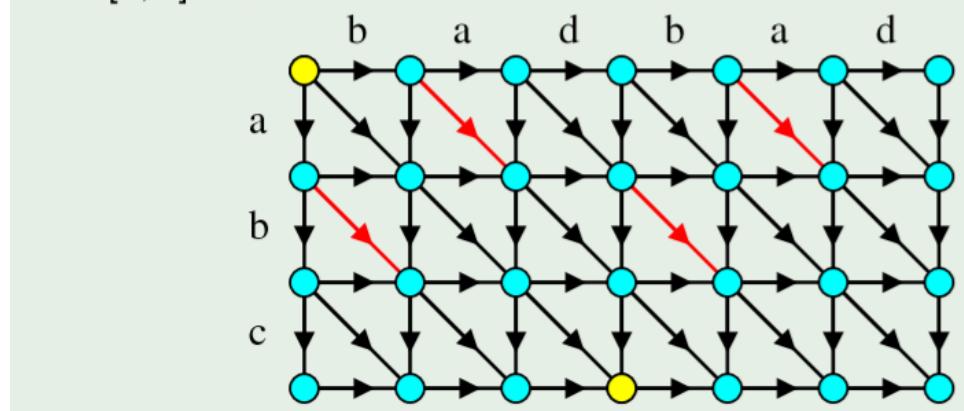
For $A = abc$ and $B = bad$, the LCS ab ($B' = adb (= B \gg 1)$)

פתרון: נחשב את מטריצת DIST של A ו- B^* ע"י LCS score נמצא את maksimum מבין ($i, i+n$) DIST $(i, i+n)$ עבור $i = 0, \dots, n-1$

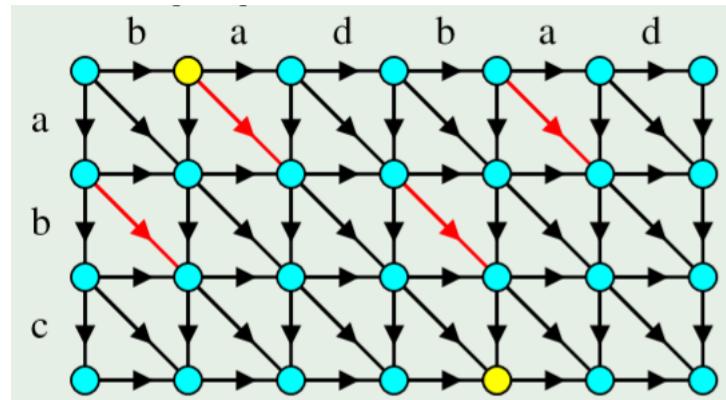
לדוגמא עבור $A = abc$ and $B = bad$



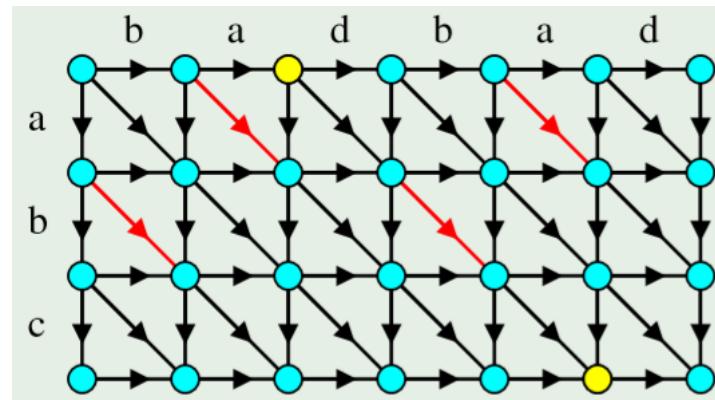
$DIST[0, 3] = 1$,



$DIST[1, 4] = 2$



$$DIST[2,5] = 1$$



סיכום הפתרון:

1. נחשב את מטריצת DIST של A ו-B^{*} ע"י LCS score. נמצא את המקסימום מבין $(a+i)(b+j)$ עבור $i=0, \dots, m-1$, $j=0, \dots, n-1$.
2. LCS תחשב על הליכה לאחרו על טבלת $DIST_k$.

סיבוכיות זמן $O(mn)$.

הרצאה 12

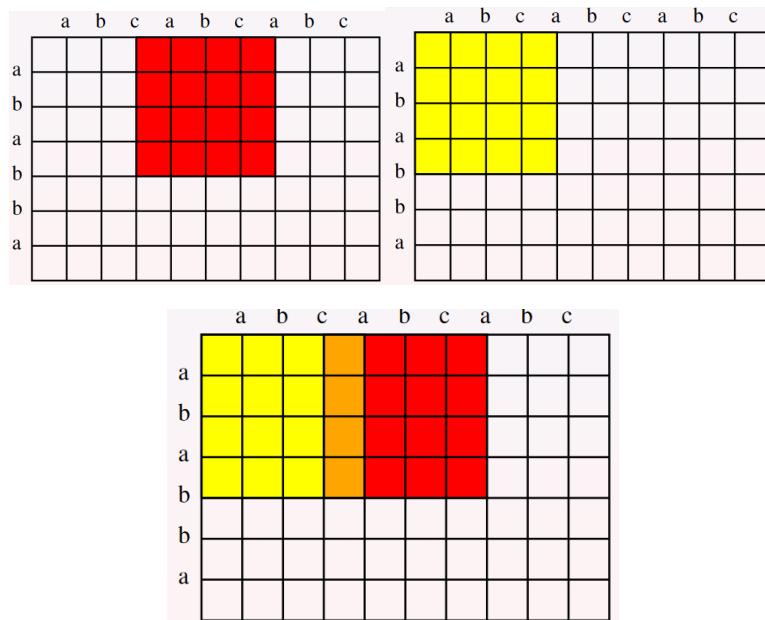
$AnO(nm/\log 2n)$ algorithm

נזכיר לאלגוריתם של ED שהישב טבלה F כאשר ($j \dots i$)

	a	b	c	a	b	c	a	b	c	
a	0	1	2	3	4	5	6	7	8	9
b	1	0	1	2	3	4	5	6	7	8
a	2	1	0	1	2	3	4	5	6	7
b	3	2	1	1	1	2	3	4	5	6
b	4	3	2	2	2	1	2	3	4	5
a	5	4	3	3	3	2	2	3	3	4
a	6	5	4	4	3	3	3	2	3	4

האלגוריתם המהיר מבוסס על חלוקה של הטבלה לבלוקים.

הפרדה של F לכל בлокים כאשר בלוקים סמוכים חולקים שורה או עמודה של תאים.



Zones of a block

בכל בלוק נחשב את ערכי התאים באזורי $D/B/C/A$ של הבלוק.

	a	b	c	a	b	c	a	b	c
a									
b									
a									
b									
b									
a									
a									
b									
a									

A 2x2 block labeled A is highlighted in yellow. To its right is a 2x2 block labeled B. Below A is a 2x2 block labeled C. To the right of B is a 2x2 block labeled D. Below C is a 2x2 block labeled E.

אלגוריתמים למחוזות / סיכום מאת אורי שביט

כל בלוק יחולק ל 5 חלקים.

הרעין של האלגוריתם הוא מילוי הטבלה ללא אזורי E של הטבלה:

	a	b	c	a	b	c	a	b	c	
a	0	1	2	3	4	5	6	7	8	9
b	1			2			5			8
a	2			1			4			7
b	3	2	1	1	1	2	3	4	5	6
b	4			2			2			5
a	5			3			2			4
a	6	5	4	4	3	3	3	2	3	4

מה שיחסור את זמן הריצה.

מעבר על הבלוקים משמאלי לימין ומלמעלה למטה, תחיליה נמלא:

	a	b	c	a	b	c	a	b	c	
a	0	1	2	3	4	5	6	7	8	9
b	1									
a	2									
b	3									
b	4									
a	5									
a	6									

לאחר מכן לפי בלוקים :

	a	b	c	a	b	c	a	b	c	
a	0	1	2	3	4	5	6	7	8	9
b	1			2						
a	2			1						
b	3	2	1	1						
b	4									
a	5									
a	6									

	a	b	c	a	b	c	a	b	c	
a	0	1	2	3	4	5	6	7	8	9
b	1			2						
a	2			1						
b	3	2	1	1						
b	4									
a	5									
a	6	5	4	4						

...

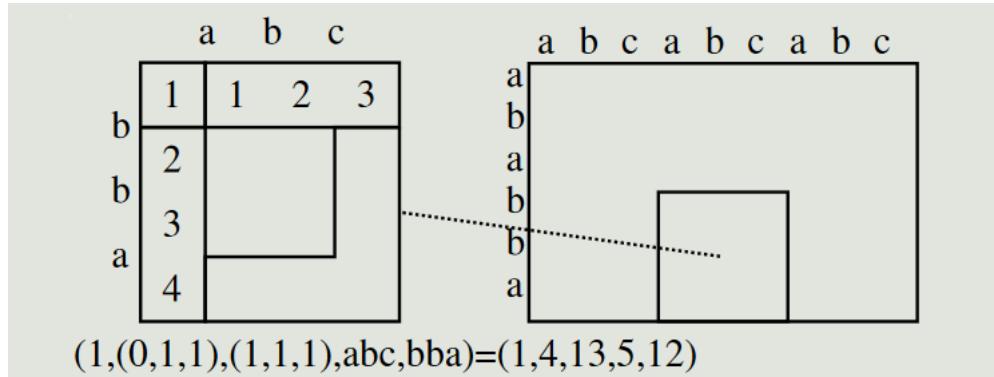
אם נסתכל על הטבלה אם מסתכלים על 2 ערכים סמוכים ההפרש ביניהם הוא 0 או 1 או -1.

לכן אנו יכולים לקודד בלוק עם מעט זיכרון :

אלגוריתמים למחוזות / סיכום מאת אורי שביט

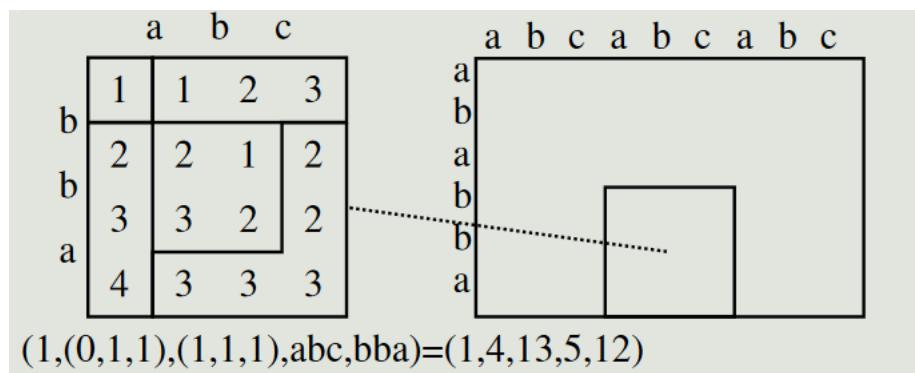
אנו רוצים לקודד את הערכים עם מעט זיכרון, נקודד ע"י רישום הערך באזור A לאחר מכן נרשום את הערכים של האיברים משמאלו. נרשום את ההפרש שמעל עבור C, עבור B ערך משמאלי.

נקרא לוקטור קידוד הכניסה של הבלוק, כשאנו מטפלים בבלוק אנו יודעים את הקידוד הנ"ל. במקום לשמוך את הווקטורים הנ"ל אנו נקודד אותם בעזרת מספרים כלומר כמספר בסיס 3 כולל 1.



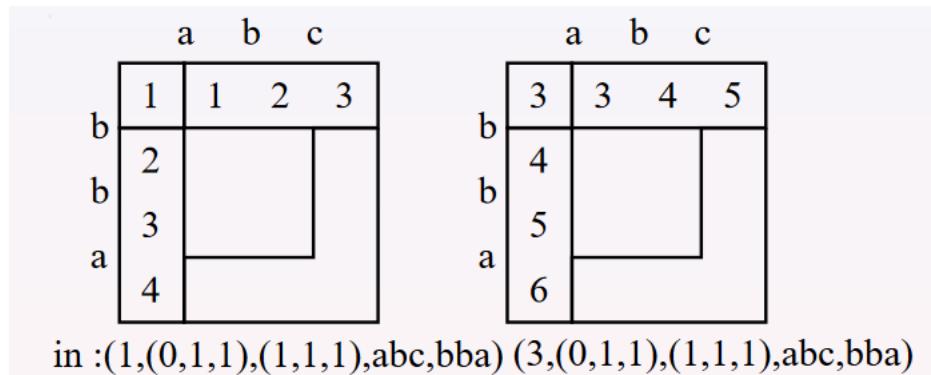
גם למחוזות ניתן להסתכל על המחרוזות כמספרים בסיס הא"ב.

תהליך עד שקי 18



אנו רוצים לקודד את הערכים של השורה והעמודה של D.

אם קידודים זהים פרט לאיבר הראשון אז אם נמלא את כל הערכים בתחום הבלוק ההפרש תמיד יהיה אותו הפרש : ההפרש 2 ולכן בשורה הראשונה כל האיברים בהפרש 2 וזה גובל בכך שהקידוד זהה.



чисוב האיבר הפנימי 2 – לוקחים את המינימום מסביב כלומר 2
כナルחישוב הפנימי של 4 – זהה למינימום של השלשה מסביב.

לכן קידוד היציאה של הבלוקים יהיה זהה פרט לאיבר הראשון שם ההפרש יהיה 2.

האלגוריתם ישתמש בעובדה שאם יש קידוד זהה פרט לאיבר הראשון נוכל ליצור את כל הקידודים האפשריים וליצור בלוק מטאים, כל אפשרות כזו ניתן למלא את הערכים בתוך הבלוק ע"י האלגוריתם הנאיבי ולחשב את קידוד היציאה.

a a a				a a a				b a a			
a	0	0 0 0		a	0	1 1 1		a	0	0 0 0	
a	0	0 0	0	a	0	0 1	1	a	0	1 0	0
a	0	0 0	0	a	0	0 0	1	a	0	1 1	0
a	0	0 0	0	a	0	0 0	0	a	0	1 1 1	

in $(0,(0,0,0),(0,0,0),aaa,aaa)$ $(0,(1,0,0),(0,0,0),aaa,aaa)$ $(0,(0,0,0),(0,0,0),baa,aaa)$
 out $(0,(0,0,0),(0,0,0))$ $(0,(0,0,0),(0,0,-1))$ $(1,(1,0,0),(0,0,1))$
 $A[0,0,0,0]=(0,0,0)$ $A[9,0,0,0]=(0,0,2)$ $A[0,0,9,0]=(1,9,1)$

-1 זה 2 בקידוד בסיס 3 ע"מ לייצג את -1.

נשמר טבלה חדשה F אשר נשמר בה את קידוד היציאה של הבלוק.

ונחשב קידוד היציאה של כל בלוק בהתאם לשאל לימיון ומלמעלה למטה

a b c a b c a b c											
a	0	1	2	3	4	5	6	7	8	9	
b	1			2			5				
a	2			1			4				
b	3	2	1	1	1	2	3				
b	4			2							
a	5			3							
a	6	5	4	4							

$(1,(-1,-1,0),(-1,-1,0))$	$(3,(0,1,1),(-1,-1,-1))$	
$(4,(-1,-1,0),(1,1,1))$		

דרך לחישוב וקטור היציאה

עד שקף 40

סיבוכיות זמן

- The size of table $A = \text{number of in-encodings}$
 $(0, v_1, v_2, s, t): 3^{b-1} \cdot 3^{b-1} \cdot |\Sigma|^{b-1} \cdot |\Sigma|^{b-1}$.

- v_1, v_2 are vectors of length $b - 1$ over $\{0, 1, -1\}$.
- s, t are string of length $b - 1$ over Σ .

- Time for filling A :
 $O((3|\Sigma|)^{2(b-1)} \cdot b^2) = O(2^{2\log_2(3|\Sigma|) \cdot b} \cdot b^2)$.
- Time for filling F' : $O(\frac{n}{b} \cdot \frac{m}{b})$.
- Total time $O(2^{2\log_2(3|\Sigma|) \cdot b} \cdot b^2 + nm/b^2)$.
- Taking $b = \frac{1}{4\log_2(3|\Sigma|)} \cdot \log_2 n$, we get
 $O(nm \log^2 |\Sigma| / \log^2 n)$.

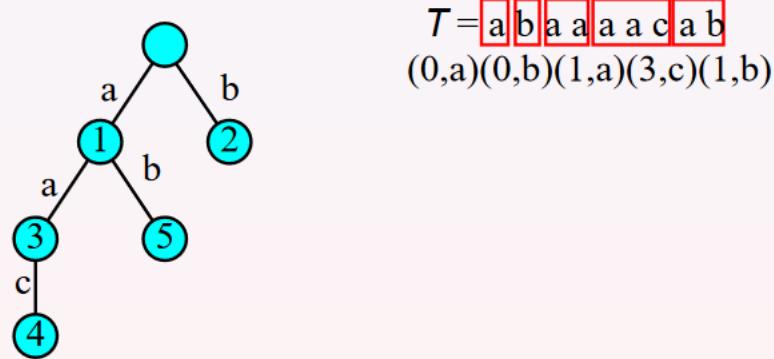
https://www.youtube.com/results?search_query=edit+distance

נרצה לפתור בעיה יותר כללית : לחשב מחיר לפי פונקציית מחיר להעודה אופטימלית.

באופן כללי האלגוריתם לעיל יכול להיות לא יעיל. נראה אלגוריתם דומה, נחلك לבLOCKים כאשר הגודל יהיה לפי גודלים משתנים לפי גודול היקלט. הBLOCKים יהיו לפי האלגוריתם של למפל זיו: 78:

אנו רוצחים לכזוץ טקסט : מחלקים את הטקסט לבLOCKים, מסתכלים על התווים שלא צוינו :
 לוקחים אותן ראשונה ומיצגים בעזרה *trie* כל בLOCK יתאים לצומת כאשר המספר מתאים למספר הBLOCK כאשר מחזרות היא שרשרת הBLOCKים.

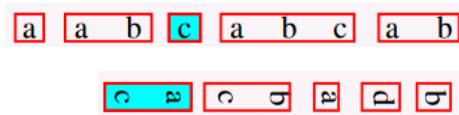
Partition a text T into blocks. Each block consists of a maximum prefix of the remaining text which is equal to a previous block plus one character.



אחרי שחילקנו לבLOCKים לכל BLOCK מדפיסים את מספר האב ותו אחרון בBLOCK ואת הסדרה מקודדים, סדרת הזוגות הן הכיוך.

עובדה: אם $a'b$ קבוע אז מספר הBLOCKים הוא $(\frac{n}{\log n})^0$. עבור אלפבית בגודל קבוע אז מספר הBLOCKים המתקבלים ממחרוזת באורך n בכיווץ 78 הוא $O(n \log n)$

ניקח את המחרוזות S וונחلك לבLOCKים לפי למפל זיו:



לכל בחירה של BLOCK זה יגדר BLOCK במטריצה

We partition the strings S and T into blocks.
For each block in S and block in T , we have a block in the alignment table.

	a	a	b	c	a	b	c	a	b
b									
d									
a									
b									
c									
a									
c									

Block (5,2) –ab/ac

אם מסתכלים על בלוק מעניין אותו הערכים של D, C, B, A כשר סדר המספר בכיוון השעון

table.

	a	a	b	c	a	b	c	a	b
b									
d									
a									
b									
c									
a									
c									

$I_3 \quad I_4 \quad I_5 \quad I_6$
 $I_2 \quad \quad \quad O_4$
 $I_1 \quad O_1 \quad O_2 \quad O_3$

cut נסמן בז את מספר תא ה-O למשל בדוגמה 5 – 4.

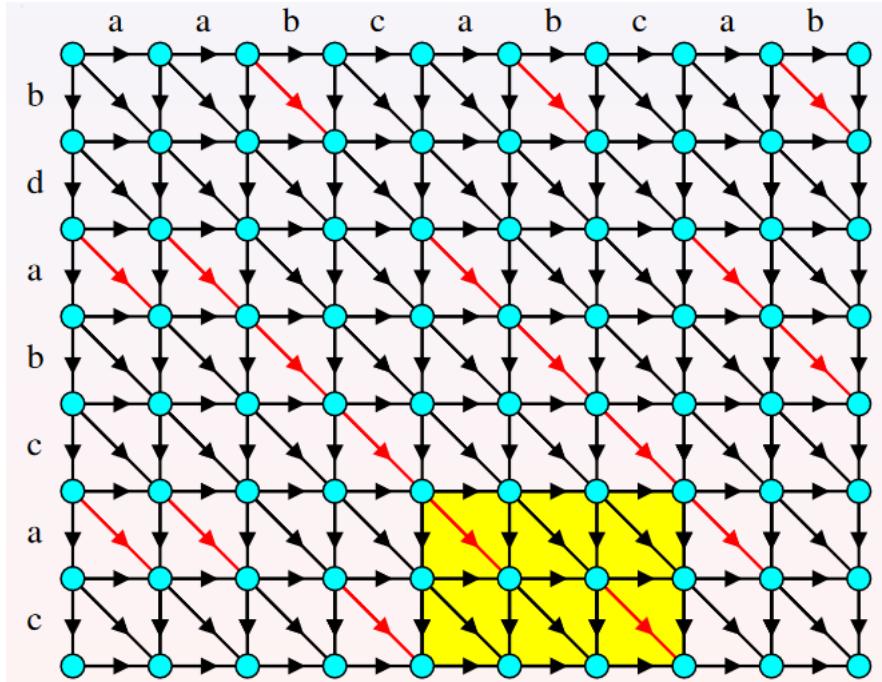
נניח שאם מישחו נתונים את ערכי / נוכל בזמן לינארי לגודל הבלוק את ערכי O.

נקבל שזמן הריצה יהיה לינארי בגודל כל הבלוקים שאנו מלאים.

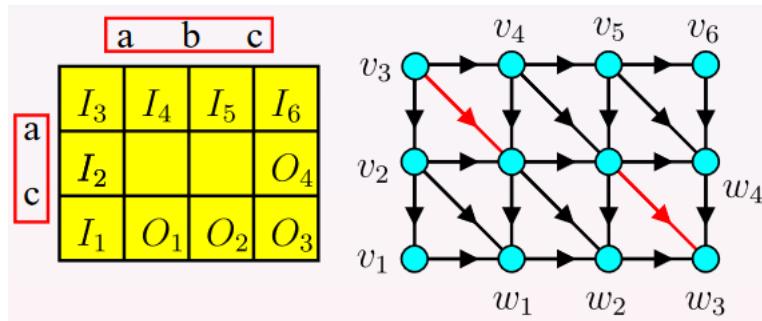
מילי השורות בבלוקים : אורך השורה m – אורך T וכן יש $(\frac{m}{\log m})$ בלוקים של S. אותו דבר עבור העמודות.

לכן סה"כ זמן הריצה יהיה $O(\frac{mn}{\log m})$.

כל בלוק בטבלה מתאים לבлок בגרף העמדה, נניח שאנו רוצים למצוא מסלול מקסימלי : קשת אנכית או אופקית 1-, והאdomות במשקל 1



אם נסתכל על תת-גרף של הבלוק כאשר אנו צריכים למסוף אותו לפני כיוון השעון :



בנוסף נסמן את הצלתיים של D ב- ω .

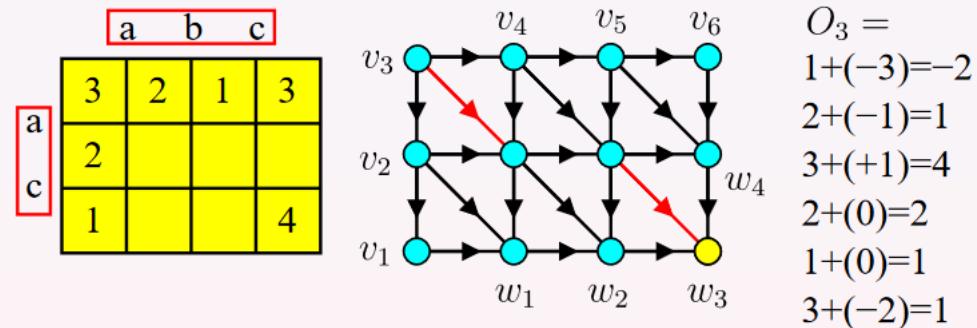
נניח שאנו רוצים לחושב את O_3 :

נבדוק את כל האפשרויות : נסתכל על מסלול מ- v_1 ל- w_4 ונוסיף אותו למשקל המסלול הנוכחי ב- v_1 .
נעשה זאת עבור כל אחד מקודקוד, 6 ביטויים ולקחת את המקסימום שהוא 4.

הערכים בכל היחסובים כוללים ערכים ידועים וצריך לדעת את ערכי המסלולים שנוספים: צריך להכיר מטריצה שמכירה משקל של מטריצת $DIST$:

$$DIST[a, b] = d(v_a, w_b) = \max. \text{weight of path from } v_a \text{ to } w_b$$

Scores in the following examples: $\delta(a, a) = 1$, otherwise $\delta(a, b) = -1$.



- Let $d(x, y) = \max$. weight of path from x to y .
- $I_a = d((0, 0), v_a)$.
- $O_b = d((0, 0), w_b)$.
- $O_b = \max_a \{I_a + d(v_a, w_b)\}$.

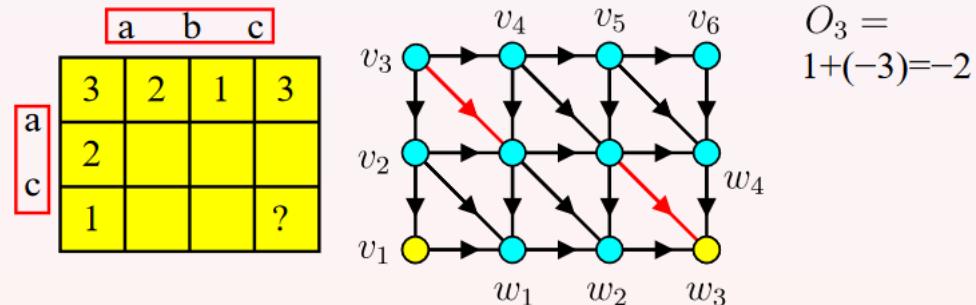
המטריצה $DIST$ היא Monge או הוכחה כמו הרצאה קודמת.

נסתכל בחישוב שאנו עושים:

$$I = [1 \ 2 \ 3 \ 2 \ 1 \ 3] \quad DIST =$$

-1	-2	-3	$-\infty$
-1	-2	-1	-3
0	0	1	-1
-2	-2	0	-2
$-\infty$	-2	0	-1
$-\infty$	$-\infty$	-2	-1

$$O = I \odot DIST = [3 \ 3 \ 4 \ 2]$$

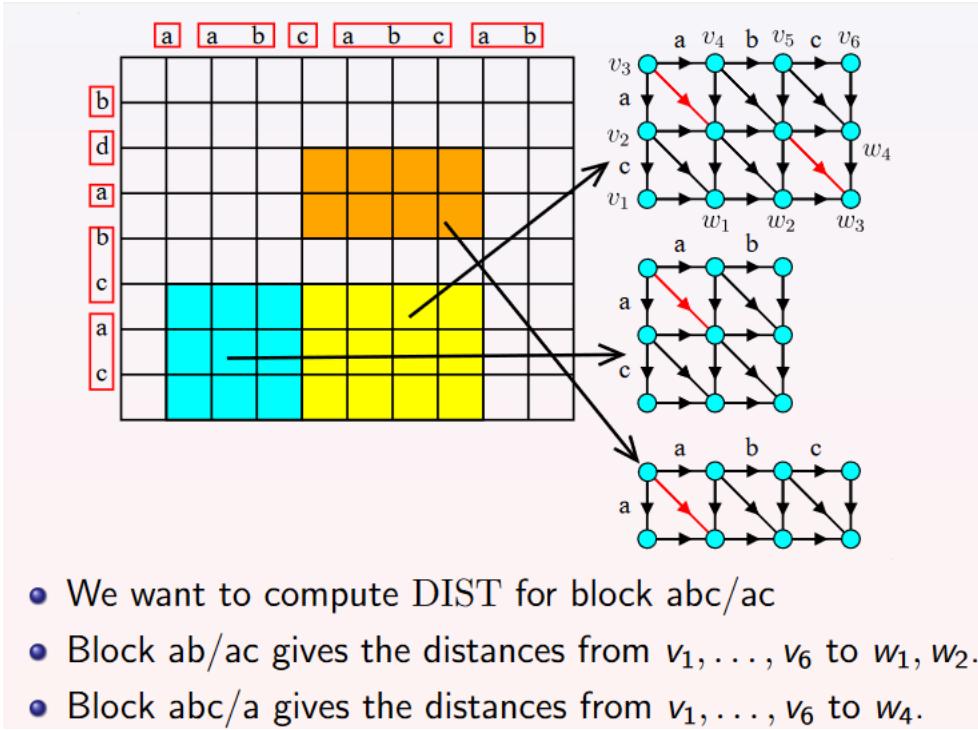


לקחhim איבר מ/ I וכן ממטריצת $DIST$ כולם עושים מכפלה בין $I^T * DIST$ ולכון נקבל את הערכים הרצויים.

אם ב/ I יש S ערכים אט $DIST$ יש $(2-S)^2$ ערכים

We have seen an algorithm that can compute $DIST$ in $O(S)$ time.

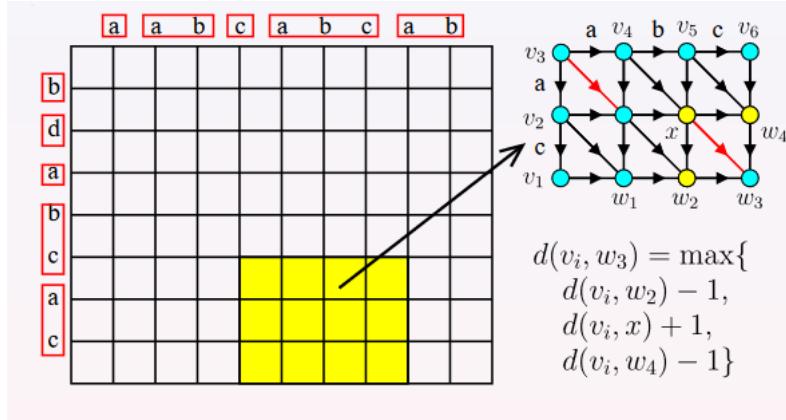
הבעיה היא שאנו רוצים את מטריצת $DIST$:



נסתכל על הבלוק ab , למול ac נטען שם נוריד את c איזי קיבל את הבלוק הקודם לפי תכונה של למפל זיו. لكن אנו מקבלים כבר חלק ממנו שבו חישבנו את מטריצת $DIST$ שלו.

משקלן הקשיות זהות כיוון שהאותיות זהות. אנו יודעים את המרחקים $l^{2,1}, l^{2,2}$.

לכן נשאר לנו רק לחשב את המרחקים $l^{3,2}$, אך על מנת לעשות זאת אנו יכולים להסתמך על מה שיש לנו לעומת מה צריך לחשב:

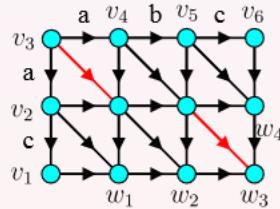
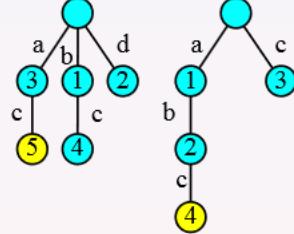


את ערכי d יש לנו ממטריצת $DIST$ שקיבלנו וכן אנו צריכים לחשב z ערכים כאלה.

בנייה מטריצת $DIST$: נשמר את המטריצה בתוור מערך של וקטורים כאשר כל מצביע לוקטור עמודה, אך רק צריך בכל פעם לשמור מצביע לורטוק מסוים שכבר חושב.

לכן נותר לחשב וקטור $3a$ לפי מה שאמरנו לעיל וכן ניתן לבנות את המטריצה בזמן לנארלי לגודל הבלוק. לכן סה"כ זמן הטיפול לנארלי בגודל הבלוק וכן סה"כ זמן הריצה כפי שנטען.

The matrix DIST of a block (i, j) is kept using array of pointers.



	1	2	3	4	2
1	a	ab	c	abc	ab
2	d				
3	a				
4	bc				
5	ac				

A

	w_1	w_2	w_3	w_4
DIST	-1	-2	-3	
	-1	-2	-1	-3
	0	0	1	-1
	-2	-2	0	-2
	-2	-2	0	-1
	-2	-2	-2	-1

סיבוכיות:

- $O(mn/\log^2 n) . 1$
- $O(mn \log^2 |\Sigma| / \log^2 n) . 2$

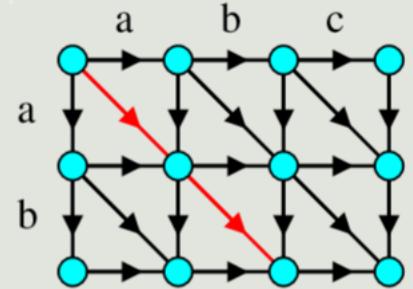
מצגת 13 Alignment of straight line programs

הגדה: עבור 2 מחוזות A, B היא מטריצת $DIST$ של גוף העמדה של A ו- B .

. $B[i \dots j]$ הוא משקל מקסימום של העמדה של A ו-

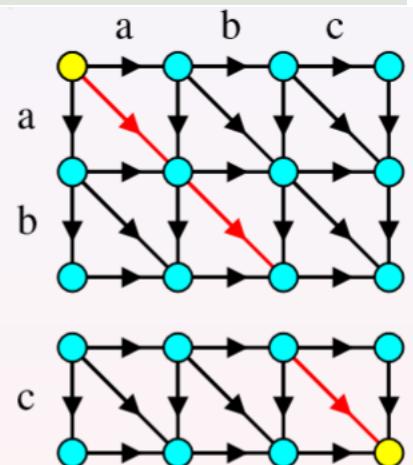
For LCS score,

$DIST_{ab,abc} =$	0	1	2	2
	$-\infty$	0	1	1
	$-\infty$	$-\infty$	0	0
	$-\infty$	$-\infty$	$-\infty$	0



$DIST_{ab,abc} =$	0	1	2	2
	$-\infty$	0	1	1
	$-\infty$	$-\infty$	0	0
	$-\infty$	$-\infty$	$-\infty$	0

$DIST_{c,abc} =$	0	0	0	1
	$-\infty$	0	0	1
	$-\infty$	$-\infty$	0	1
	$-\infty$	$-\infty$	$-\infty$	0



$$DIST_{abc,abc}[i,j] = \max_k DIST_{ab,abc}[i,k] + DIST_{c,abc}[k,j]$$

$$DIST_{abc,abc} = DIST_{ab,abc} \cdot DIST_{c,abc}$$

הציגת השרשור – 4-8

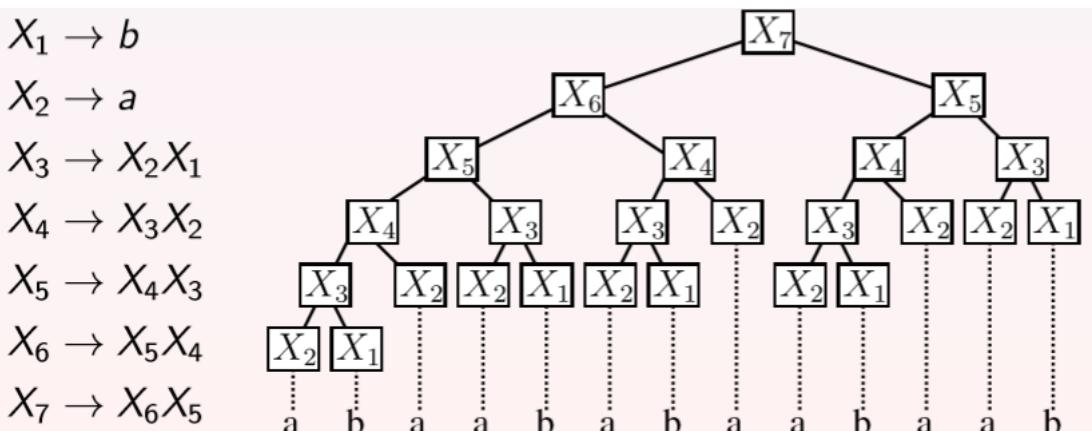
עובדה: המכפלה של 2 מטריצות $DIST$ בגודל $n \times n$ יכול להיות מחושב ב($n^2 \cdot O$).

עובדה: מרחוק ההפליה של 2 מטריצות LCS בגודל $n \times n$ יכול להיות מחושב ב($n \log n$).

DEFINITION: SLP (straight line program) – זה דקיקן חסר הקשר עם משתנים $X_1 \dots X_n$ המקיימים אחד מה הבאים:

- $a \rightarrow X_i$ כאשר a הוא טרמינל או (a) או $a \rightarrow X_p X_q$.
- $X_i \rightarrow X_j$ או $X_i \rightarrow X_j$.

דוגמא:



ניתן להפוך הרובה שיטות דחיסה ל- SLP .

$LZ77$ דוחס מחוזות באורך n ל- SLP ב- $O(\log n)$ כאשר L הוא אורך המחרוזת הלא דחוסה.

הגדרת הבעיה:

קלט: SLP עם משתנים X_1, \dots, X_{n_A} המיצרים את מחרוזת A ו- SLP עם המשתנים Y_1, \dots, Y_{n_B} המיצרים את מחרוזת B .

פלט: סכום אורכי המחרוזות A ו- B של A ו- B .

נסמן:

1. $n_A + n_B = n$ סכום אורכי SLP
2. L סכום אורכי המחרוזות A ו- B

נציין כי ניתן לפתור את הבעיה ב- $O(L^2)$ ע"י ייצור A ו- B וריצה בעזרת תכנון דינامي על A ו- B .

A simple algorithm

נסמן את מטריצת $DIST$ של A , B עבור A_i, B מחרוזת הנוצרת ע"י X_i . לדוגמה אצלנו:

$$A_3 = ab, X_3 = X_2 X_1 = ab$$

האלגוריתם המחשב את $DIST_{X_i, B}$ עבור $i = 1, \dots, n$:

1. אם $X_i \rightarrow a$ ניציר את $DIST_{X_i, B}$ ישירות
2. אם $X_i \rightarrow X_p X_q$ נחשב: $DIST_{X_i, B} = DIST_{X_p, B} \circ DIST_{X_q, B}$

בהתו: אם מספר הגזירות הוא לכל היותר n

סיבוכיות הזמן היא ($O(nL \log L)$) עבור LCS score

עבור SCORE כללי הזמן הוא ($O(nL^2)$) אשר גורע יותר מ($O(L^2)$).

Better algorithm

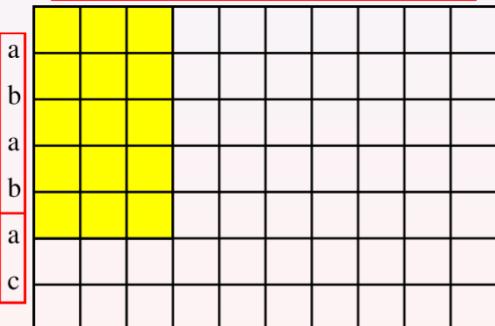
תהיה F טבלת העמדה של מחרוזות A ו- B (בדוגמא scores הם LCS scores)

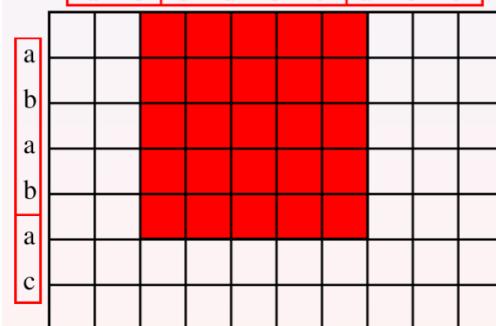
	a	b	c	a	b	c	a	b	c
a	0	0	0	0	0	0	0	0	0
b	0	1	1	1	1	1	1	1	1
a	0	1	2	2	2	2	2	2	2
b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	4	4	4	4
b	0	1	2	2	3	4	4	4	4
a	0	1	2	2	3	4	4	5	5
c	0	1	2	3	4	5	5	5	6

אלגוריתמים למחוזות / סיכום מאות אורי שביט

חלוקת של A ו- B לבלוקים כאשר כל זוג בלוקים ' A ' של A ו-' B ' של B מגדיר בלוק של טבלת העמדה.

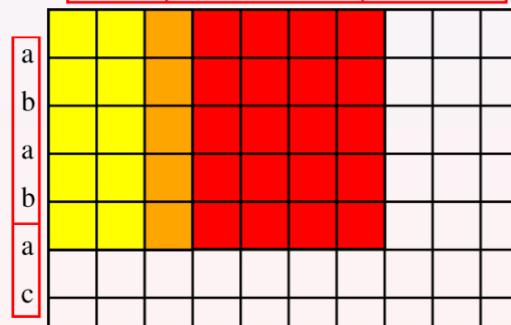
	a	b	c	a	b	c	a	b	c
a									
b									
a									
b									
a									
c									



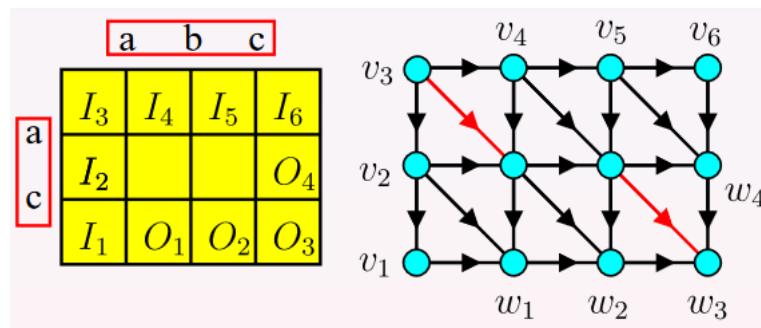


כאשר בלוקים סמוכים חולקים עמודה או שורה של תאים.

	a	b	c	a	b	c	a	b	c
a									
b									
a									
b									
a									
c									



בכל בלוק נסמן $k/1, k/2, \dots, k/4$ את התאים בעמודה והשורה הראשונות וב- $2k-0, 2k-1, \dots, 2k-3$ את התאים שבשורה והעמודה האחרונות.



נחשב רק את התאים שהם קלט או פלט של בלוק כלשהו :

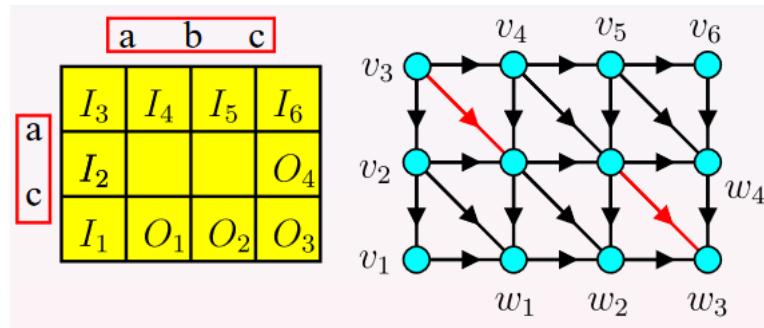
	a	b	c	a	b	c	a	b	c
a	0	0	0	0	0	0	0	0	0
b	0		1						1
a	0		2				2		2
b	0		2				3		3
a	0	1	2	2	3	4	4	4	4
b	0		2				4		5
a	0	1	2	3	4	5	5	5	6
c	0								

אלגוריתמים למחוזות / סיכום מאות אורי שביט

אנו מטפלים בבלוק שבו ערכי הקלט ידועים והמטרה היא לחשב את ערכי הפלט:

	a	b	c	a	b	c	a	b	c
a	0	0	0	0	0	0	0	0	0
b	0		1						
a	0		2						
b	0		2						
a	0	1	2						
c	0								

Computing the O cells



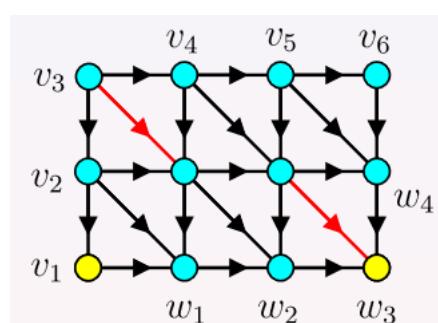
יהי $d(x, y)$ משקל המקיים של מסלול מ- x ל- y .

$$I_a = d((0,0), v_a)$$

$$O_b = d((0,0), w_b)$$

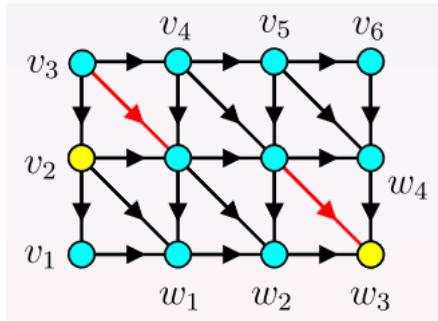
למשל אם נרצה לחשב את?

$$O_b = \max_a\{I_a + d(v_a, w_b)\}$$



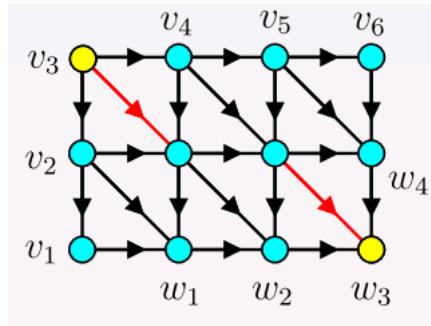
$$O_3 = 5 + 0 = 5 . 1$$

$$O_3 = 4 + 1 = 5 . 2$$



אלגוריתם למחuzeות / סיכום מאת אורי שביט

$$\begin{aligned}
 O_3 = & \\
 5+0=5 & \\
 4+1=5 & \\
 4+2=6 & \\
 4+1=5 & \\
 4+1=5 & \\
 4+0=4 &
 \end{aligned}$$

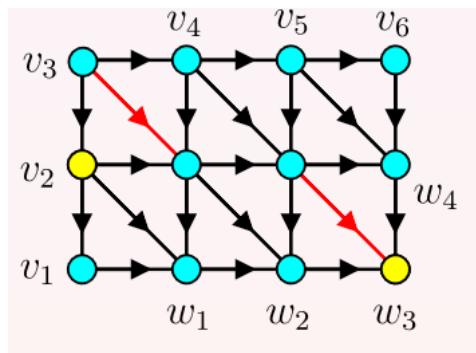


$$O_3 = 4 + 2 = 6 . 3$$

וכך הלאה

The DIST matrix

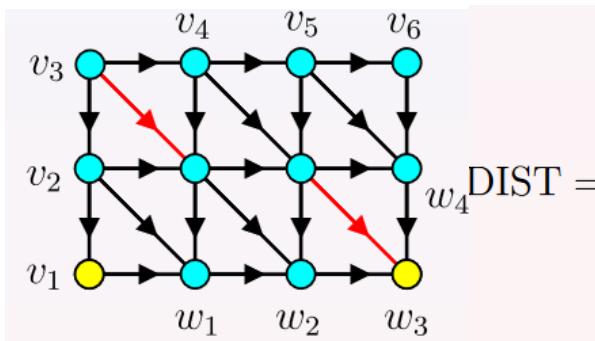
$$DIST[a, b] = d(v_a, w_b) = \max. weight of path from v_a to w_b$$



DIST =

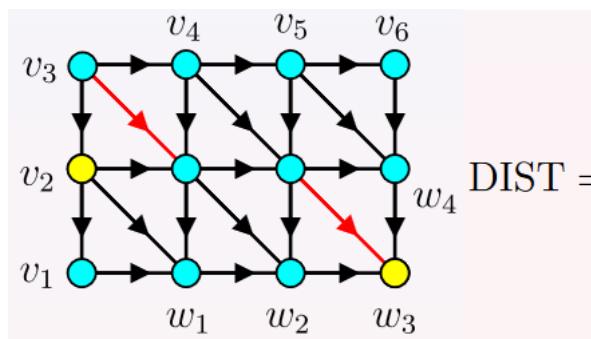
0	0	0	$-\infty$
0	0	1	0
1	1	2	1
0	0	1	0
$-\infty$	0	1	0
$-\infty$	$-\infty$	0	0

כפי שהוכח בשיעורים קודמים.



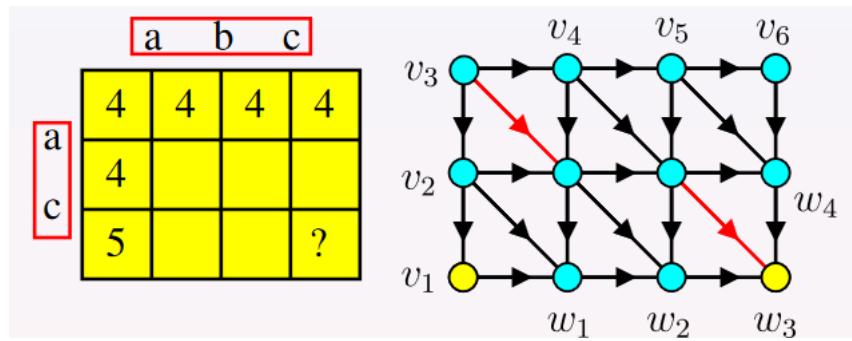
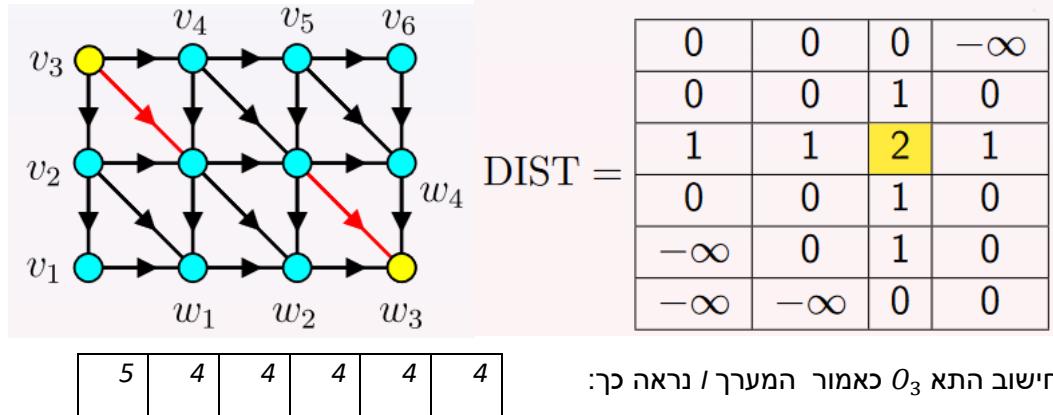
DIST =

0	0	0	$-\infty$
0	0	1	0
1	1	2	1
0	0	1	0
$-\infty$	0	1	0
$-\infty$	$-\infty$	0	0



DIST =

0	0	0	$-\infty$
0	0	1	0
1	1	2	1
0	0	1	0
$-\infty$	0	1	0
$-\infty$	$-\infty$	0	0



$$O = I \cdot DIST = [5, 5, 6, 5]$$

עבור $_1$ קיבל 5 והוא התא הראשון במערך O .

עבור $_2$ קיבל 5 והוא התא השני במערך O .

עבור $_3$ קיבל 6 הוא התא השלישי במערך O .

נניח של I יש s איברים ו- $DIST$ בגודל $(s - 2) \times s$.

ראינו שיש לנו אלגוריתם המחשב את $DIST \cdot I$ ב(s) זמן.

סיבוכיות זמן $(L \cdot \#blocks(A) + \#blocks(B)) \cdot O$ (ניתוח כמו פעם שעבירה)

קווים מנהיים לאלגוריתם

1. חלוקה של A ו- B לבלוקים לפי SLP .

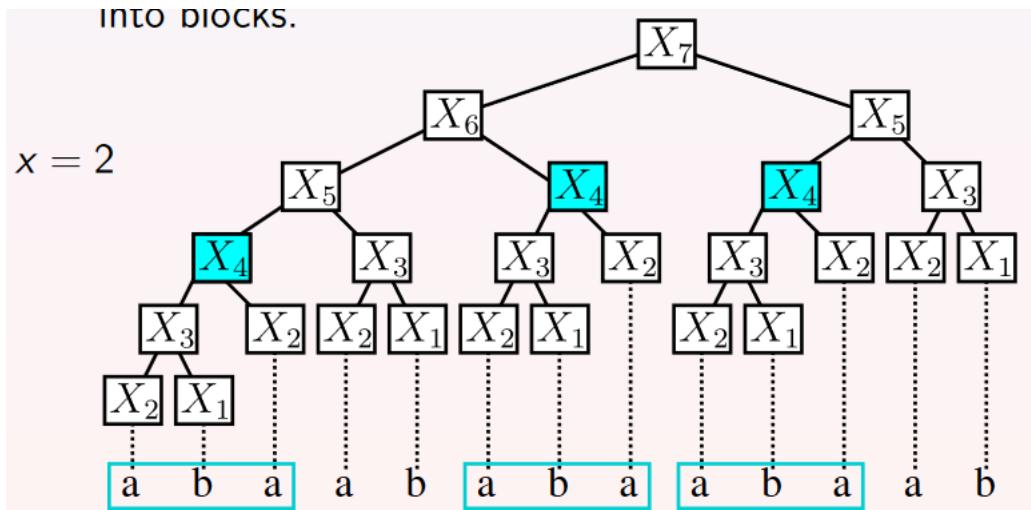
2. חישוב מטריצת $DIST$ עבור כל בלוק של F .

3. חישוב ערכי O עבור כל בלוק F .

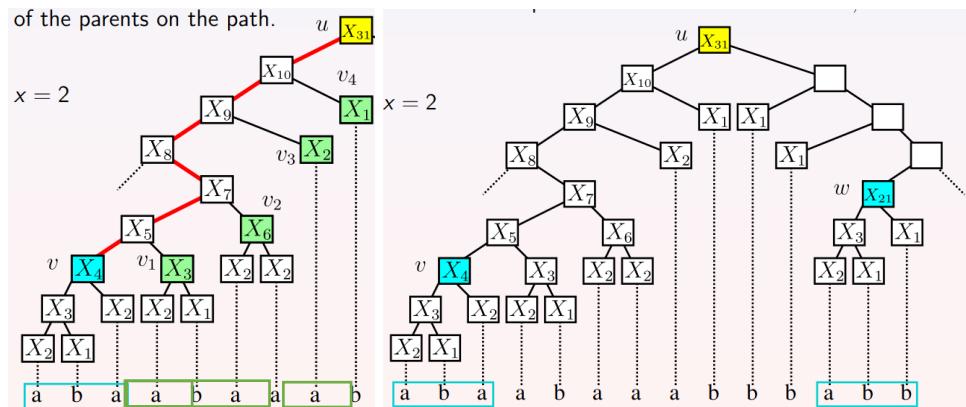
חלוקת לבlokים

נחלק את A ו- B לבלוקים בגודל $(x/L) \times (x/L)$ (L סכום אורכי המחרוזות A ו- B) נקבע את x אחר כך. כזכור שלב 3 ייקח $O(\frac{L^2}{x})$.

נסמן כל קודקוד של עץ הגזירה של A שבו המחרוזת ארוכה מ- x . כל מחרוזת עם צומת מסומן נסמנת כבלוק מסוג 1 של A . אנו רוצחים את החלוקת לתת מחרוזות של בלוקים מסווג 1.



יהיו 2 בלוקים רצופים של A התאימים לקודקודים v ו- w בעץ הגזירה של A . יהי u ה- LCA של v ו- w .



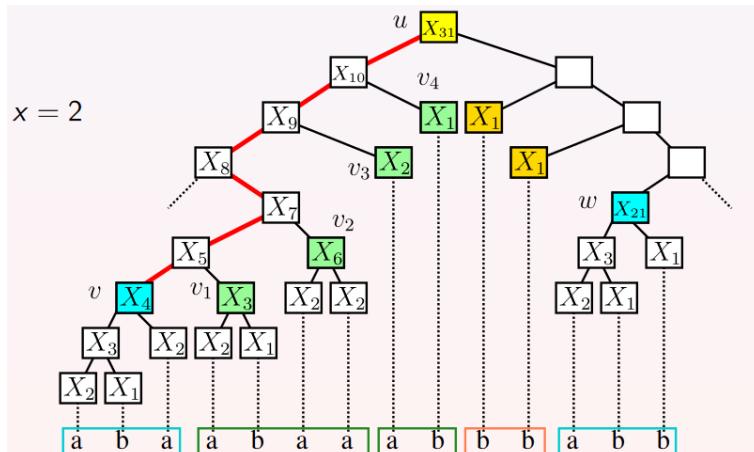
יהיו u_k, \dots, u_1 הקודקודים שהם הילדיים הימניים המסלול מה u , מסודרים לפי סדר ההורים במסלול. בירוק ניצר בלוק מסווג 2 – הם הוספת תת המחרוזות \dots, u_2, u_1 עד להגעה לבלוק שגודלו גדול מ- a . (ריבועים בירוק)

נניח שהבלוק הראשון המתאים ל u_i, \dots, u_1 נמשיל את תחיליך החלוקה עבור u_k, \dots, u_{i+1} .

חלוקת תת המחרוזות הנותרות ע"י הסתכלות על קודקודים שהם בניים שמאליים במסלול מה u .(בכתום)

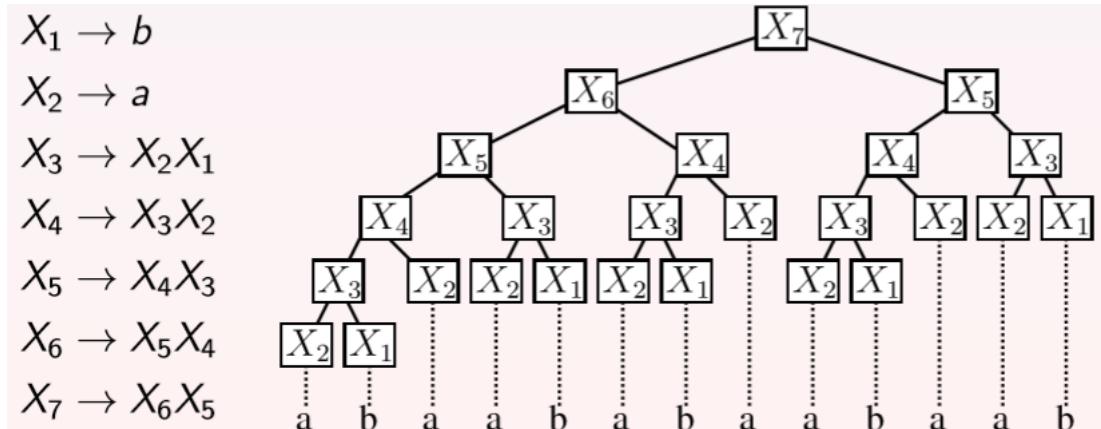
בלוקים מסווג 1 – מחרוזת של קודקוד גזירה שאורכה גדול מ- a וקטן שווה לא- 2 . ולכן יש לכל היוטר $\frac{L}{x}$ כאלה.

בלוקים מסווג 2 – הוספת תת מחרוזות של בניים ימניים במסלול עד לקבלת בלוק בגודל גדול מ- a . ולכן נצפה שייהו לכל היוטר $\frac{2L}{x}$ בלוקים כאלה. ולכן מספר בלוקים מסווג 2 הוא $O\left(\frac{L}{x}\right)$.



חלוקת A-B לבלוקים - כשמחלצים את ה- SLP נוצר עץ שהצמתים שלו הם כלליםXi. נסמן כל צומת בעץ שיש בתת-העץ שלו יותר $M^{3/2} = \Theta(L^2)$ עליים ולכל אחד משני בניו יש לכל היותר x עליים. מחוזות בתת-עץ של צומת מסומן נקראת בלוק מסוג 1. את המחווזות בין צמתים מסווג 1 נחlik כרך - נמצא את ה-LCA של הצמתים, ונסמן את הבנים הימניים של הצמתים על המסלול מהצומת השמאלי ב- $a_1 \dots a_L$, נוצרו עליים שבתתי-העצים של $a_1 \dots a_L$ (כל העליים של תת-עץ אחד ביחד), וכשנגייע למחווזה $< a$ נתחיל בלוק חדש. הבלוק שנוצר הוא בלוק מסוג 2. הבלוק האחרון עשוי להיות $> a$. ככל עם הבנים השמאליים של המסלול מה-LCA לצומת מסווג 1 הימני.

Computing the DIST tables



מחשב את $DIST_{X_1,B'}$, $DIST_{X_2,B'}$ לכל בלוק' B' של B

מחשב את $DIST_{X_3,B'} = DIST_{X_2,B'} \cdot DIST_{X_1,B'}$ לכל בלוק' B' של B

מחשב את $DIST_{X_6,B'} = DIST_{X_2,B'} \cdot DIST_{X_4,B'}$ לכל בלוק' B' של B

מחשב את $DIST_{X_4,B'} = DIST_{X_3,B'} \cdot DIST_{X_2,B'}$ לכל בלוק' B' של B

מחשב את $DIST_{X_3X_6,B'} = DIST_{X_3,B'} \cdot DIST_{X_6,B'}$ לכל בלוק' B' של B

מחשב את $DIST_{X_2X_1,B'} = DIST_{X_1,B'} \cdot DIST_{X_2,B'}$ לכל בלוק' B' של B

чисוב $DIST_{x,y}$ - עבור כל בלוק' B' ב- B מחשב את $DIST_{xi,B'}$ עבור כל Xi שהמוחוזת שלו $= > a$ ועבור כל צומת מסומן X_i , וכן את $DIST_{Xij,B'}$ עבור כל רישא $X_{i1} \dots X_{iL}$ של בלוק מסוג 2 $X_{i1} \dots X_{iL}$.

בינות

1. חישוב המרחקים לוקח $(x^2)O$ גדול כל בלוק או תת בלוק של A או B אשר קטן שווה לא- x .

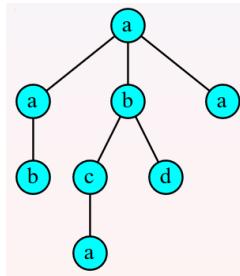
2. כל בלוק או תת בלוק של A או B יכול להתאים למשתנה ייחודי X_i . כלומר יש לכל היותר $n_A * n_B$ מרחקים המחשבים.

3. זמן בניית מטריצת DIST הוא $O(n^2x^2)$.

4. לכן זמן האלגוריתם כולל הוא $O(n^2x^2 + \frac{x^2}{n})$.

על מנת להביא לסבירות זמן מינימלית אנו רוצים שיטק'ים $x = \frac{n^2}{L^2}$ ונקבל $(\frac{n}{L})^{\frac{2}{3}}$

סבירויות זמן $O(n^{\frac{2}{3}} * L^{\frac{4}{3}})$.



אלגוריתמים למחuzeות / סיכום מאות אורי שביט

מצגת 14 Tree Edit Distance

הגדרות:

עץ שבו לקודקוד יש label מא'ב ס. – labeled tree

זה עץ מושרש עם סדר לינארי על יד של לקודקוד : – ordered rooted tree

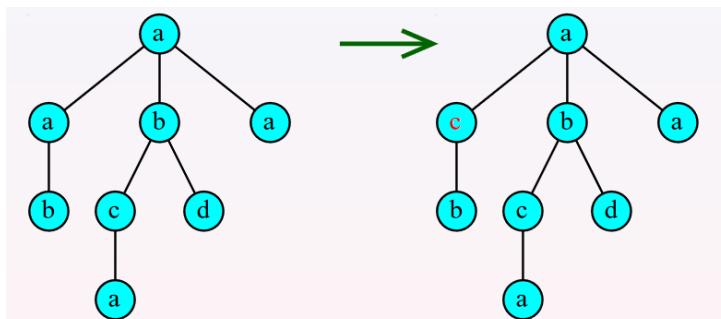
aab, abca, tbd, aa

.ordered trees – זהו אוסף מסודר של ordered forest

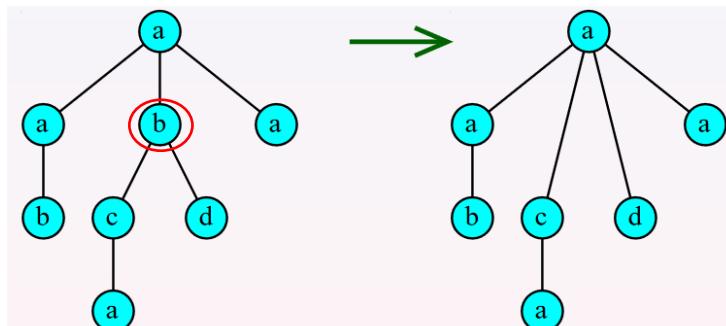
מוטיבציה: RNA, XML

Edit operations

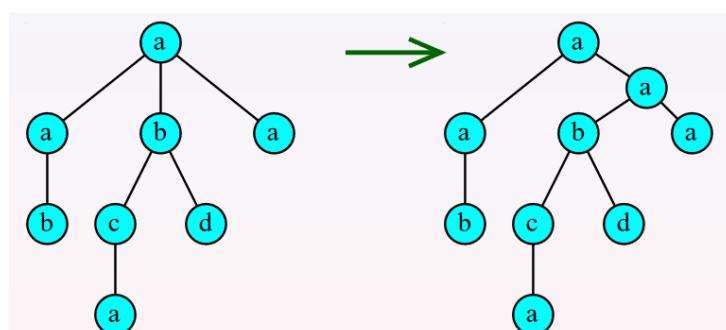
1. שינוי label של קודקוד אחד.



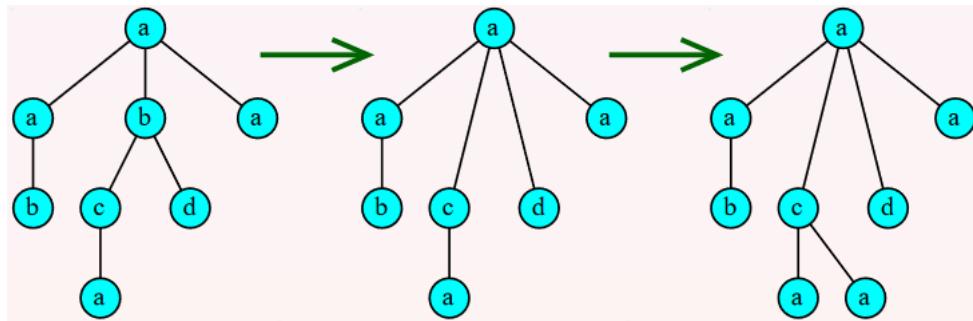
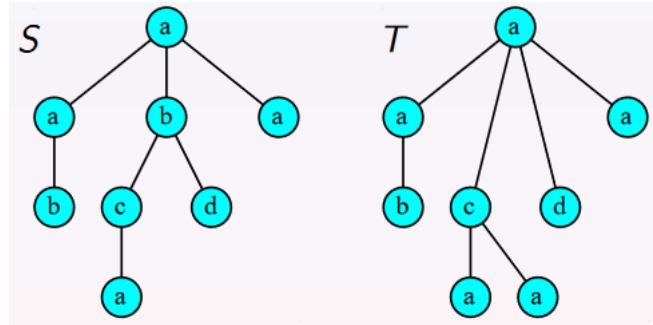
2. מחיקת קודקוד ותליית כל ילדים שלו על ההורה שלו.



3. הכנסת קודקוד חדש בילד של קודקוד כשלשו ו, כאשר חלק מהילדים של יתלו על הקודקוד החדש.

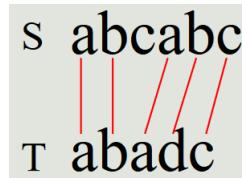


עריכה שהופכים את S ל T . לדוגמה: מרחק העריכה של S ו T הוא 2, הפעולות להפיכה S ל T – $Tree edit distance$ – עברו 2 – $ED(S, T)$ הוא המספר המינימאלי של פעולות



Strings alignment

העמדה של S ו T היא התאמה M בין $\{1, \dots, |S|\}, \{1, \dots, |T|\}$ כאשר לכל $M \in \{1, \dots, |S|\}, \{1, \dots, |T|\}$ כהן $(i, i'), (j, j') \in M$ בזאת $S[i] = T[i']$.



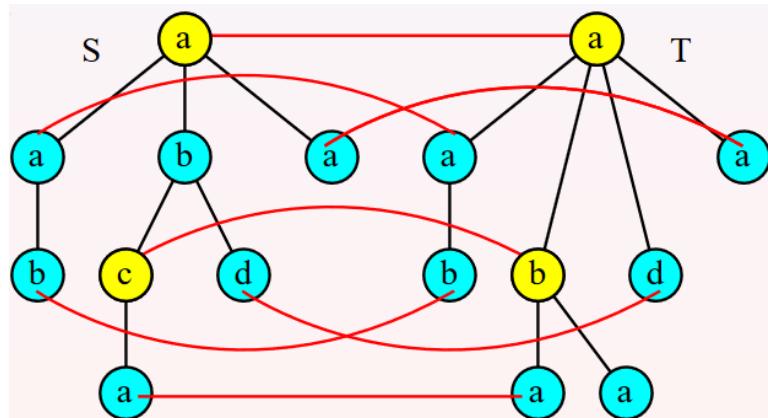
מחיר העמדה M הוא מספר האינדקסים הלא מתאימים של $M \in \{1, \dots, |S|\}, \{1, \dots, |T|\}$ כהן $(i, i'), (j, j') \in M$ בזאת $S[i] \neq T[i']$.

העמדה של יערות S ו T היא העמדה M בין $\{1, \dots, |S|\}, \{1, \dots, |T|\}$ כך שעבור כל $M \in \{1, \dots, |S|\}, \{1, \dots, |T|\}$ מתקיימים:

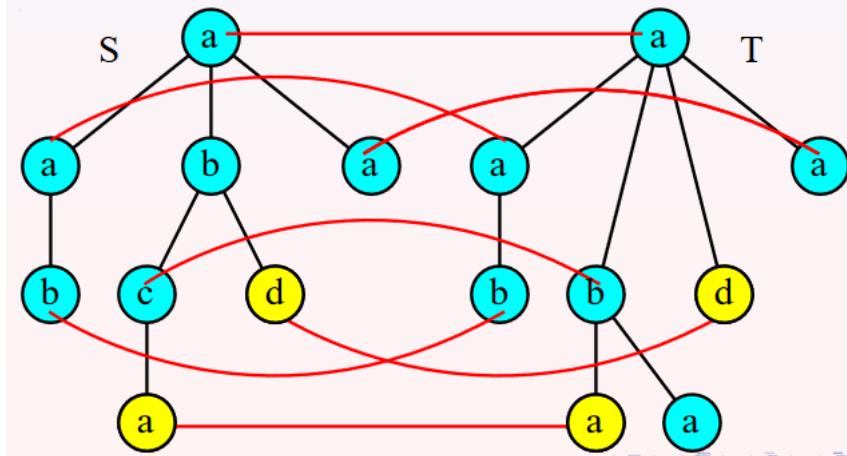
1. הוא אב קדומות של j אם i' הוא אב קדומות של j' .

2. $.postorder(i) < postorder(j) \iff postorder(i') < postorder(j')$

לדוגמא עבור תנאי 1:

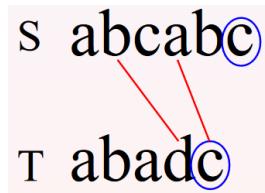


דוגמא לתנאי 2:

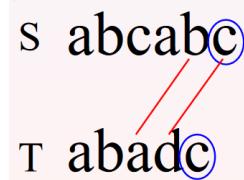


Computing string edit distance

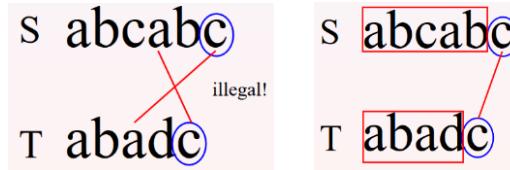
עבור מחרוזות S ו- T נסתכל על התו האחרון של המחרוזות. לדוגמה : $S = abcabc$, $T = abadc$: $S[n]$ לא מתאים להערכה האופטימלית M .



$T[m]$ לא מתאים ב- M



$S[n]$ ו- $T[m]$ מתאים ב- M . $S[n]$ חייב להתאים ל- $T[m]$.



illegal!

עבור מחרוזות S ו- T נסתכל על התו האחרון של המחרוזות ולכн 3 במקרים :

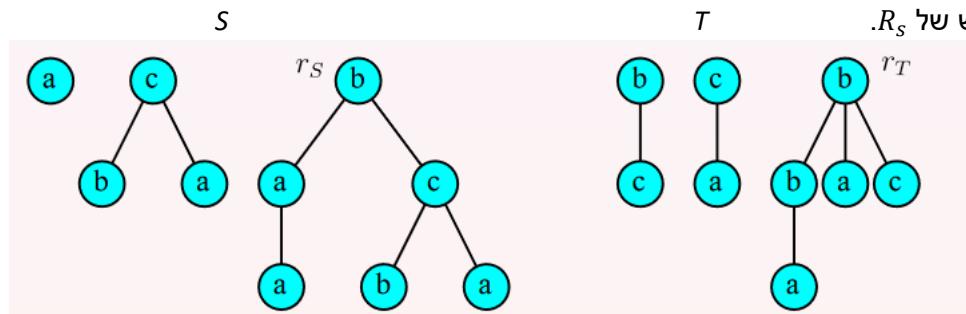
- $S[n]$ לא מתאים להערכה האופטימלית M .
- $T[m]$ לא מתאים ב- M
- $S[n]$ ו- $T[m]$ מתאים ב- M . $S[n]$ חייב להתאים ל- $T[m]$

$$ED(S, T) = \min \begin{cases} ED(S[1..n-1], T) + 1 \\ ED(S, T[1..m-1]) + 1 \\ ED(S[1..n-1], T[1..m-1]) + \delta(S[n], T[m]) \end{cases}$$

הגדרות:

R_s – העץ הימני ביותר בס.

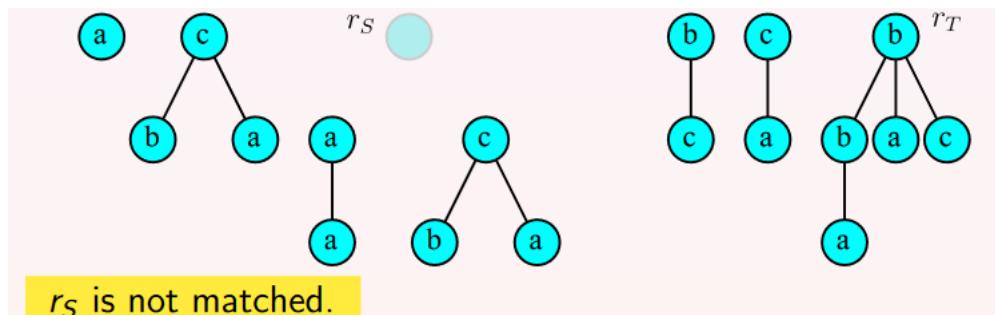
r_s – השורש של R_s .



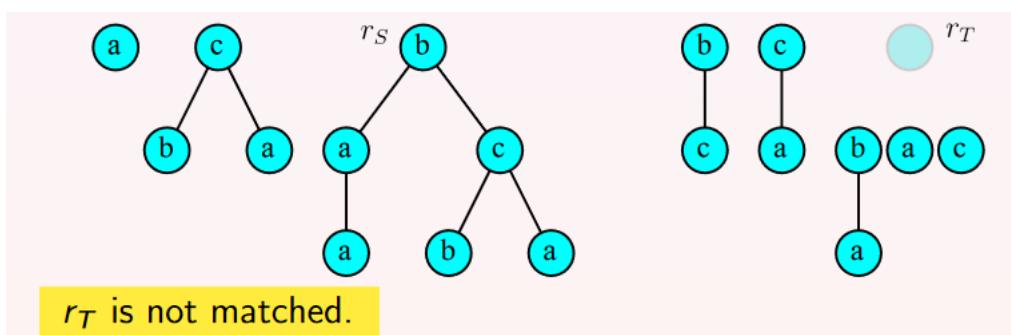
ולכן נציב בנוסחה נקבע:

$$ED(S, T) = \min \begin{cases} ED(S - r_s, T) + 1 \\ ED(S, T - r_t]) + 1 \\ ED(S - R_s, T - R_t) + \delta(r_s, r_t) \end{cases}$$

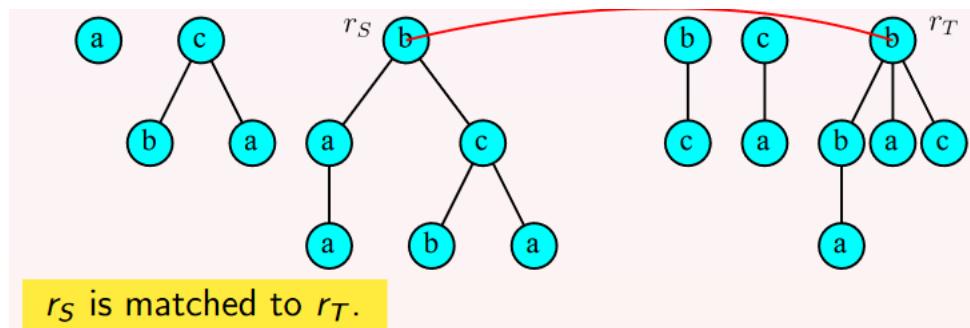
$ED(S - r_s, T) + 1 . 1$



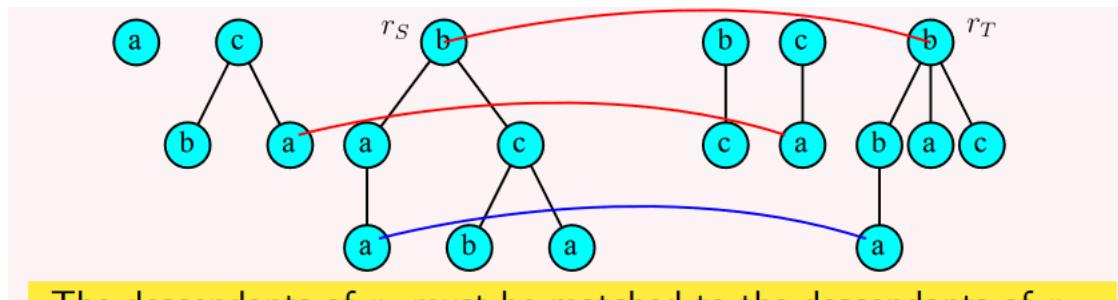
$ED(S, T - r_t]) + 1 . 2$



$ED(S - R_s, T - R_t) + \delta(r_s, r_t) . 3$

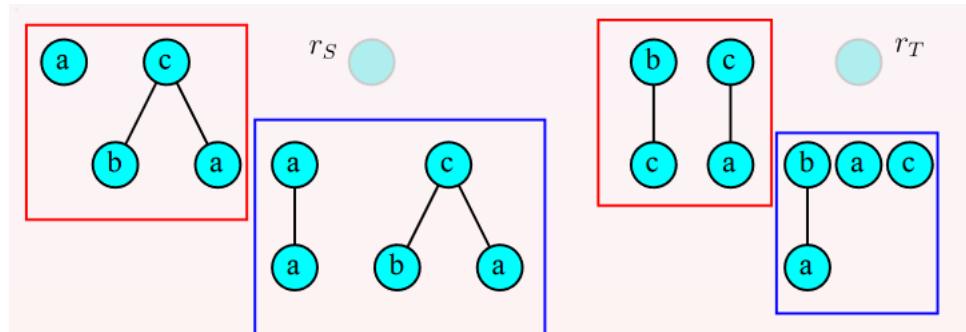


$$ED(S - R_s, T - R_t) + ED(R_s - r_s, R_T - r_T) + \delta(r_s, r_t)$$

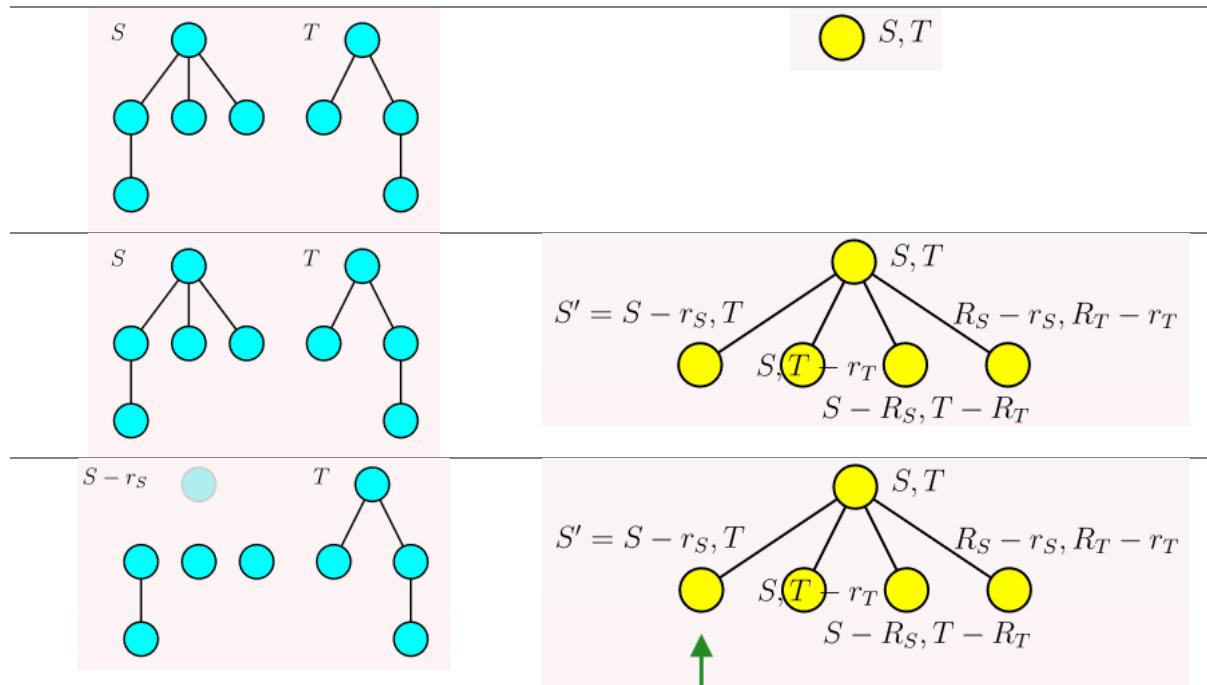


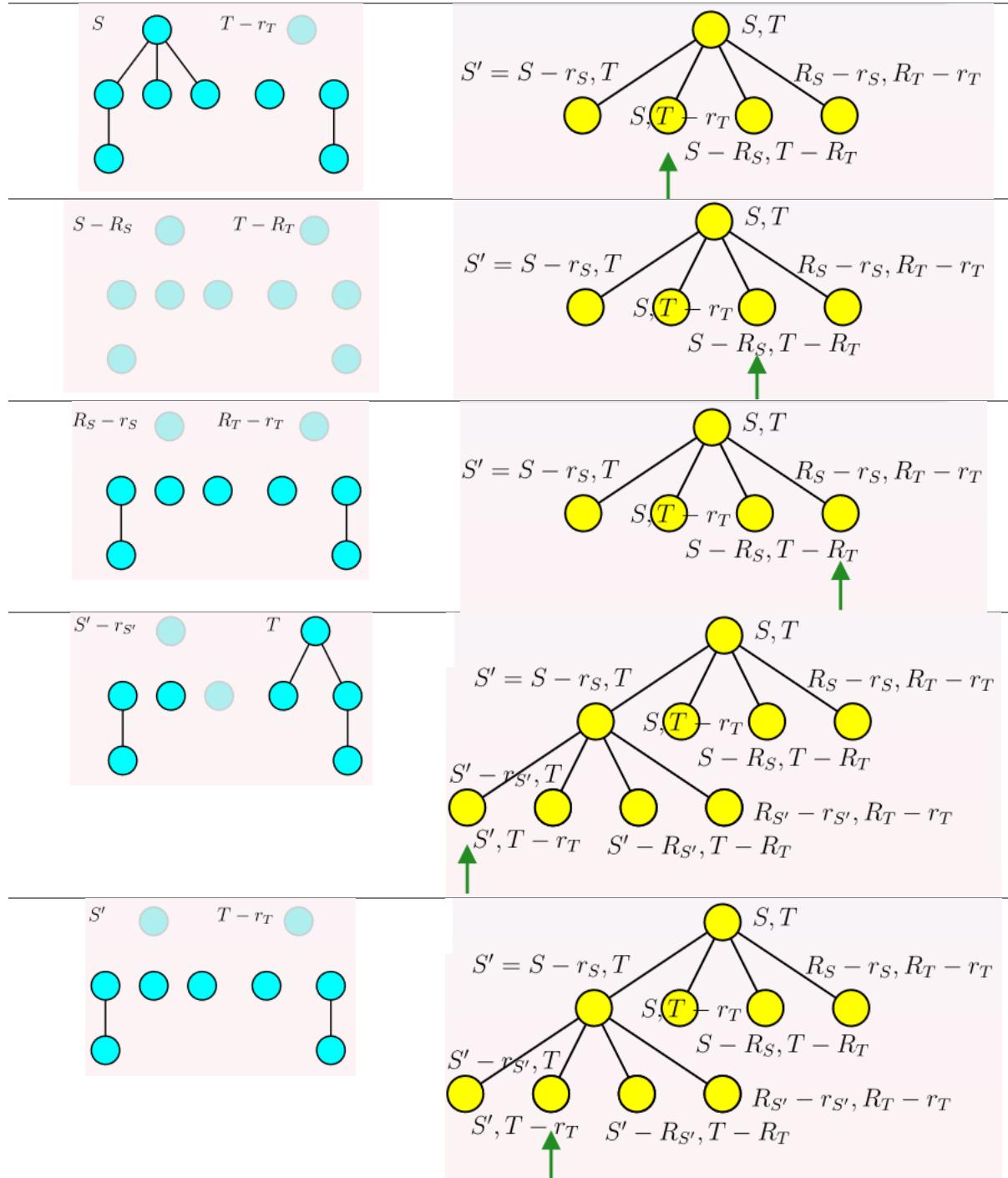
The descendants of r_S must be matched to the descendants of r_T .

נקבל את היעורות:

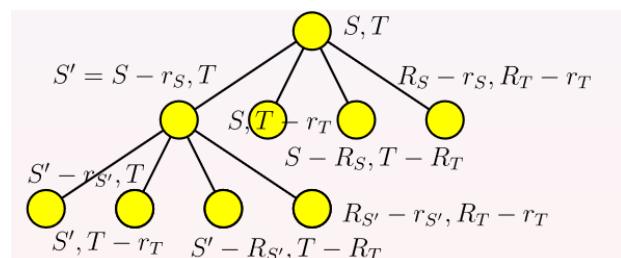


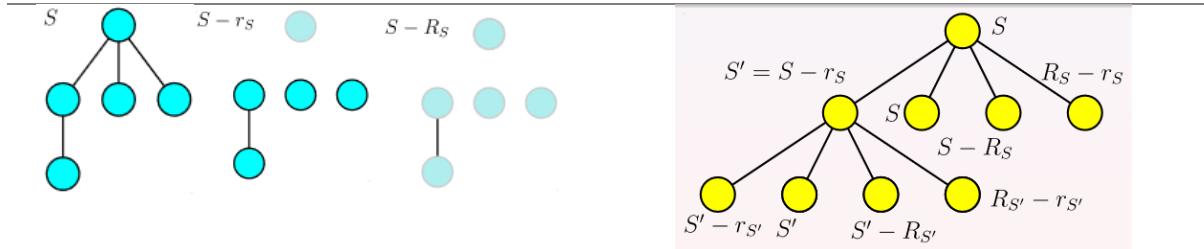
הגדירה: בהינתן זוג רלוונטי (S, T) הוא זוג (S', T') עבורו אנו מחשבים את $ED(S', T')$ בזמן $.ED(S, T)$





הגדרה: (תת יער רלוונטי) של S הוא יער S' כך ש(S', T') הוא **relevant pair** עבור T' .
כלשהו. (S', T') עבורי אם מחשבים את $ED(S', T')$ בזמן חישבו $ED(S, T)$.

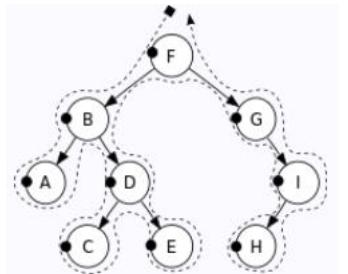




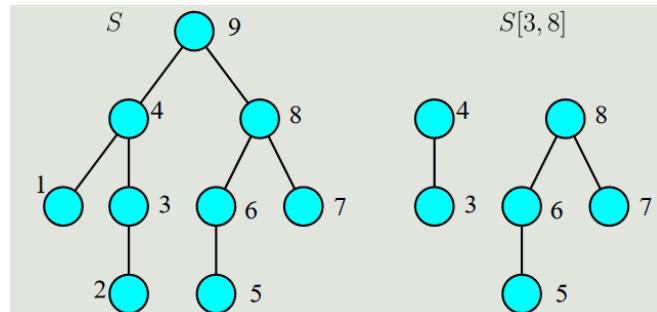
Relevant subforests
הו תחת עיר של S עבור כל הקודוקודים בין / לפני סדר $postorder$ $S[i, j]$

Algorithm Postorder(tree)

1. Traverse the left subtree, i.e., call Postorder(left-subtree)
2. Traverse the right subtree, i.e., call Postorder(right-subtree)
3. Visit the root.



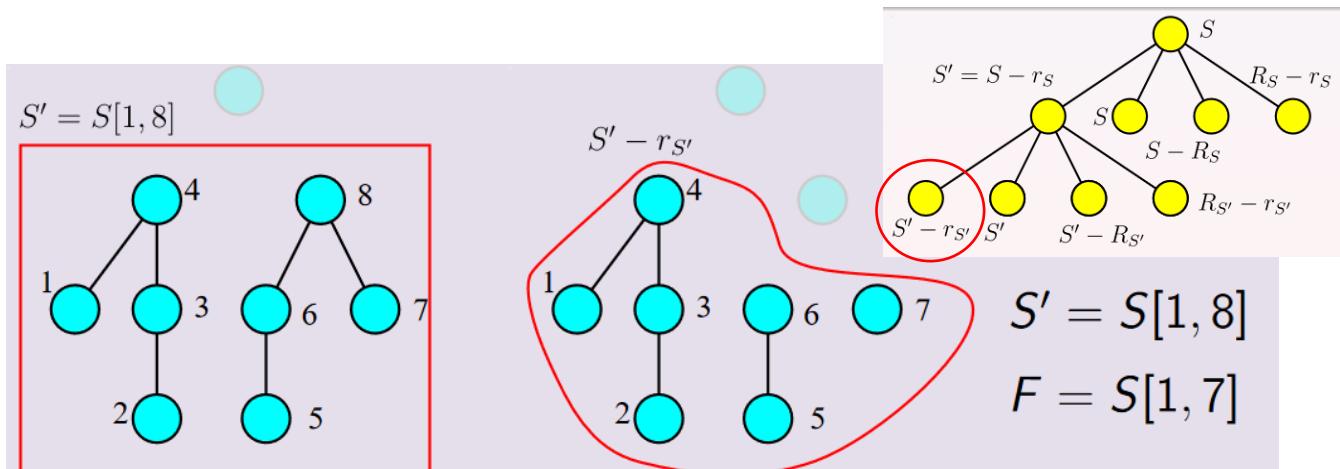
דוגמה:



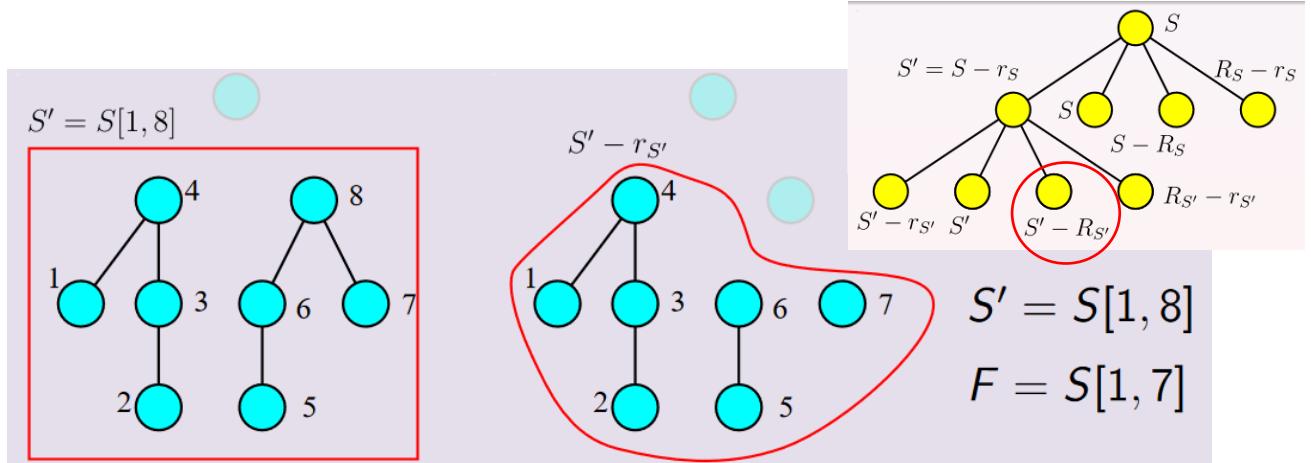
הו תחת עיר של S הוא עיר S' כך ש(S', T') הינו *relevant pair* עבור T' קלשחו.
הו זוג (S', T') עבורו אנו מחשבים את $ED(S', T')$ בזמן חישבו ($ED(S, T)$)
למה: אם F הוא תחת עיר של S כך ש(i, j) $F = S[i, j]$ קלשחו.

הוכחה: נוכיח באינדוקציה על עומק F בעץ רקורסיבי.

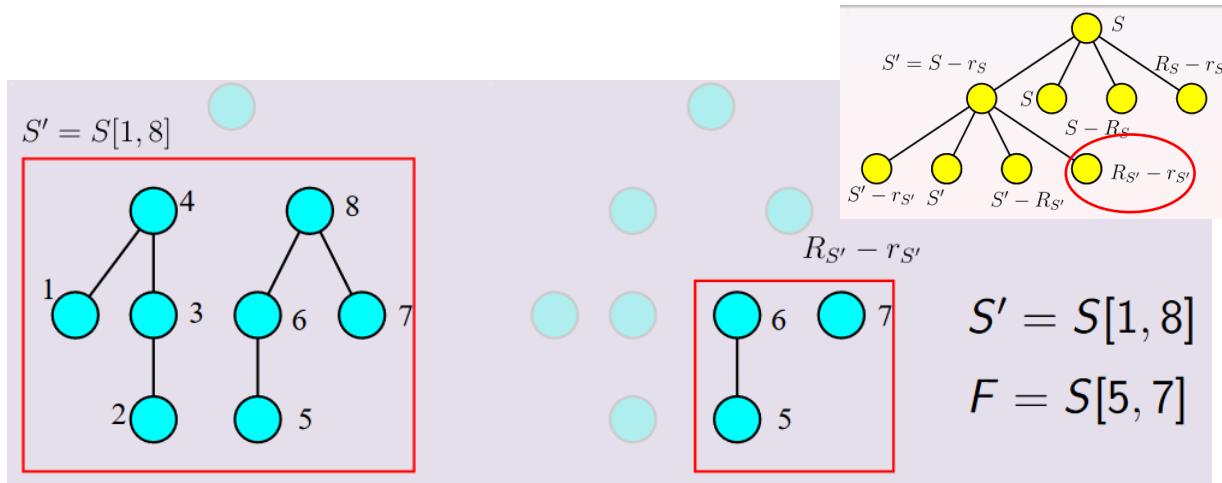
יהי $F = S[i, j - 1]$ הוראה של תחת עיר של F . אם $i' = S' - r_{S'}$ אז $F = S[i', j]$



$.F = [i, j - 1]$ ואם $F = S' - r_{S'}$



אם $.F = S[j - |R_{S'}| + 1, j - 1]$ ואם $F = R_{S'} - r_{S'}$



הוכחה: נוכיח באינדוקציה על עומק F בעץ רקורסיבי.

יהי $[i, j]$ הורה של $S' = S[i, j]$ יער של F .

אם $.F = S[i, j - 1]$ ואם $F = S' - r_{S'}$

אם $.F = [i, j - 1]$ ואם $F = S' - r_{S'}$

אם $.F = S[j - |R_{S'}| + 1, j - 1]$ ואם $F = R_{S'} - r_{S'}$.

מספר ה-*relevant subforest* של S הוא לכל היותר n^2 .

מספר ה-*relevant subforest* של T הוא לכל היותר m^2 .

מספר ה-*relevant pair* הוא לכל היותר n^2m^2 .

אנו שומרים טבלא F עבורה ($F(i, j, i', j') = ED(S[i, j], T[i', j'])$)

אנו ממלאים את הטבלה *top-down* החל מ($F(1, n, 1, m)$)

סיבוכיות זמן: $O(n^2m^2)$.

אלגוריתמים למחוזות / סיכום מאות אורי שביט

אלגוריתם אסטרטגיה מהיר יותר עבור ED tree

R_s – העץ ימני ביותר בס.

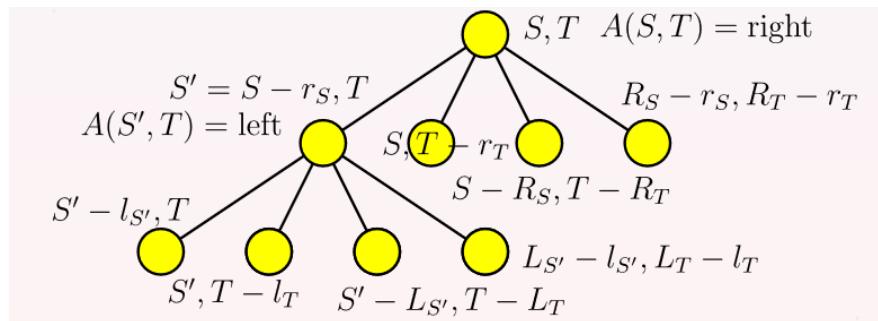
r_s – השורש של R_s .

$$ED(S, T) = \min \begin{cases} ED(S - r_s, T) + 1 \\ ED(S, T - r_t) + 1 \\ ED(S - R_s, T - R_t) + ED(R_s - r_s, R_T - r_T) + \delta(r_s, r_t) \end{cases}$$

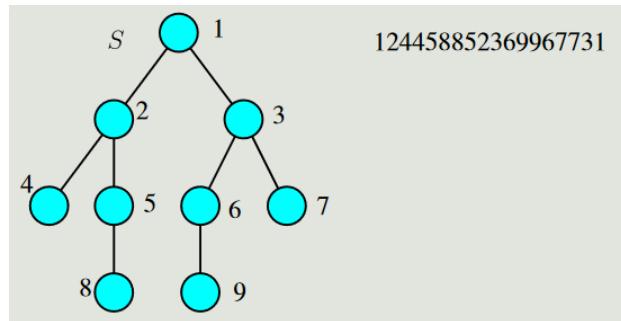
L_s – העץ השמאלי ביותר בס.
 l_s – השורש של L_s .

$$ED(S, T) = \min \begin{cases} ED(S - l_s, T) + 1 \\ ED(S, T - l_t) + 1 \\ ED(S - L_s, T - L_t) + ED(L_s - l_s, L_T - l_T) + \delta(l_s, l_t) \end{cases}$$

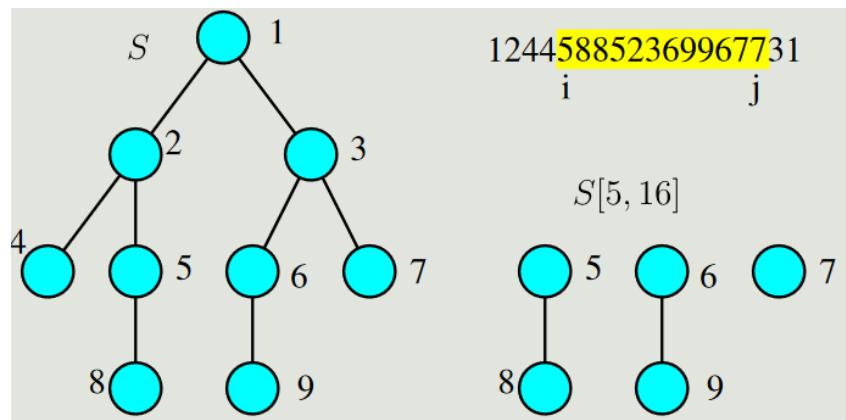
הגדרה: הוא אלגוריתם רקורסיבי שמחשב את $ED(S, T)$ ברקורסיה ע"י
שיםוש בLeft-right rule and the Right-left rule. (בכל צומת נחליט האם להשתמש בחוק הימני או השמאלי)



הגדרה: Euler string של עיר S היא מחזורת המתקבלת כאשר מבצעים DFS ledt-to-right של מחוזות S וכתיבת השמות של כל צומת פעמיים: כאשר DFS נכנס פעם ראשונה לצומת וכאש יוצא ממנה.



הגדרה: $[i, j]$ הוא יער שני האינדקסים שלו במחזורת האולר נמצאים בין i ל j עבור S .



אלגוריתמים למחוזות / סיכום מאות אורי שביט

למה: אם F הוא *relevant subforest* של S vr כרך ש $[j,i]$ כלשהם.

מספר ה-*relevant subforest* של S vr הוא לכל היותר $2n^2$.

מספר ה-*relevant subforest* של T vr הוא לכל היותר $2m^2$.

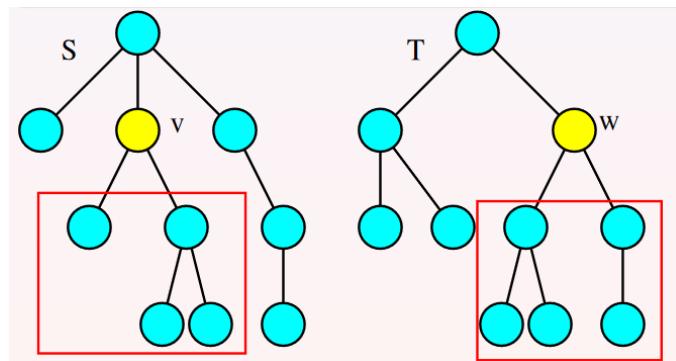
מספר ה-*relevant pair*vr הוא לכל היותר $4n^2m^2$.

סיבוכיות זמן לאלגוריתם האסטרטגי: $O(n^2m^2)$.

הגדרה:vr הוא S_v^o vr של S vr המכיל את היצאים של v .

למה:vr הוא *A*vr אלגוריתם אסטרטגיvr ו S, T vr 2 יערות.vr עבור כל קודקוד D vr כרך ש (S_v^o, T_w^o) vr הוא *relevant pair*vr.

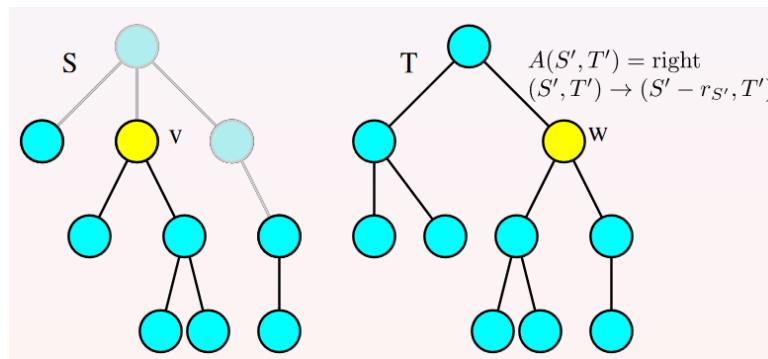
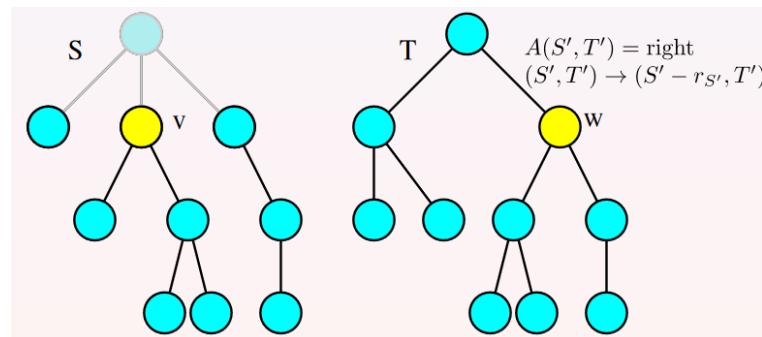
*ED(S, T)*vr בזמן חישבוvr עבורו אנו מחשבים את (S', T') vr כרך ש $ED(S', T')$ vr הוא זוג (S', T') vr.



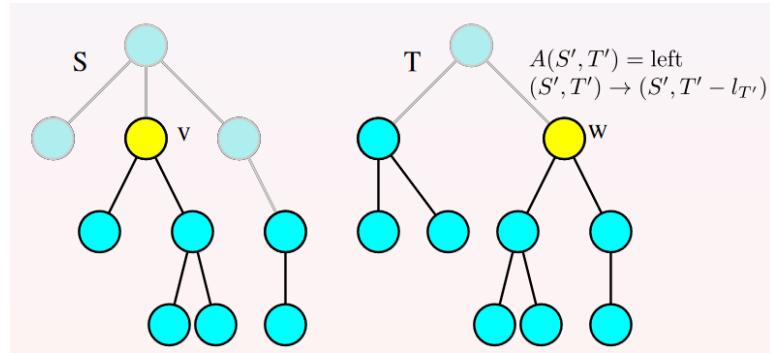
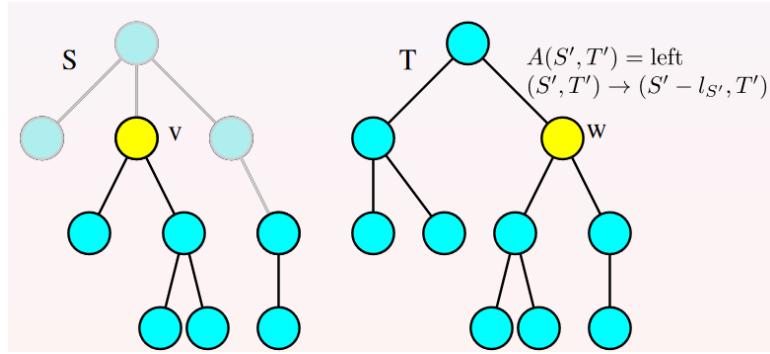
אלגוריתם

אתחלול:vr $S' = S, T' = T$

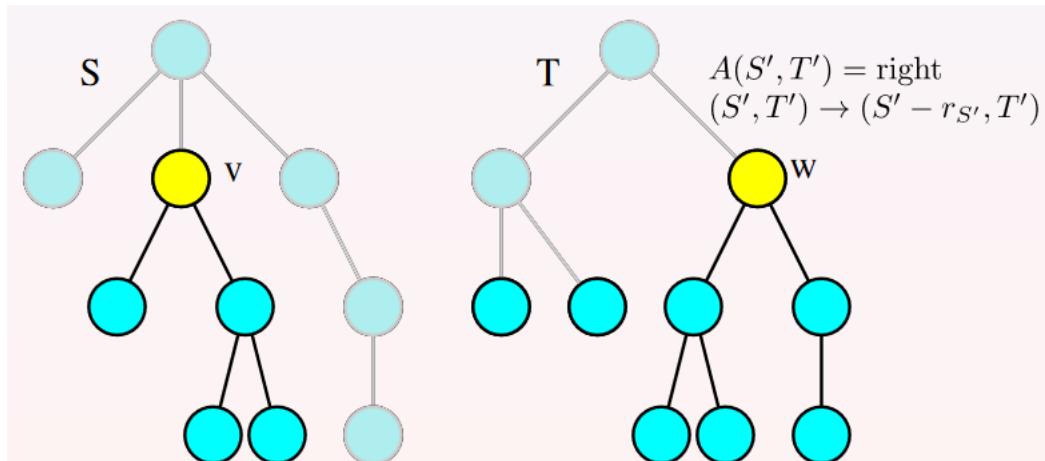
צעד:vr נמחק את השורש הימני/השמאלי ביוטר של T vr או S vr כאשר בחירת היער ממנה נמחק נמנעת ממחיקת v vr ואו המשך צעד.



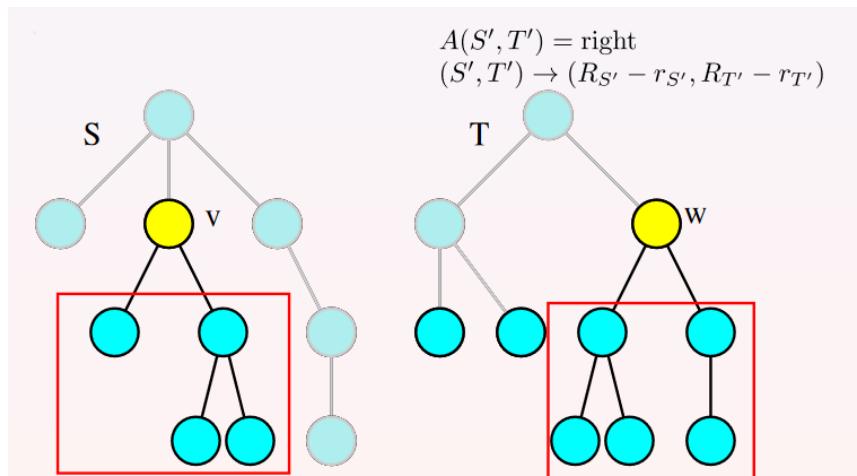
אלגוריתמים למחוזות / סיכום מאות אורי שביט



...

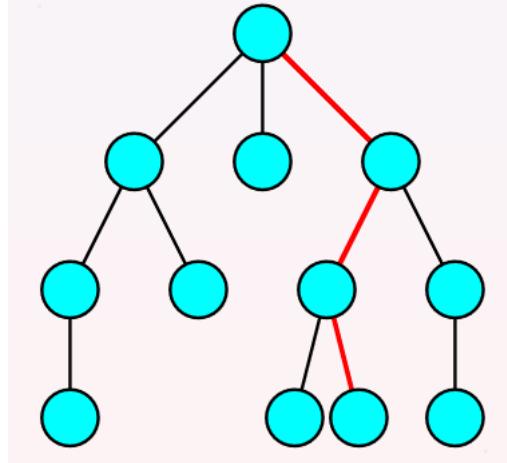


כאשר אנו לא יכולים למחוק יותר שורש, ניקח את הזוג $(R_{S'} - r_{S'}, R_{T'} - r_{T'})$

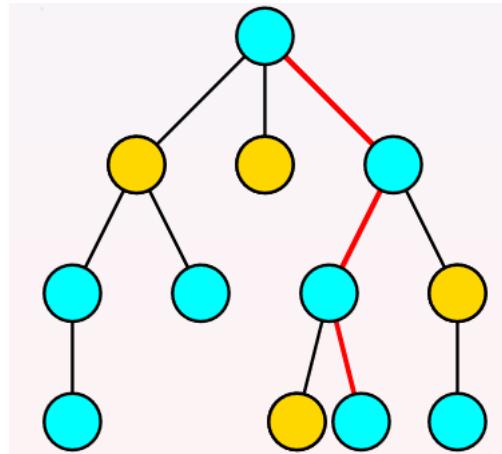


Heavy path

Heavy path של יער S הוא מסלול אשר מתחילה בשורש של העץ הגדול ביותר וועובר בכל קודקוד שיליד שלו שהחתה יער שלו מכיל יותר קודקודים.



$H(S) = \text{כל הקודקודים } v \text{ אשר } v \text{ הוא לא מסלול כבד של } S \text{ ושההורה של } v \text{ הוא חלק ממסלול כבד.}$

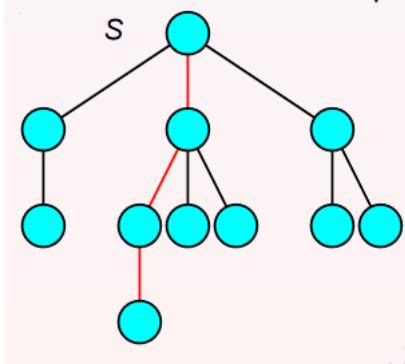


- למה:
1. אם $(S_v \in H(S) \text{ ו } |S_v| < \frac{|S|}{2})$ תת עץ של S המכיל את v ואת היצאים שלו
 2. $\sum_{v \in H(S)} |S_v| < |S|$

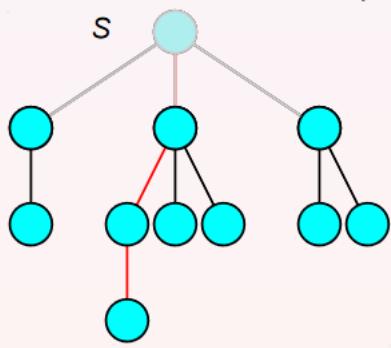
Algorithm for tree edit distance

1. אם $|T| < |S|$ החלף את S ו- T
 2. חשב מסלול כבד של S
 3. חשב $ED(S, T)$ בעזרת חוקי ימין/שמאל.
- כasher מחשבים (S', T') אם S' מכיל קודקודים של מסלול כבד אז:
1. אם v_S הוא לא קודקוד של מסלול כבד, השתמש בחוק ימין לחישוב $ED(S', T')$.
 2. אחרת השתמש בחוק שמאל לחישוב $ED(S', T')$.

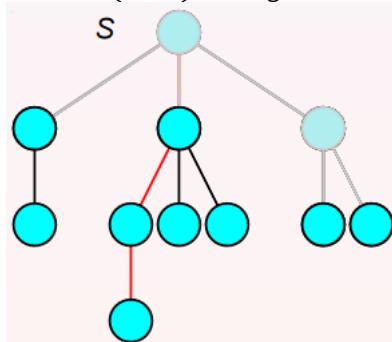
$A(S', T') = \text{left}$



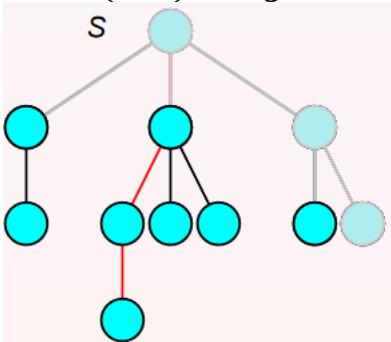
$A(S', T') = \text{right}$



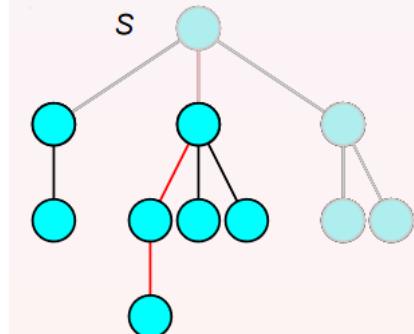
$A(S', T') = \text{right}$



$A(S', T') = \text{right}$

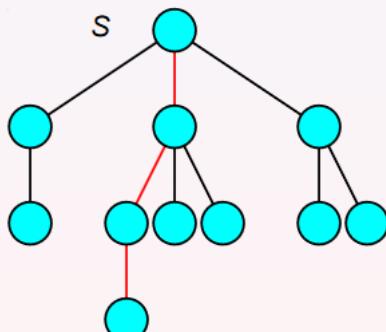


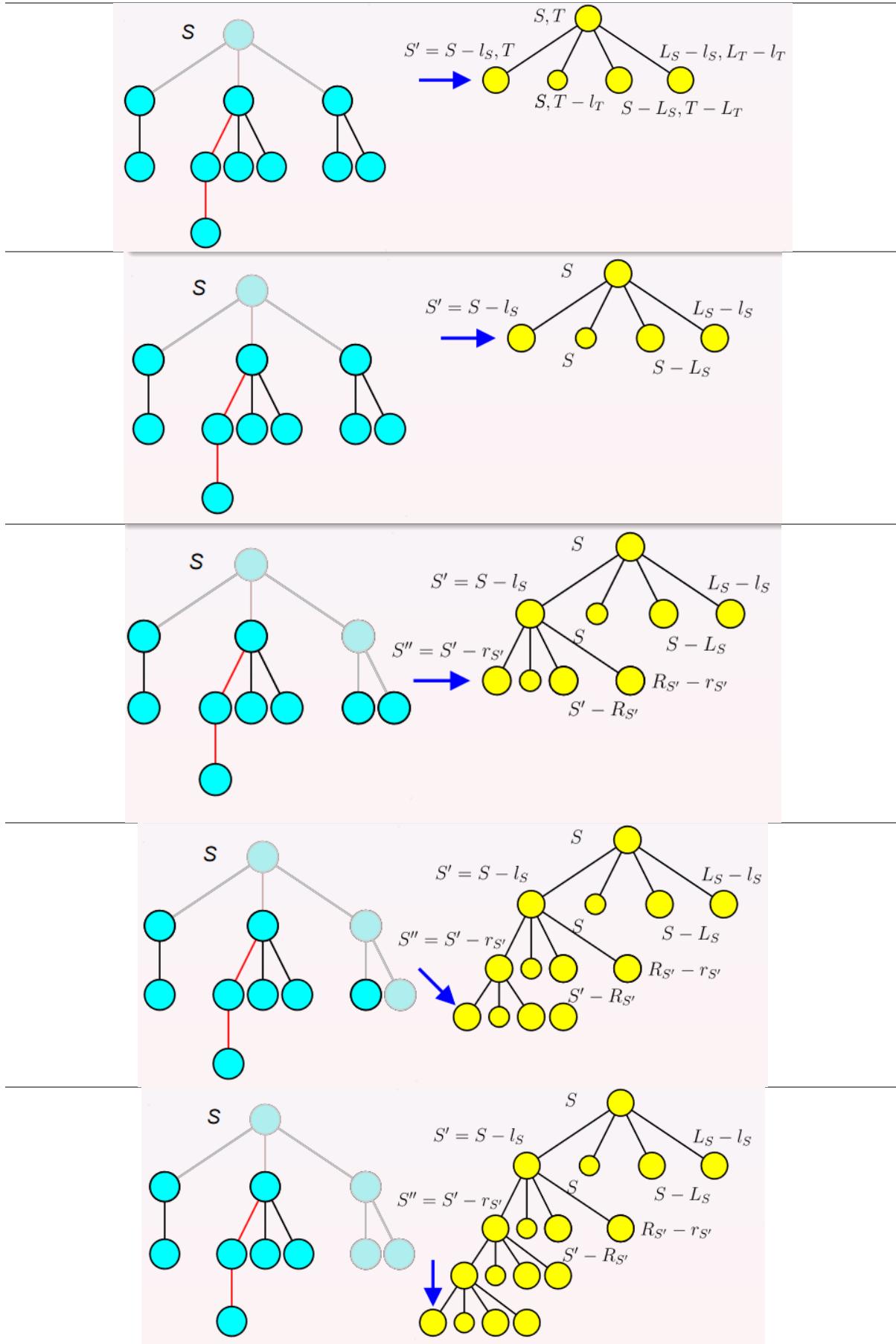
$A(S', T') = \text{left}$



עובדה: יש $|S|$ תת יערות של S המכילים קודקודים של מסלול כבד.

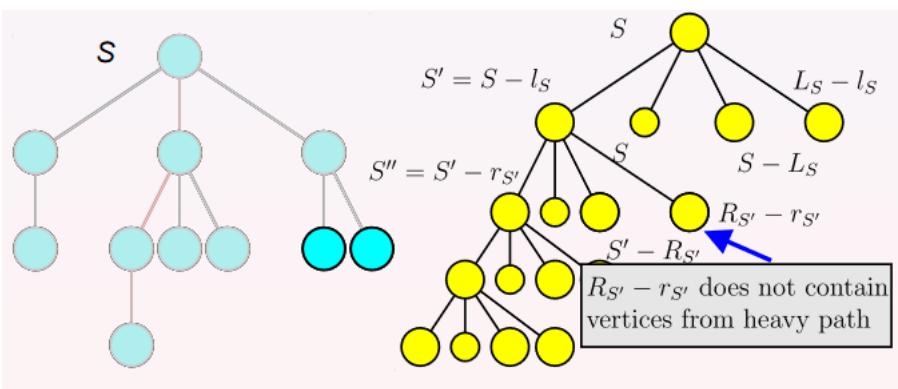
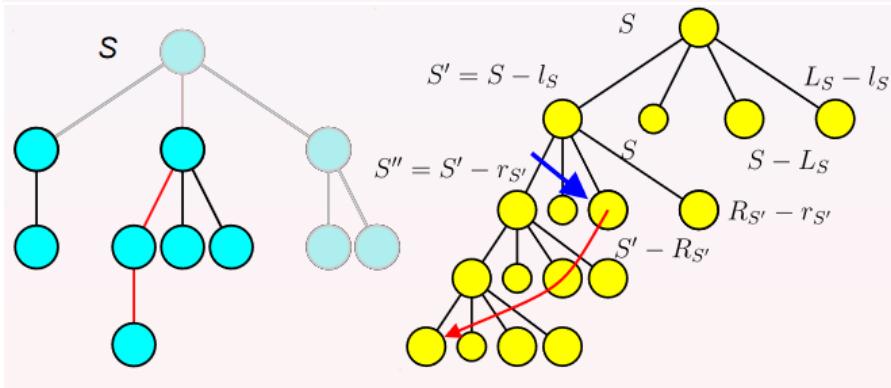
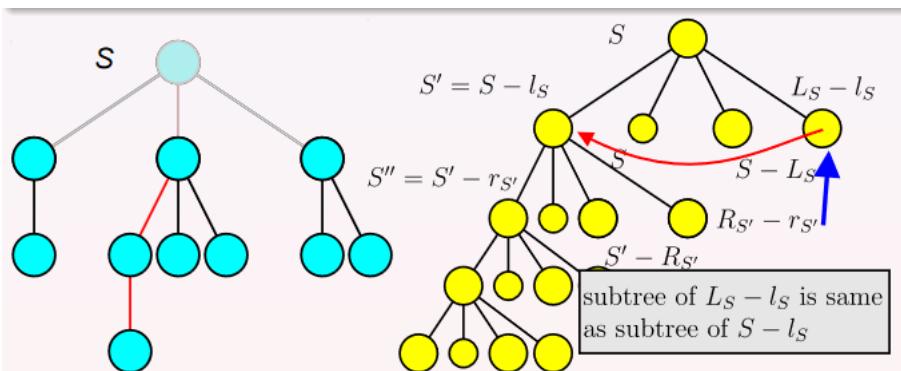
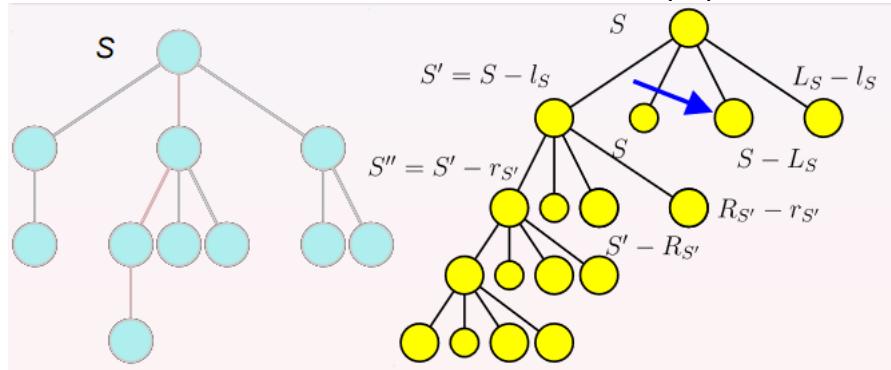
S, T

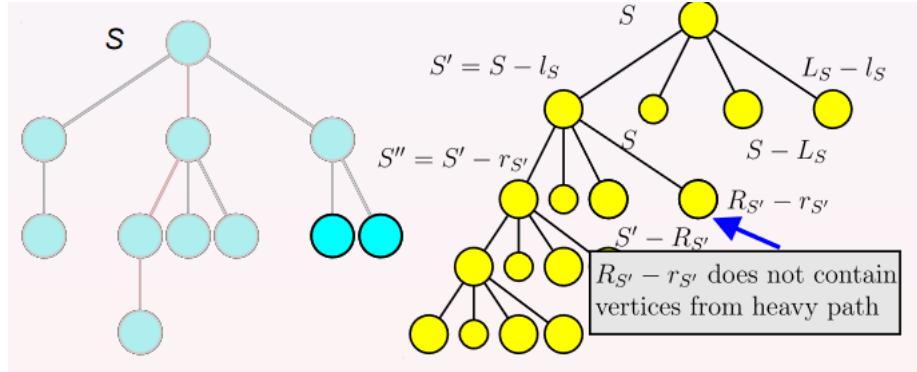




אלגוריתמים למחוזות / סיכום מאות אורי שביט

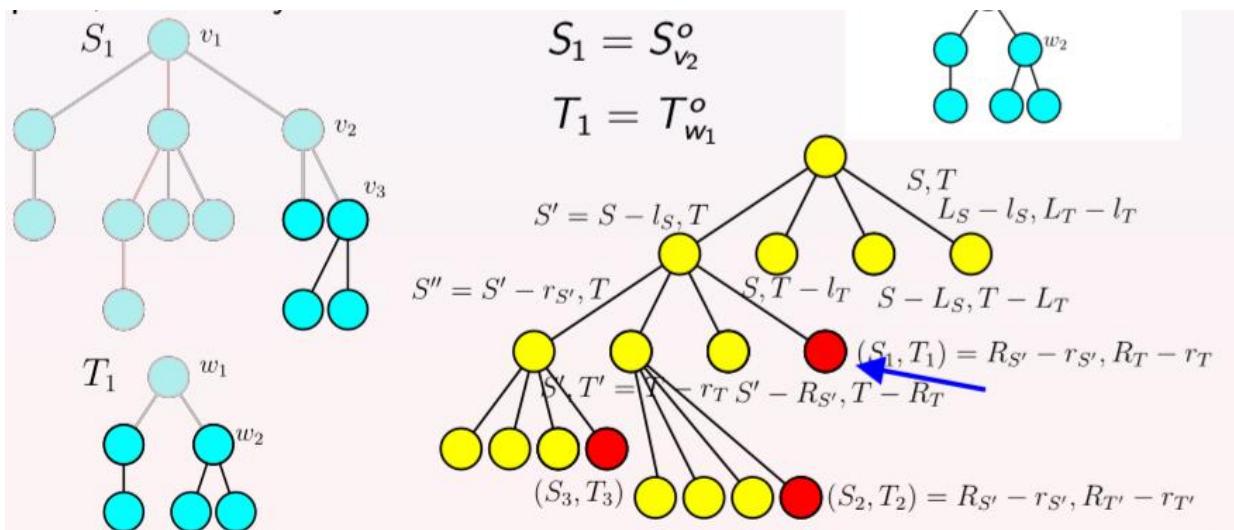
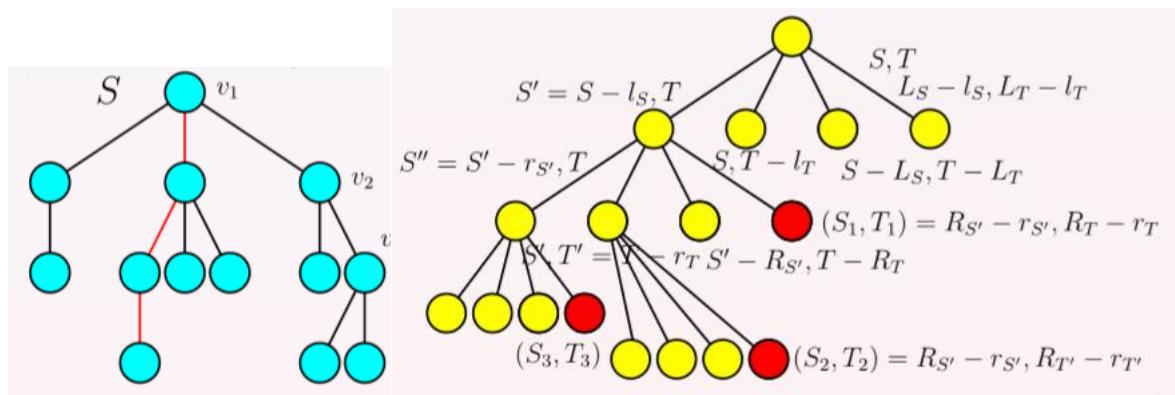
וכך הלאה עד שנוריד את כל הקודקודים.

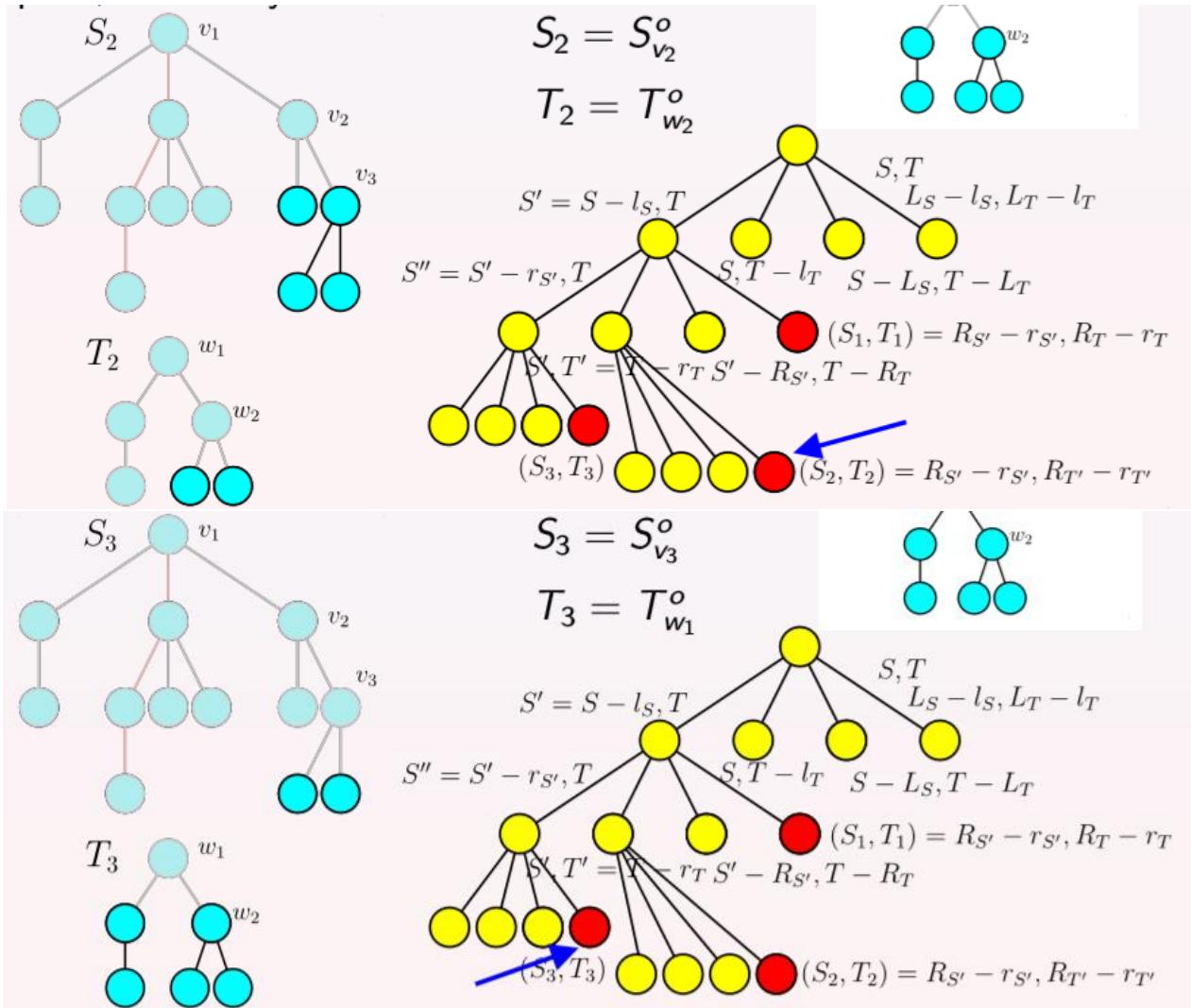




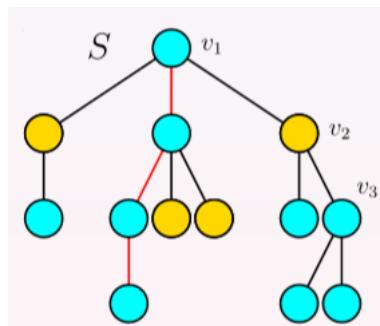
יש לפחות $|T||S|^2$ זוגות רלוונטיים (S', T') עבורם S' מכיל קודקוד של מסלול כבד. הדבר שלא התייחסנו אליו באlgorigthm הוא אם S' לא מכיל קודקודים של מסלול כבד!

Handling the remaining relevant pairs
הזוגות הרלוונטיים (S', T') עבורם לא חישבנו את (S', T') הם $ED(S', T')$ הם $S \in \mathcal{V}$ ו- $T \in \mathcal{W}$



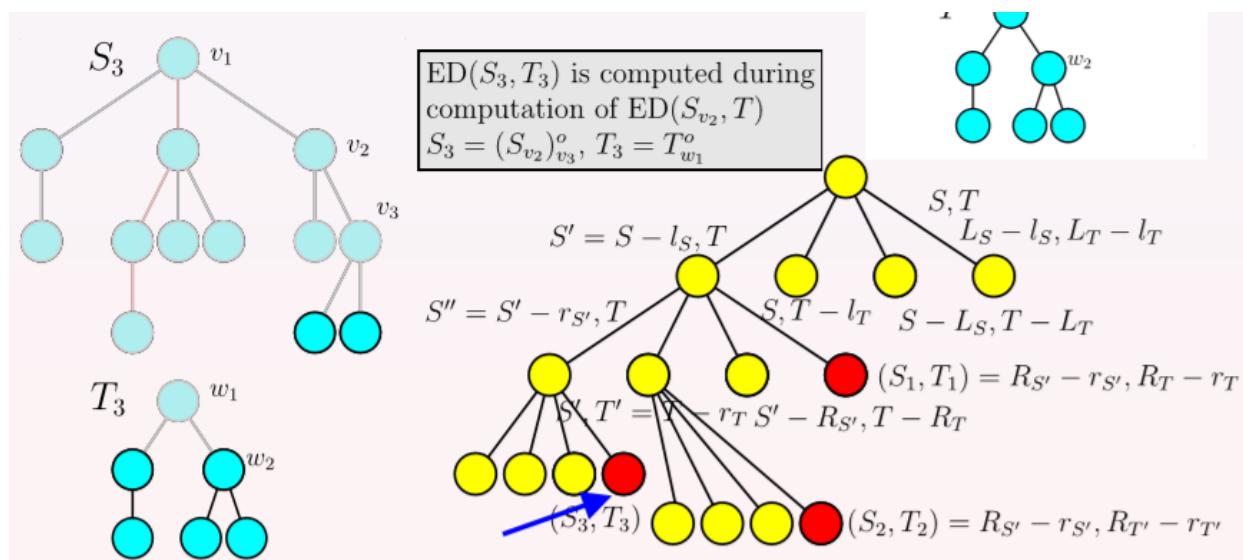
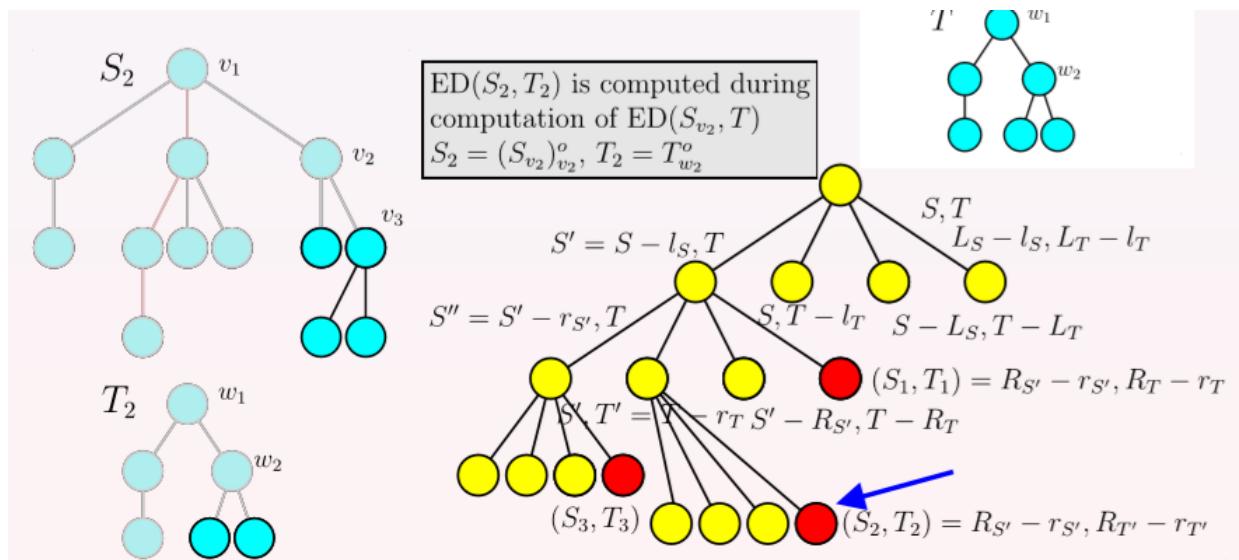
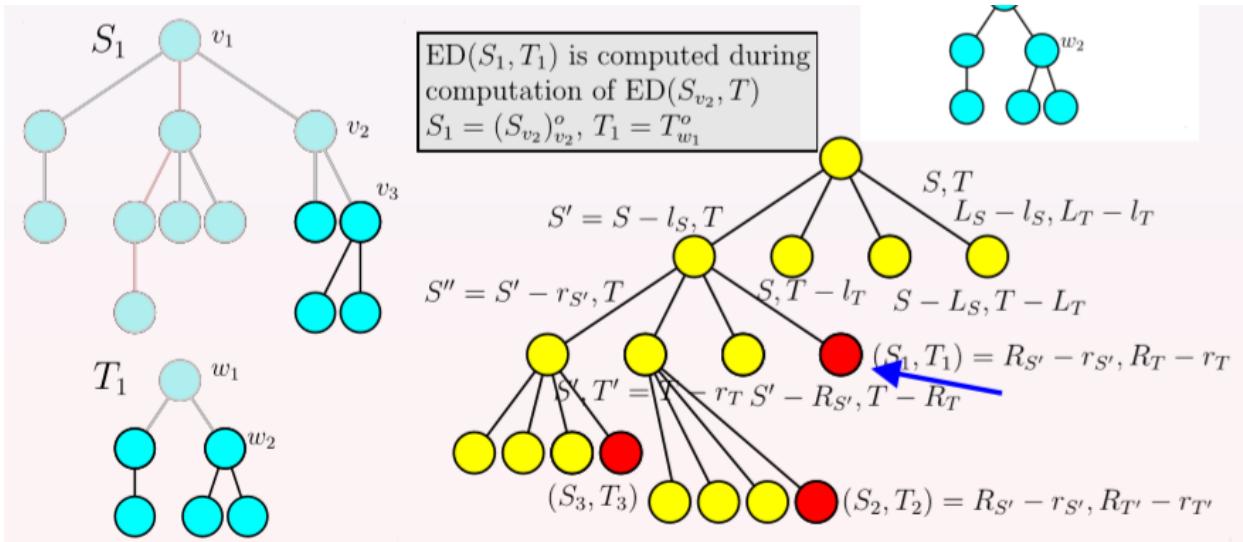


אם מחשבים את (S_v^o, T_w^o) הוא זוג רלוונטי לכל $v \in S$ ו $w \in T$, נחשב $ED(S, T)$ בעזרת רקורסיה.



$.S_1 = (S_{v_2})_{v_2}^o, T_1 = T_{w_1}^o$, $ED(S_{v_2}, T)$ מחושב בזמן חישוב $ED(S_1, T_1)$

אלגוריתמים למחuzeות / סיכום מאת אורי שביט



Algorithm for tree edit distance

1. אם $|T| < |S|$ החלף את S ו- T
 2. חשב מסלול כבד של S
 3. לכל קודקוד $v \in H(S)$ חשב $ED(S_v, T)$ בעזרת רקורסיה.
 4. חשב $ED(S, T)$ בעזרת חוקי ימין/שמאל.
- כאשר מחשבים $ED(S', T')$ אם S' מכיל קודקודים של מסלול כבד אז:
1. אם v_S הוא לא קודקוד של מסלול כבד, השתמש בחוק ימין לחישוב $ED(S', T')$.
 2. אחרת השתמש בחוק שמאל לחישוב $ED(S', T')$.
- אחרת ערך של $ED(S', T')$ כבר חושב בשלב 3.

ניתנו

$k(S) =$ כל הקודקודים v אשר v הוא לא מסלול כבד של S ושהורה שלו v הוא חלק מסלול כבד.

$k(S, T) =$ מספר הזוגות הרלוונטיים (S'_v, T') של S ו- T .

למה:

$$k(S, T) \leq \begin{cases} |T|^2|S| + \sum_{v \in H(S)} k(S_v, T) & \text{if } |S| \geq |T| \\ |S|^2|T| + \sum_{w \in H(T)} k(S, T_w) & \text{otherwise} \end{cases}$$

טענה: $k(S, T) \leq 4(|S| * |T|)^{1.5}$

הוכחה: נוכיח באינדוקציה על הגודל של $|T|$. אם $|S| \geq |T|$ אז

$$k(S, T) \leq |T|^2|S| + \sum_{v \in H(S)} 4k(S_v, T) \leq |T|^2|S| + \sum_{v \in H(S)} 4(|S_v| * |T|)^{1.5} =$$

$$|T|^2|S| + 4|T|^{1.5} * \sum_{v \in H(S)} |S_v| \sqrt{|S_v|} \leq |T|^2|S| + 4|T|^{1.5} * \sqrt{\frac{|S|}{2}} \sum_{v \in H(S)} |S_v|$$

אחרת קלומר $|T| \geq |S|$, נסתכל על המקרה השני

$$\begin{aligned} k(S, T) &\leq |T|^{3/2}|T|^{1/2}|S| + 4|T|^{3/2}\sqrt{|S|/2} \sum_{v \in H(S)} |S_v| \\ &\leq |T|^{3/2}|S|^{3/2} + 4|T|^{3/2}\sqrt{|S|/2} \cdot |S| \\ &= (1 + 4/\sqrt{2})(|S| \cdot |T|)^{3/2} \\ &< 4(|S| \cdot |T|)^{3/2} \end{aligned}$$

□

פרטי האלגוריתם

1. שומרים טבלה $F[i, j]$ ו- $ED[i', j'] = F((i, j), (i', j'))$ בין $[i, j]$ ו- $[i', j']$.
2. גודל הטלבה $\Theta(n^2m^2)$, למרות זאת אנו ממלאים $O(n^{1.5}m^{1.5})$ תאים ממנה.
3. סיבוכיות זמן היא $O(n^{1.5}m^{1.5})$ כאשר אנו מניחים $n \leq m$.

מצגת 15

Exact string matching

Input: A string T of length n , a string P of length $m \leq n$.**Output:** All locations i such that P appears in T starting at i .

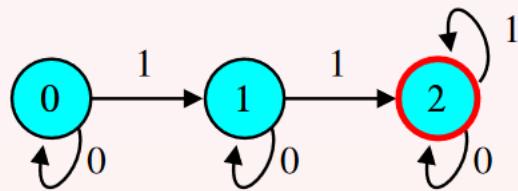
Example	Example	Example
$P = \text{ABCAAB}$	$P = \text{ABCAAB}$	$P = \text{ABCAAB}$
$T = \text{ABABCABAABCABAABAA}$	$T = \text{ABABCABAABCABAABCAABAA}$	$T = \text{ABABCABAABCAABCAABAA}$
Answer: 3, 7, 11	Answer: 3, 7	Answer: 3

אנו רוצים אלגוריתם יותר פשוט כי עצם סיווגו הוא מורכב.

נדיר שפה L כל המילים שמשיימות ב- P .נريץ את האוטומט על L כשנגייע למצב מקבל נגיד שיש מופע של P .**Definition**

A **deterministic finite automaton** is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, where

- Q is a set of **states**.
- Σ is an **alphabet**.
- $\delta : Q \times \Sigma \rightarrow Q$ is a **transition function**.
- $q_0 \in Q$ is a **start state**.
- $F \subseteq Q$ is a set of **accepting states**.



כל המילים שיש בהן לפחות 2 אחדים.

תזה L שפה 2 מילים י'א שקוילות אם'ם לכל z **Example**

For $\Sigma = \{0, 1\}$ and $L = \{x : x \text{ has } \geq 2 \text{ ones}\}$, there are 3 equivalence classes of \equiv_L ,

$$\begin{aligned} A_0 &= \{x : x \text{ has 0 ones}\} \\ A_1 &= \{x : x \text{ has exactly 1 one}\} \\ A_2 &= \{x : x \text{ has } \geq 2 \text{ ones}\} \end{aligned}$$

שפה היא רגולרית אם ו רק אם יש לה מספר סופי של מחלקות שקולות. ניתן לבנות אוטומט נורמלי שכמויות המ מצבים היא ככמויות מחלקות השקולות.

Theorem

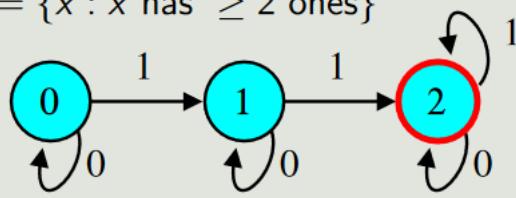
A language L is regular if and only if \equiv_L has finite number of equivalence classes.

Theorem

If L is regular, a minimum automaton accepting L can be constructed by defining Q to be the set of all equivalence classes of \equiv_L , and $\delta([x]_{\equiv_L}, a) = [xa]_{\equiv_L}$ for each equivalence class $[x]_{\equiv_L}$ and each $a \in \Sigma$.

Example

$\Sigma = \{0, 1\}$, $L = \{x : x \text{ has } \geq 2 \text{ ones}\}$



3 מצבים – 3 מחלקות שקולות 0 אחדים, 1 אחד, לפחות 2 אחדים.

שפה כל המילים שמסתיימות ב P .

נגדיר : $\alpha(x) = \text{longest suffix of } x \text{ which is a prefix of } P$.

$$P = ababc. \alpha(acabab) = abab$$

Theorem

$x \equiv_{L_P} y$ if and only if $\alpha(x) = \alpha(y)$.

Proof.

(\Rightarrow) Suppose that $\alpha(x) \neq \alpha(y)$. W.l.o.g. $|\alpha(x)| > |\alpha(y)|$. Let z be the suffix of P of length $|P| - |\alpha(x)|$. Then $xz \in L_P$ and $yz \notin L_P$. Therefore $x \not\equiv_{L_P} y$.

(\Leftarrow) Suppose that $\alpha(x) = \alpha(y) = x'$.

Let z be a string s.t. $xz \in L_P$, namely P is a suffix of xz .

Case 1: $|z| \geq |P|$.

It is clear that $yz \in L_P$.

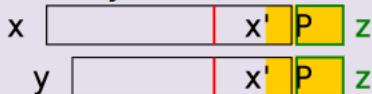
x				x'		P		z
y				x'		P		z

Case 2: $|z| < |P|$.

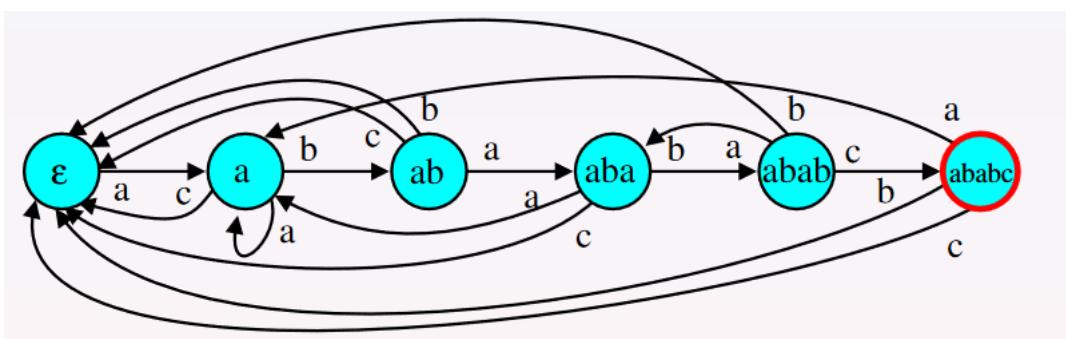
$x = x_1x_2$ where $x_2z = P$.

By definition, $|x_2| \leq |\alpha(x)|$, so the suffix of y of length $|x_2|$ is equal to x_2 .

Thus, $yz \in L_P$.



דוגמא: $p=ababc$

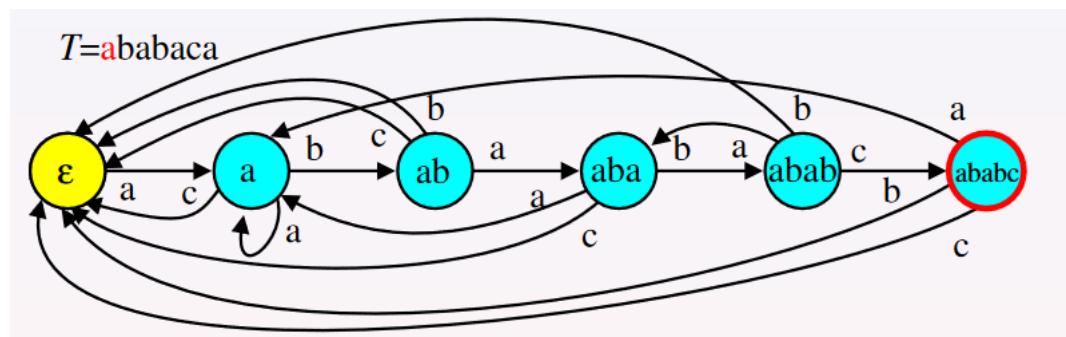


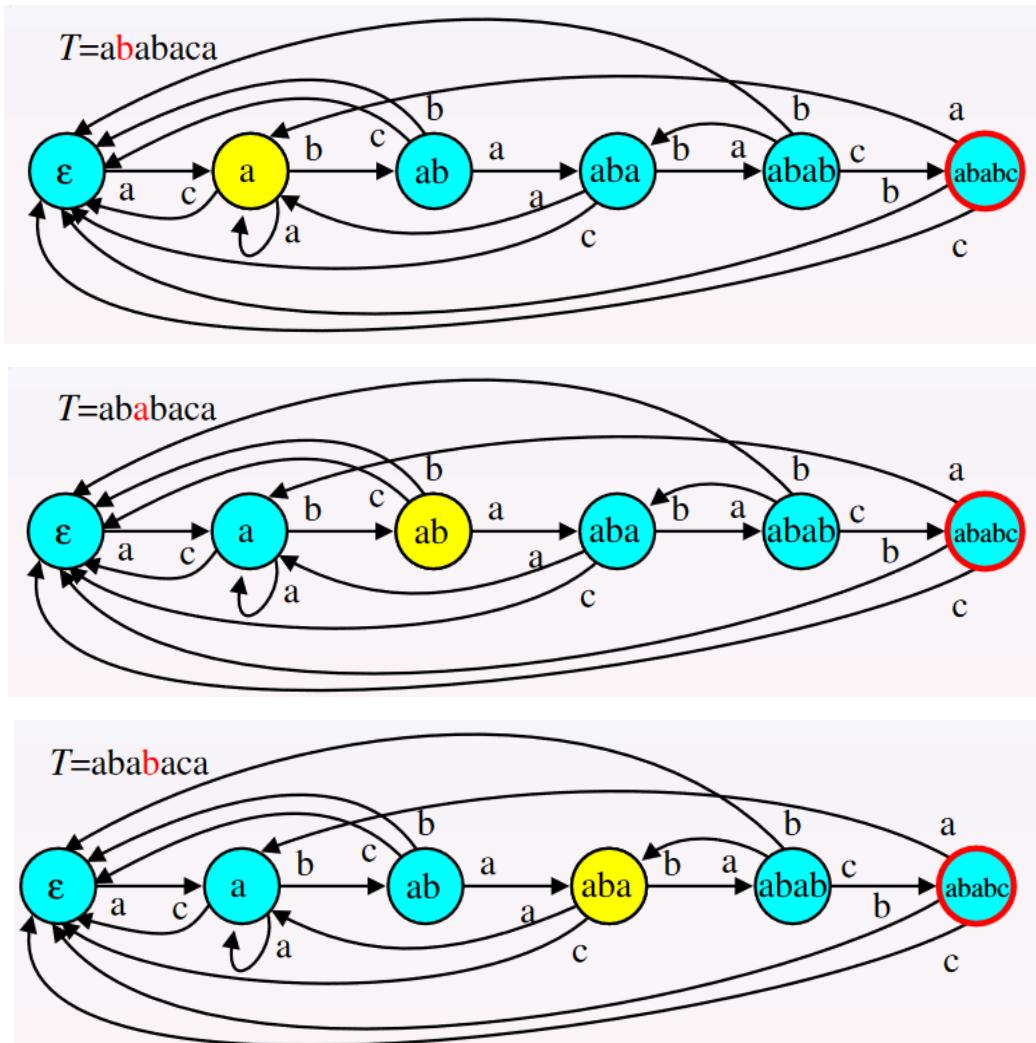
ניצר מצב לכל מחלקת שキילות, מחלקת שקיילות נקבעת לפי פונקציית α ולכן עוברים על כל הרישות של P .

$$\alpha(ababa) = aba, \alpha(ababb) = \epsilon, \alpha(ababc) = ababc$$

- There is a state q for every prefix of P .
- For state q and every $a \in \Sigma$, $\delta(q, a) = \alpha(qa)$.
- To find the occurrences of P in T , scan the characters of T and move to the appropriate states.
Every time state P is reached report an occurrence of P in T .

דוגמא ריצה:





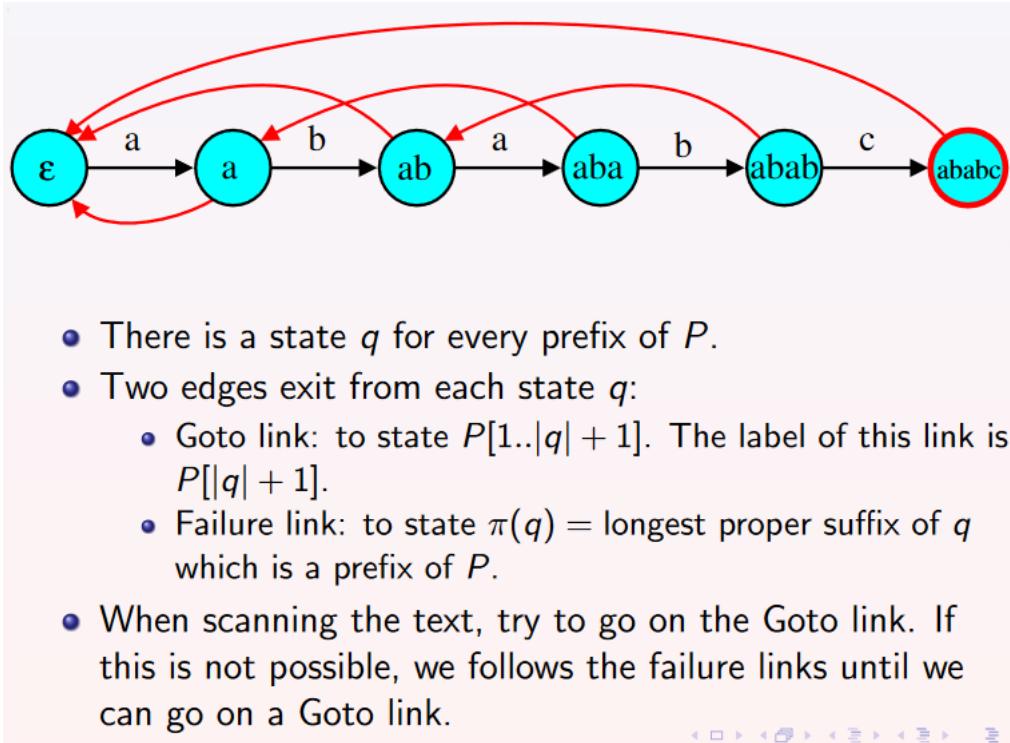
סיבוכיות

- זמן סריקת הטקסט ממצב למצב $O(n)$ מסpter המ מצבים באוטומט הוא $1+m$ כי כוללים גם את המילה הריקה ולכן מספר הקשרות הוא $\Sigma^{*(m+1)}$ ולכן ניתן לבנות את האוטומט בזמן $(|\Sigma| m)^{\Omega}.$

הפתרון לבעה נקרא **KMP** ובונים משהו שהוא דמיי אוטומט:

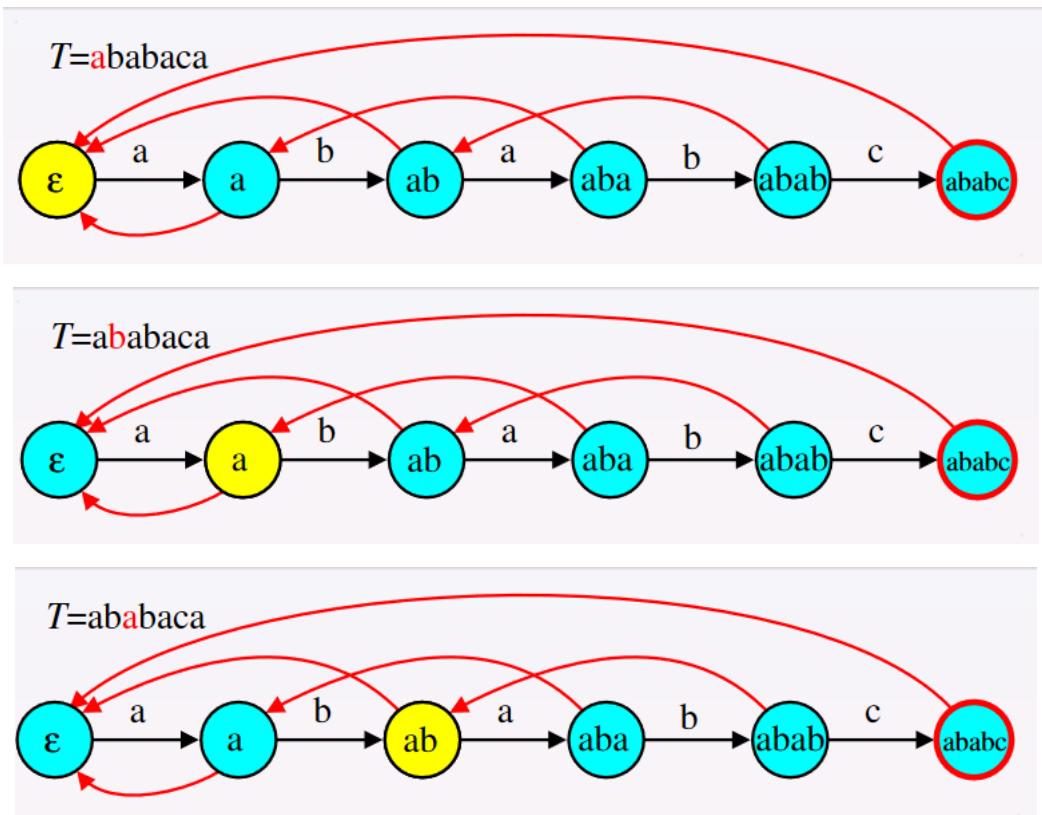
- מצב עبور כל רישא של P

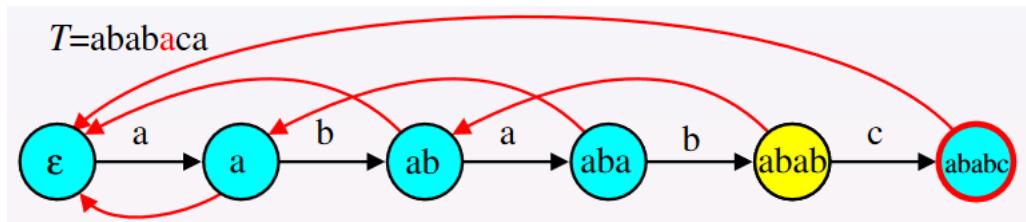
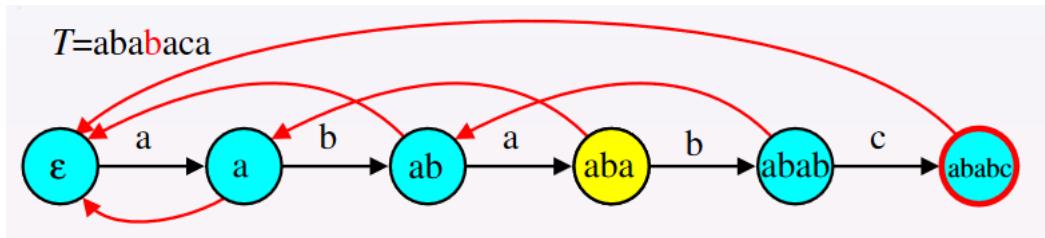
מעברים שחורים – רגילים כמו בקודם מעבורים *goto*
מעברים אדומים – יצאת קשת אחת בדיק – קשת האדומה אומרת שם אנו ב q נעברו לסיפא הארוכה ביותר של של q שהוא רישא של P .



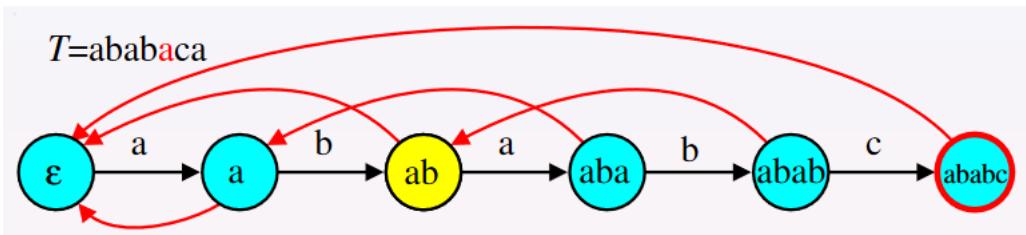
cut air MRIZIM AT haotomut?

נקראתו מ- T ונעבור למצב, תמיד ננסה ללכת בעזרת קשתות *goto*

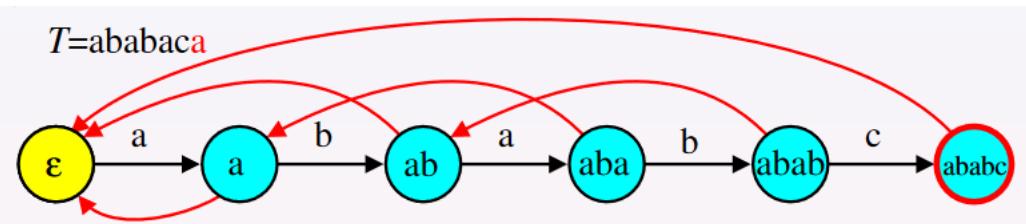
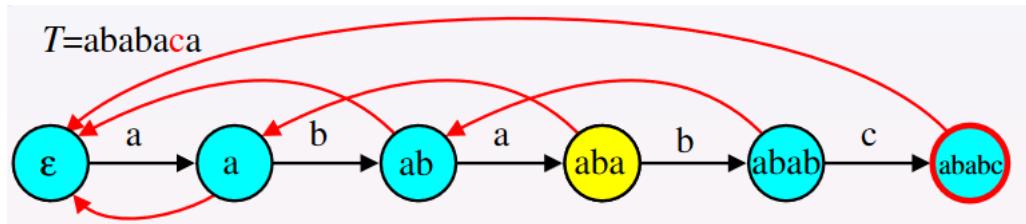




cut נקרא a ولكن נעבור בקשת אדומה כי לא ניתן לעبور עם c .



שוב נרצה להתקדם אבל לא ניתן כי רשם c ולרך עם הקשת האדומה.



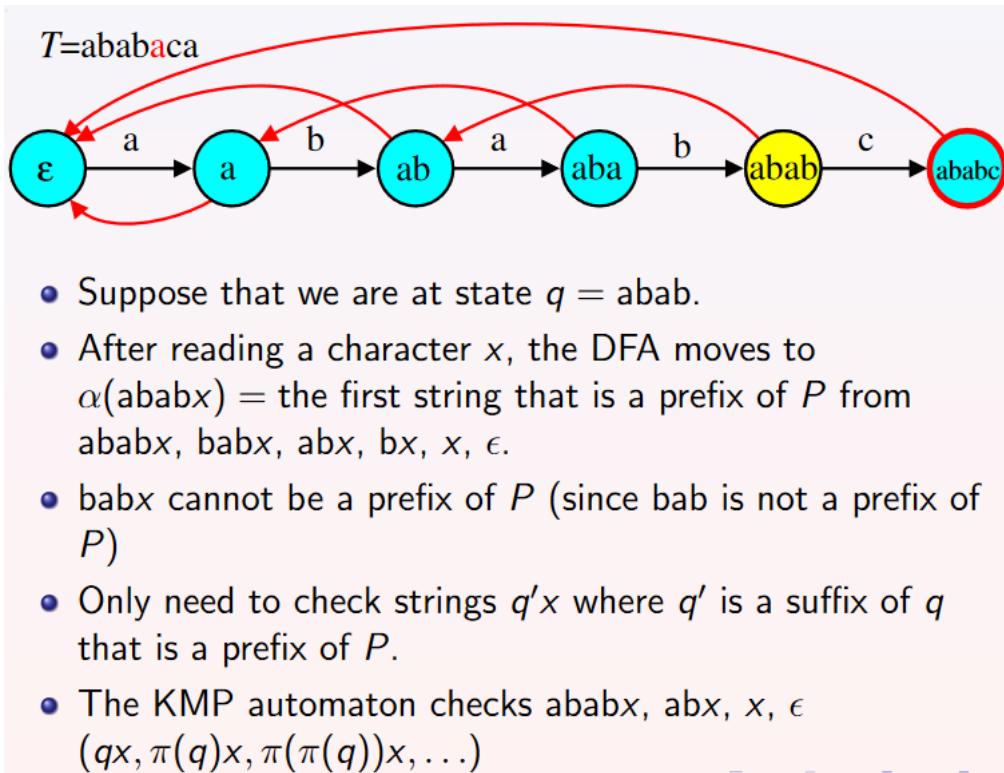
הרנו אוטומט, נרצה להראות שהאוטומט מסמלץ את האוטומט המקורי שהרנו קודם, ההוכחה באינדוקציה.

נניח שהרנו את $abab$ ואנחנו במצב המתאים, cut אם קוראים את התו x נעבור ל($ababx$) זהה הסיפה הארוכה ביותר שהיא רשא של P . (המעבר תלי ב-

נרצה למצוא את הריסאה הארוכה ביותר שהיא רשא של P : $.ababx,babx,abxbx,x,e$.

עבור על המחרוזות והראשונה שהיא סיפה.

נשメ לב שאנו יכולים לדלג על כמה בדיקות למשל ab לבדוק את המצב עבור $abab$



הKMP עובד שאמנו נמצאים abab בודקים האם רישא של P , אם לא אז אנו עוברים למצב $\pi(\text{abab}) = \text{abx}$ אם לא עוברים $\pi(\text{abx}) = \pi(\pi(\text{ab}))$ וכך הלאה עד לאפסילון. בדוגמה על הלוח עשינו את אותן בדיקות. קלומר האוטומט הריגל עושה את אותן בדיקות של KMP.

Definition

For a prefix q of P define:

$$\pi^*(q) = \{q, \pi(q), \pi(\pi(q)), \dots\}$$

$\beta(q) = \text{all suffixes of } q \text{ which are prefixes of } P.$

Theorem

$$\pi^*(q) = \beta(q)$$

Proof.

By induction on $|q|$. The base $q = \epsilon$ is trivial.

If $|q| > 0$ then

$$\begin{aligned} \pi^*(q) &= \{q\} \cup \pi^*(\pi(q)) && \text{definition} \\ &= \{q\} \cup \beta(\pi(q)) && \text{induction} \\ &= \beta(q) \end{aligned}$$

Proof.

We need to prove that $A = \beta(q)$ where $A = \{q\} \cup \beta(\pi(q))$.

(\subseteq) Suppose $p \in A$. If $p = q$ then $p \in \beta(q)$.

Otherwise, p is a suffix of $\pi(q)$ which is a prefix of P .

Since p is a suffix of $\pi(q)$ and $\pi(q)$ is a suffix of q , it follows that p is a suffix of q .

Thus, $p \in \beta(q)$.

הסיפה של הסיפה היא סיפה של המחרוזת הראשונה.

Proof.

We need to prove that $A = \beta(q)$ where $A = \{q\} \cup \beta(\pi(q))$.

(\supseteq) Suppose $p \in \beta(q)$. If $p = q$ then $p \in A$.

Otherwise, p is a proper suffix of q which is a prefix of p .

Since both p and $\pi(q)$ are suffixes of q , and $|p| \leq \pi(q)$ (from the maximality of $\pi(q)$), it follows that p is a suffix of $\pi(q)$.

Thus, $p \in \beta(\pi(q))$. \square

סוף השורה P במקומ d . אם יש לנו מחרוזת q ויש לנו 2 סיפות כאשר אורךה מהשניה אז הקצהה סיפוא של האורךה.

מהמשפט הנ"ל נסוע נכונות האלגוריתם.

Corollary

If the KMP is at state q and it processes a , it moves to state $\alpha(qa)$. (KMP simulates the DFA)

Proof.

A symbol ' a ' moves the automaton to state q' where q' is the longest string in the set $\{xa : x \in \pi^*(q), P[|x|+1] = a\} \cup \{\epsilon\}$. By the theorem, $q' = \alpha(qa)$. \square

האלגוריתם

- ➊ Compute $\pi(q)$ for every prefix q of P .
- ➋ $q \leftarrow \epsilon$.
- ➌ **for** $i = 1$ to n **do**
- ➍ **while** $q \neq \epsilon$ **and** ($q = P$ **or** $P[|q| + 1] \neq T[i]$) **do**
- ➎ $q \leftarrow \pi(q)$
- ➏ **If** $P[|q| + 1] = T[i]$ **then** $q \leftarrow P[1..|q| + 1]$
- ➐ **If** $q = P$ **then** output $i - m + 1$

Time complexity: $|q|$ is always non-negative.

$|q|$ is increased by 1 at most n times (line 6).

At each execution of line 5, $|q|$ is decreased by at least 1.

The number of decrements is $\leq n$.

Therefore the time complexity is $O(n)$ (not including step 1).

בכל פעם שאנו מפעילים את שורה 6 מגדילים את אורך q ב1.

כל פעם שאנו מרים את **while** אנו זזים לאורך קשת אדומה ולכז מקטינים את q לפחות ב1.

האורך של q הוא תמיד אי שלילי, כאמור האורך מתחילה ב0 ובגדל לכל היותר n פעמים ולכז האורך של q הוא לכל היותר n .

לכן סיבוכיות הזמן הוא (n) .

Computing the failure function

Let P be the a string of length m .

- ➊ $\pi(P[1]) \leftarrow \epsilon$
- ➋ $q \leftarrow \epsilon$
- ➌ **for** $i = 2$ to m **do**
- ➍ **while** $q \neq \epsilon$ **and** $P[|q| + 1] \neq P[i]$ **do**
- ➎ $q \leftarrow \pi(q)$
- ➏ **if** $P[|q| + 1] = P[i]$ **then** $q \leftarrow P[1..|q| + 1]$
- ➐ $\pi(P[1..i]) \leftarrow q$

Time complexity: $O(m)$.

/https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching

https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_KMP

התאמת פרמטרים

אם יש לנו 2 מחוזות A ו- B אם תואמת L^B אם יש פונקציה חח"ע עבורה $B = f(A[1])f(A[2]) \cdots f(A[|A|])$

$$abac \sim bdbc \quad (f(a) = b, f(b) = d, f(c) = c)$$

The parametric matching problem is:

Input: A string T of length n , a string P of length $m \leq n$.

Output: All locations i such that $P \sim T[i..i+m-1]$.

Example

$P = ABAB$

$T = A\textcolor{red}{XYXYX}$

Answer: 2

Example

$P = ABAB$

$T = AX\textcolor{red}{YXYX}$

Answer: 2, 3

מוטיבציה – קוד – 2 קודים זהים:

```
x=y-z;
if (y>z)
    m=1;
h=f(x);
```

```
diff=b-c;
if (b>c)
    n=1;
h=f(diff);
```

The two code fragments above are equivalent.

ניתן לפתור בעיה זו ע"י אלגוריתם דומה לKMP

- To solve the parametric matching problem, we transform it a problem on **numeric strings**.
- For a string A define a string \hat{A} as follows: $\hat{A}[i]$ is the distance to the closest occurrence of $A[i]$ to the left of i , or i if there is no such occurrence.

Example

$A = \text{aabac}$, $\hat{A} = 11325$.

For two numeric strings A, B write $A \approx B$ if for every i , either $A[i] = B[i] < i$ or ($A[i] \geq i$ and $B[i] \geq i$).

Example

$A = 11325$

$B = 31426$

Let P, T be strings.

$P \sim T[i..i + |P| - 1]$ if and only if $\hat{P} \approx \hat{T}[i..i + |P| - 1]$.

Example

$P = \text{aabac}$ $T = \text{zx}\text{yyxyzz}$ $\hat{T} = 12\text{313261}$

$\hat{P} = 11325$

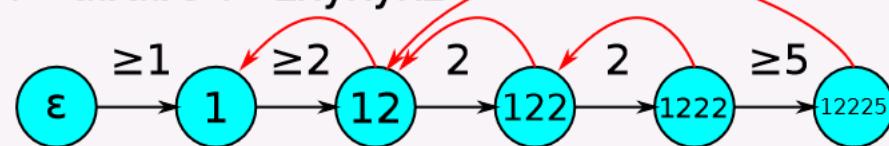
$\hat{T}[3..7] = 31326$

מה שנאנו נסתמך עליו היא העובדה – P תואם פרמטרים ל T מ"מ אם יש התאמה מסכירה.

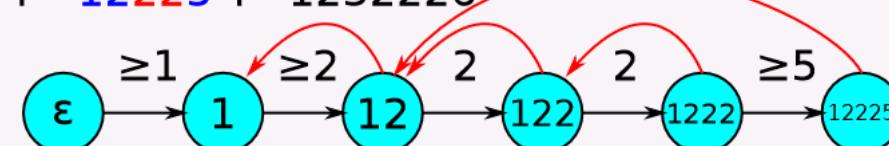
מה שניתן לעשות זה לבנות אוטומט בדומה לKMP – לוקחים את P ומחליפים אותו בהתאם לאות כובע.

מכל מצב ייצאות 2 קשיות: שחורה – התו המתאים ב P , עבר או תיירות

$P = \text{ababc}$ $T = \text{zxyxyxz}$

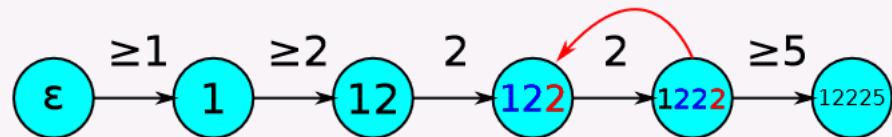


$\hat{P} = 12225$ $\hat{T} = 1232226$



- There is a state q for every prefix of \hat{P} .
- Two edges exit from each state q :
 - Goto link: to state $\hat{P}[1..|q| + 1]$. The label is $\hat{P}[|q| + 1]$ if $\hat{P}[|q| + 1] \leq |q|$, and $\geq \hat{P}[|q| + 1]$ otherwise.
 - Failure link: to state $\pi(q) = \text{longest proper suffix } q' \text{ of } q \text{ such that } q' \approx \hat{P}[1..|q'|]$.

$\hat{P} = 12225$ $\hat{T} = 1232226$



בכל פעם מנסים את מצב *goto* אחרית עוביים לקשות אדומות כמו ב-KMP.

דוגמת הרצה: 62 – 53

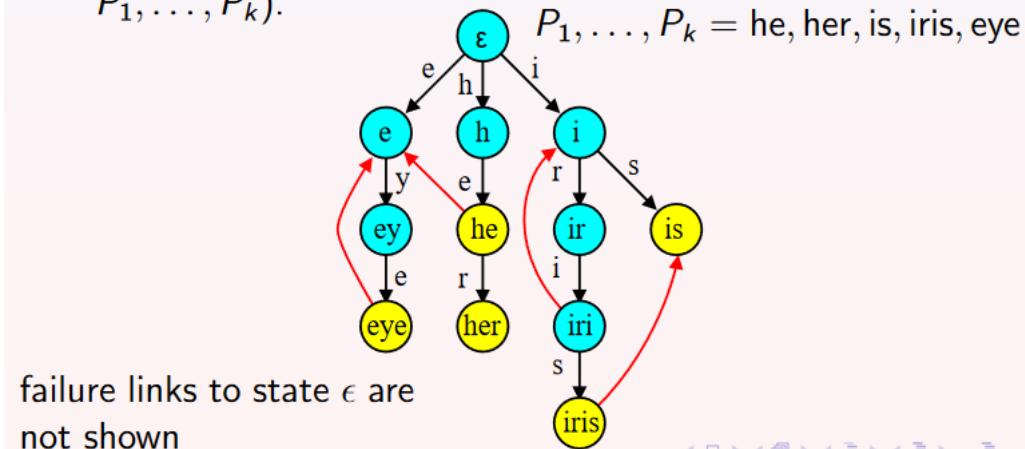
Input: Strings P_1, \dots, P_k and T .

Output: All occurrences of the strings P_1, \dots, P_k in T .

מקבלים k מחוזות ורוצים למצוא את כל המופיעים של k המחרוזות ב- T .

נרצה להכיל את האלגוריתם לכמויות המחרוזות:

- A state q for each prefix of some word from P_1, \dots, P_k .
- Goto links (from state p to state q if p is the prefix of length $|q| - 1$ of q)
- Failure links (from state p to state q , where q is longest proper suffix of p which is a prefix of a word from P_1, \dots, P_k).



- Let $m = \sum_i |P_i|$, $n = |T|$.
- Suppose that $\Sigma \subseteq \{1, \dots, m\}$.
- Constructing the Aho-Corasick automaton takes $O(m)$ time and space.
- Scanning the text takes $O(n \log |\Sigma| + \#occ)$ time, where $\#occ$ is the number of occurrences of P_1, \dots, P_k in T .
- Using hash tables, the scan time is $O(n + \#occ)$.

נייצר אוטומט שיש בו מצב לכל הרישאות של כל המחרוזות. כך שנכלי אותו לכל המילים. אם ניקח 2 רישאות עוקבות נעביר קשת שחורה בינהם כשאר התווית תהיה האות הנוספת.

אם מסתכלים רק על הקשתות השחורות קיבל עץ שהוא *trie* של קבוצת המחרוזות.

יש קשתות מכל הכהולים לאפסילון עבור המשך קריאה.

ההרצאה של האוטומט היא למשה כמו KMP, כל פעם מנסה ללקת בקשת שחורה כאשר עכשו יש מספר אופציות ללקת, כל פעם שלא נוכל ללקת קדימה נלך בקשת אדומה או אפסילון וכך הלאה.

זמן ריצה – נסמן בה את אורך T ובו אורך כל המחרוזות.

בנייה האוטומט לוקחת $O(m)$ זמן ומקום.

אלגוריתמים למחוזות / סיכום מאתורי שביט

סרייה ע"י טבלאת האש תיקח ($occ + n$) O כאשר אנו מגיעים למצב מקבל מדפיסים את המצב הנוכחי ולכן גם אם יש קשת אדומה למצב מקבל אחד לשני צריך להדפיס גם את השני.

זמן n ועוד מספר המופעים שיכול להיות גדול מה (מרקם קיצוניים) ולכן הריצה ($occ + m + n$) O . אורך קלטים ועוד מספר מופעים שהוא אופטימלי לבעה.

<https://www.geeksforgeeks.org/aho-corasick-algorithm-pattern-searching>

Example

$P = ABCAAB$

$T = ABABCABCABCABCABAA$

Answer: 3, 7, 11

The Shift-Or Algorithm 16

מקבילים מחוזות ומוצאים מופיעים של אחת בשניה. – *Exact string matching*

Input: A string T of length n , a string P of length $m \leq n$.

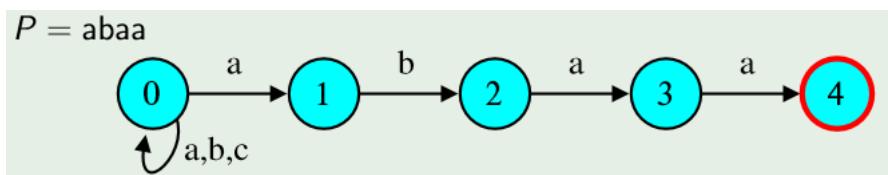
Output: All locations i such that P appears in T starting at i .

אלגוריתם KMP פותר את הבעיה ב(n). ישנו עוד אלגוריתם רבים הפוטרים את הבעיה:

- Morris-Pratt
- Knuth-Morris-Pratt
- Boyer-Moore
- Horspool
- Galil-Seiferas
- Apostolico-Giancarlo
- Karp-Rabin
- Zhu-Takaoka
- Optimal-Mismatch
- Maximal-Shift
- Quick-Search
- Anostolico-Crochemore
- Galil-Giancarlo
- Raita
- String-Matching on Ordered ALphabet
- Not-So-Naive Turbo-Boyer-Moore
- Reverse-Colussi
- Skip-Search
- Alpha-Skip-Search
- KMP-Skip-Search
- Berry-Ravindran
- Ahmed-Kaykobad-Chowdhury
- Wu-Manber for Single Pattern Matching
- Two-Sliding-Window
- Boyer-Moore-Horspool with q-grams
- Genomic Rapid Algo for String Pm
- SSEF
- Deterministic-Finite-Automaton
- Reverse-Factor Simon
- Turbo-Reverse-Factor
- Forward-DAWG-Matching
- Fast-Search
- Forward-Fast-Search
- Backward-Fast-Search
- Fast-Boyer-Moore
- Tailed-Substring
- Sheik-Sumit-Anindy Balakrishnan-Sekar
- Thathoo-Virmani-Sa Balakrishnan-Sekar
- Boyer-Moore-Horspo Probabilities
- Franek-Jennings-Sm
- 2-Block Boyer-Moor

– אוטומט לא דטרמיניסטי שמקבל את כל המילים שמותיימות ב- P . למשל מילים שמותיימות ב- $abaa$.
 NFA – אוטומט לא דטרמיניסטי שמקבל את כל המילים שמותיימות ב- P . למשל מילים שמותיימות ב- $abaa$.
 NFA המקובל שפה $P^* \Sigma^* = L_p$ הוא טרוייאלי.

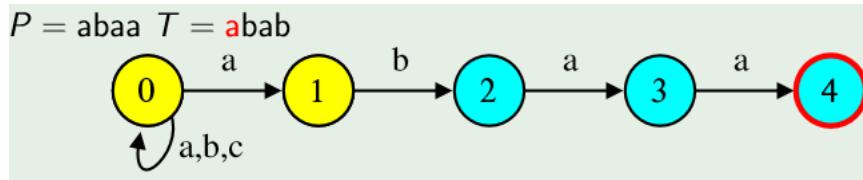
אנו רוצים לבצע את כל הריצות האפשרות עליו لكن נגדיר:



הגדרה: $A(S) = \{q \mid \text{שניתן להגעה אליו מה שטרם עבד את } S\}$. למשל

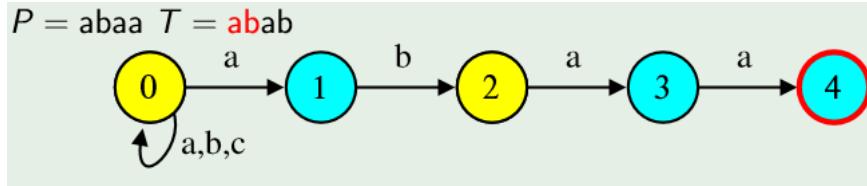
כל המצבים שמקבלים כמטרים את S באוטומט.

$$A(a) = \{0,1\}$$

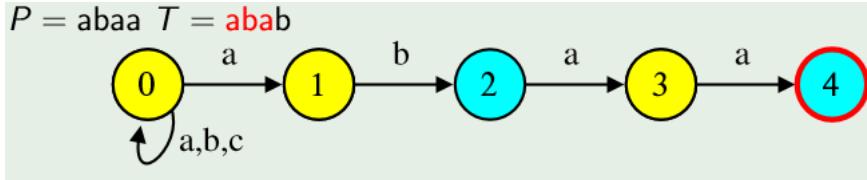


$$A(ab) = \{0,2\}$$

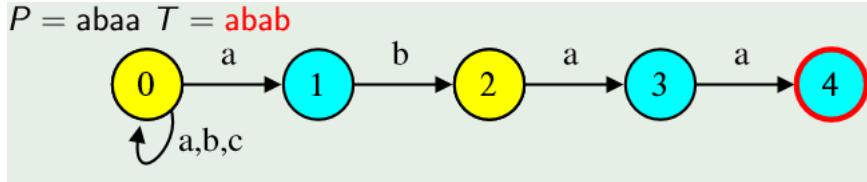
אלגוריתמים למחuzeות / סיכון מאות אורי שביט



$$A(aba) = \{0,1,3\}$$



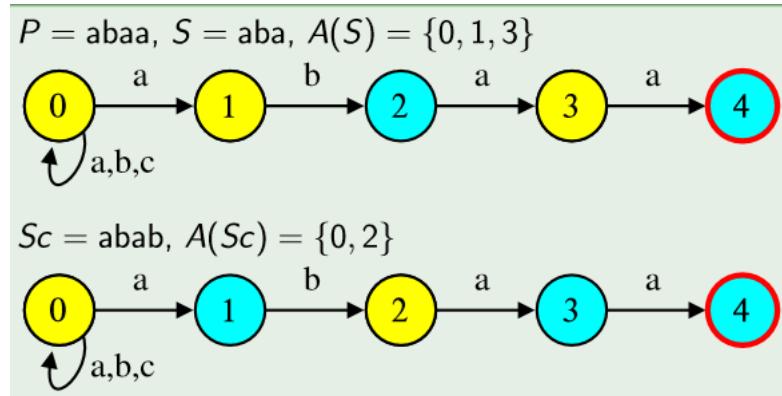
$$A(abab) = \{0,2\}$$



Computing the active states

נניח שהשיבו את $A(S)$ ונרצה לחשב את $A(Sc)$. אז החישוב הוא פשוט,

עובדת: $i \in A(Sc) \iff P[i] = c \text{ ו } i - 1 \in A(S)$.



מוסיף ל S את התו b , אז האפשרויות שיש היא בטיפול הוא רק מעבר של b או היישירות במצב 0.

Recall that $m = |P|$, $n = |T|$.

- ➊ **for** $i = 1, \dots, n$ **do**
- ➋ Compute the set $A(T[1..i])$
- ➌ **if** $m \in A(T[1..i])$ **then** print $i - m + 1$

- Computing $A(T[1..i])$ from $A(T[1..i-1])$ takes $O(m)$ time.
- Overall time complexity: $O(mn)$.
- The algorithm can be made faster with **bit-parallelism**.
- We assume a RAM machine with w bits in each word.

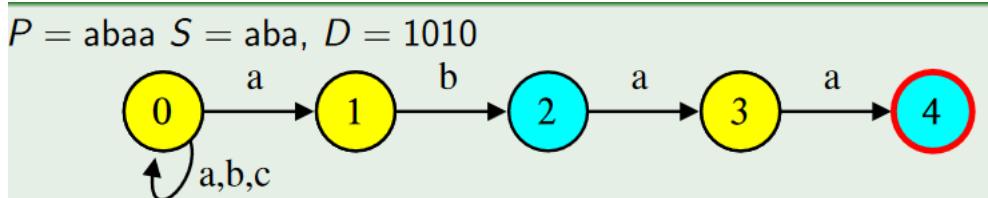
האלגוריתם הנ"ל לא יעיל כי מה שעושה עובר מ1 עד n מחשב את A של הרישא ואם המצב המתקבל שי"ר לקבוצה את מצאנו מופיע.

הבעיה היא שהאלגוריתם לא יועל כי מציאת שייכות יכולה לדרוש (m) O ולכן בעיית.

- ניתן להפוך את האלגוריתם לעיל ע"י כך שנשמר את A בזיכרון ונחשב את $[n]A[1] \dots [n]A[i]$.
1. נניח $w \leq m$ כלומר שוכול לשמר את המילה בתא זיכרון אחד.

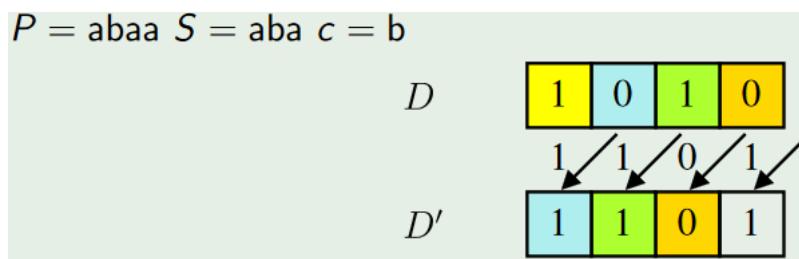
נשמר את המילה בתור וקטור. מצב 0 תמיד נמצא בקבוצה ולכן רק נשמר את המצבים האחרים.

אנחנו הולכים לרשום את הביטים מימין לשמאל(הפרק מהצ'יר) אם מצב הוא פעיל אז נשמר 0 ואחרת אם כחול נשמר 1.



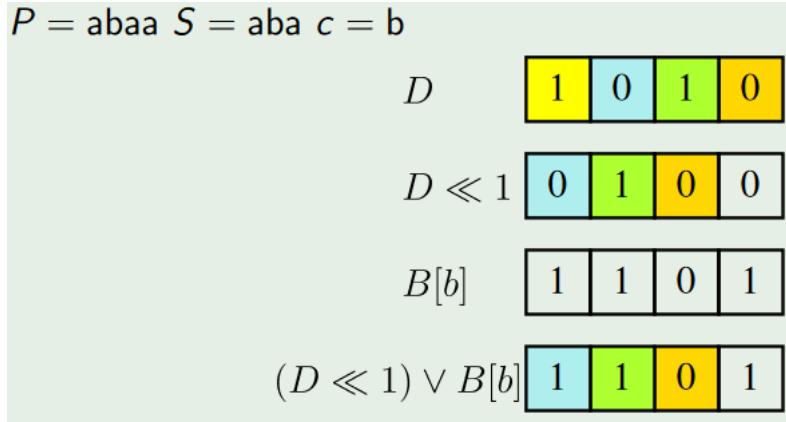
נניח ששמרנו את aba ונרצה לחשב את התו הבא:

יש לנו את הווקטור הישן: אם מסתכלים על בית מסוים של הווקטור החדש יהיה פעיל רק אם הביט הקודם ב-D היה 0 וגם התו היה מתאים כולם הקשת למעבר בין המצבים מתאימים. לכן ניקח את הווקטור D נזיז אותו שמאליה מקום אחד ולאחר כך לעשות OR לווקטור שנמצא הקשת באופןוטט.



כולם הווקטור הוא 1101 כי רק בקשת b יש התאמה (0).

חישוב בצורה מפורטת:



[b] מכיל אפסים בכל המיקומות שיש ב P תווי b .

מסיטים את D שמאלה עושים *shift*, עושים *OR* ומקבלים את התוצאה.

Pseudo code

```

① for  $i \leftarrow 1$  to  $|\Sigma|$  do  $B[i] \leftarrow \neg 0$ 
② for  $i \leftarrow 1$  to  $m$  do  $B[P[i]] \leftarrow B[P[i]] \wedge \neg(1 \ll i)$ 
③  $D \leftarrow \neg 0$ 
④ mask  $\leftarrow 1 \ll (m - 1)$ 
⑤ for  $i = 1$  to  $n$  do
    ⑥    $D \leftarrow (D \ll 1) \vee B[T[i]]$ 
    ⑦   if  $(D \wedge \text{mask}) \neq \text{mask}$  then print  $i - m + 1$ 

```

Assuming $m \leq w$, the algorithm solves the pattern matching problem in $O(n + \sigma)$ time.



זמן של שלב 1 זה $O(\Sigma)$

5-7 זמן לנארו $O(n)$

ולכן סה"כ הזמן $O(n + \sigma)$

דוגמה

$P = abaa$
 $B[a] = 0010, B[b] = 1101, B[c] = 1111, \text{mask} = 1000$

ניתן לייעל את האלגוריתם באופן מעשי:

מה שולקח הרבה זמן זה הולאות של 5 והלאה, נרצה לייעל זאת. דרך לייעל היא לעשות את הקוד פעמיים הפנימי כאשר הולאה תרוץ בכפולות של 2.

מייעל כיוון שאנו מבצעים פעולות – אם כל שורה לוקחת זמן אז במקורו יקח 4^k , בולולאה החדשה נבצע פעולות עבודה כי נבצע את הפנימי עם לפחות עבודה על *for*.

נקודה נוספת היא שבולאלת `for` יש שלב שני מסויים את `i` לה והוא שלב שלא יעל במעבדים.

For faster time in practice, the main loop can be unrolled:

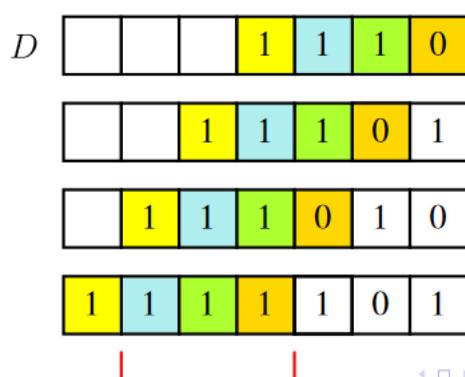
```

⑤ for  $i = 1, 4, 7, \dots$  do
⑥    $D \leftarrow (D \ll 1) \vee B[T[i]]$ 
⑦    $D \leftarrow (D \ll 1) \vee B[T[i + 1]]$ 
⑧    $D \leftarrow (D \ll 1) \vee B[T[i + 2]]$ 
⑨   if  $(D \wedge \text{mask}) \neq \text{mask}$  then report the matches

```

where `mask` contains 1 at bits $m, m + 1, m + 2$ and 0 elsewhere.

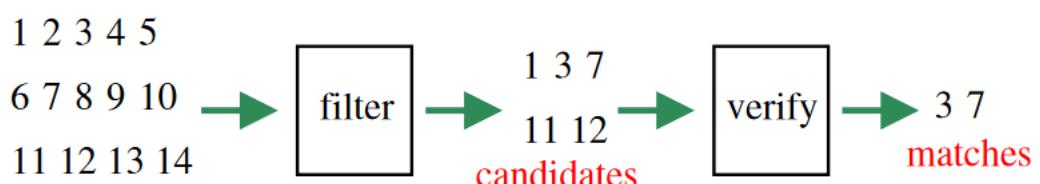
עושים בדיקה של `if` רק בסיום ولكن מה שמעניין אותנו כל פעם זה 3 ביטים האם הם 0.



בצורה זאת ניתן לחסוך הרבה זמן, הבעה היא שהשיטה מסורבלת, לטפל במקרה שהאורך לא מתחלק במספר הקפיצות.

חישוב נוסף של השיטה הוא שהקטינו את אורך המחרוזת שיכלנו לחפש, אורך השאלה יכולה להיות לא מתחלק בדיקות 64 ביט, כאן אנו צריכים לשמר 2 ביטים לאחר ביצוע 3 פעולות `shift` וכן מגביל את אורך השאלה.

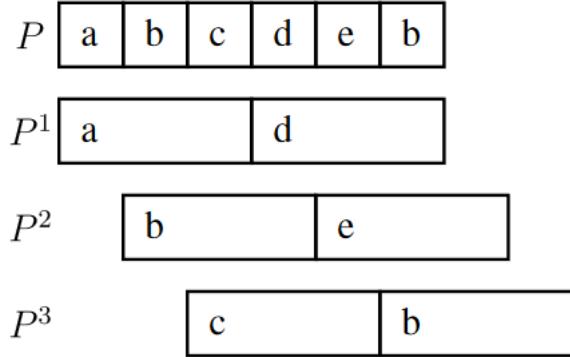
Filtering



רוצים למצוא את כל האינדקסים שבינם P מופיע. כל אינדקס הוא פוטנציאלי בהתחלה לפני בדיקה. ראשית נריץ שלב ראשון הוא יציג תرتיב רשימה של המועמדים, מה שצפוי הוא שכל מופיע של P ב- T יופיע. לאחר מכן לכל אינדקס נבדוק האם יש מופיע.

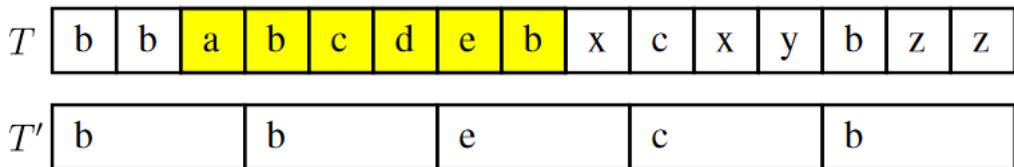
לכן יכול להיות יותרiesel מאלגוריתם נאייבי. רוצים `filter` ישאיר מעט אינדקסים לשלב השני.

Partition P into P^1, \dots, P^q , $P^i = P[i]P[i+q]P[i+2q]\dots$
 Find P^1, \dots, P^q in $T' = T[1]T[1+q]T[1+2q]\dots$



נחלק את המחרוזת Q למחרוזות, כלומר מחרוזות שמכילות תווים בדילוגים של 3 למשל.

קיבלנו 3 מחרוזות. בהינתן מחרוזת T ניציר מחרוזת חדשה ' T' ע"י כך שנייק כל תו שלישי של T .



אם יש מופיע של P ב T אז צריכה להיות מופיע של p_3, p_2, p_1 ב T' .

לכן בדוגמה נקבע מופיע של p_2 .

הכוון הפוך לא נכון!

ניתוח

למעשה מה שהוא עושים זה להניח שהקהלט מקרי ולען ניתן לחשב את ההסתברות לכך שיש לנו התאמה של i ב T' באינדקס j .

מחשבים את התוחלת של מספר ההתאמות שנתקבל.

- Suppose that the text is chosen randomly: each letter is chosen uniformly from Σ and the letters are chosen independently.
- Denote $\sigma = |\Sigma|$.
- Probability that P^i matches T' at some index j is $(1/\sigma)^{m/q}$.
- $X = \text{number of matches of } P^1, \dots, P^q \text{ in } T'$.
- $X_{i,j} = 1$ if P^i matches T' at index j .
- $X = \sum_{i=1}^q \sum_{j=1}^{n/q-m/q+1} X_{i,j}$ so

$$\begin{aligned} E[X] &= E\left[\sum_{i,j} X_{i,j}\right] = \sum_{i,j} E[X_{i,j}] = \sum_{i,j} \Pr[X_{i,j} = 1] \\ &= q \cdot \left(\frac{n}{q} - \frac{m}{q} + 1\right) \left(\frac{1}{\sigma}\right)^{m/q}. \end{aligned}$$

לכן נקבל שהתחלת תלוי בה וכן $\frac{q}{m}$ כולםן ככל ש q קטן אז הזמן ייגדל.

- $E[X] = q \cdot \left(\frac{n}{q} - \frac{m}{q} + 1 \right) (1/\sigma)^{m/q}$.
- Expected time of verification is $O(q \cdot \frac{n}{q} \cdot (1/\sigma)^{m/q} \cdot m)$.
- If we use a linear time matching algorithm for the filtering stage, the time complexity of filtering is $O(n/q)$.
- Choosing $q = m/2 \log_\sigma m$ gives filtering time $O(n \log_\sigma m/m)$ and expected verification time $O(n/m)$.
- Total time: $O(n \log_\sigma m/m)$. This time complexity is optimal.

אם נוכל לעשות את הסינון בזמן לינארי, אם לוקחים q אופטימלי מתקבל זמן בדיקה טוב.

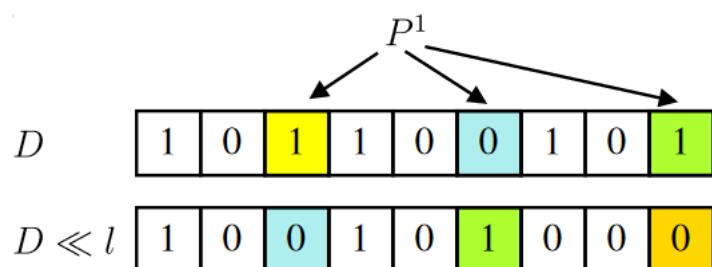
סיבוכיות זמן ($O(\log_\sigma m/m)$).

שלב הפילטור

אם אנו רוצים למצוא את המופיעים של l ב- D אנו נרצה במקביל לשמר אוטומט עבור l ולבצע סימולציה. ניקח ווקטור מצב ונאחד אותו.

כולםן D היא איחוד של h_m (P המקורי)
כאשר מחשבים ווקטור חדש עשוי הסטה של 3 צעדים – /.

- At each iteration, D is shifted by l bits.



רוצים לאפשר שינויים – החלפה של אות אחת באות שנייה

Definition

The **Hamming distance** between two strings S and R of equal length, denoted $\text{Ham}(S, R)$, is the number of indices i for which $S[i] \neq R[i]$.

Example

$S = \text{ABCABC}$

$R = \text{ACCABB}$

$\text{Ham}(S, R) = 2$

Input: A string T of length n , a string P of length m , an integer k .

Output: All the locations i such that $\text{Ham}(P, T[i..i + m - 1]) \leq k$.

Example

$$P = \text{ABCAAB}$$

$$T = \text{ABABCABAABCABAABCABA}$$

$$k = 4$$

Answer: 1,

Example

$$P \equiv \text{ABCAAB}$$

$$T = \text{ABABCABCABCABCABA}$$

$$k = 4$$

Answer: 1.

Example

$$P \equiv \text{ABCAAB}$$

$$T = \text{ABABCABAABCABAABCABAABCABA}$$

$$k = 4$$

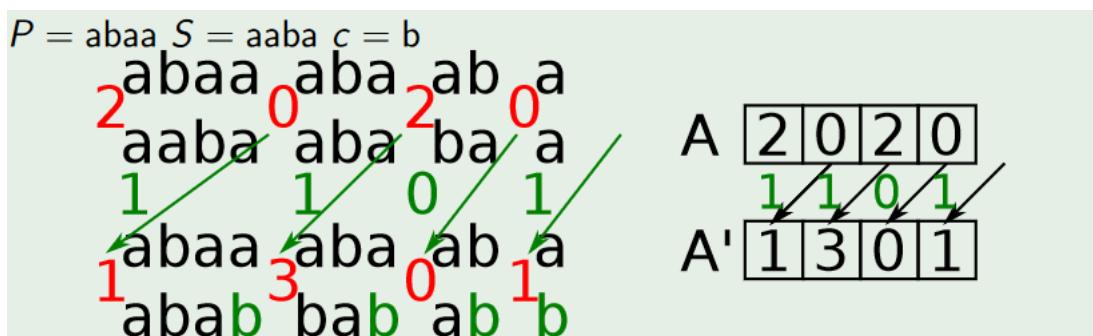
Answer: 1.3.

רעיון הוא להשתמש באלגוריתם הקודם

מה שעשינו בס-*shift* זה ליקח את *P* ולהסתכל ברישאות שלה, שמרנו ב*D* האם המחרזות שקראונו עד כה מופיעות בראות מושלמת

למשל עבור a

נרצה למצוא את כל ההתאמות של C ב- \mathcal{C} עד מספר שגיאות מסוים



חישבנו לכל רישא את מספר אי ההתאמות – בADIOm.

והוקטור שקיבלו הוא 2020.(כמויות אי התאמות)

נוסף ל S תו מימין ונחשב את הוקטור של אי ההתאמות בצורה ייעילה: $b|aab|a$ אם מסתכלים על סיפא באורך 3 2 התווים האחרונים חושבו ולכן אנו יודעים את אי ההתאמות שהו' ועכשו' משווים את זה עם רישא באורך 3 , אז משווים אותו עם התווים של P את ההשוואה מחלוקת 2 תווים ראשונים של P ו 2 תווים אחרונים של S והטו האחרון שנוסף לכך נוסיף 1 לערך שהוא בהשוואה הקודמת כי שי' אי ההתאמה.

$$S = aaba|b$$

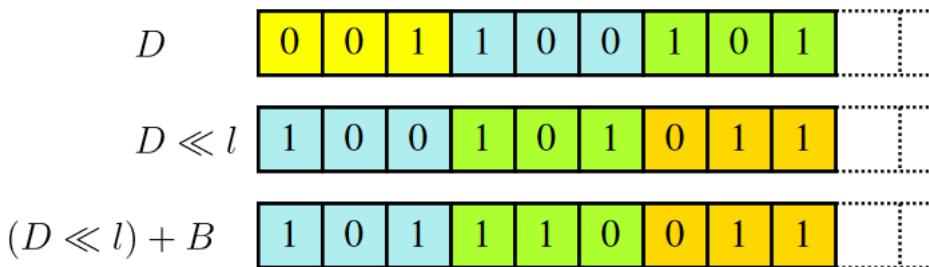
$$P = ab|a$$

קיבלנו שוב אלגוריתם שמחשב את המערך עבור S בתוסףתו מימין. אם נעשה יישום רגיל ייקח זמן לא יעל ולכן שוב נשמר את כל המערך A במלת זיכרון אחת.

בצורה הנאייבית צריך $m \log m$ ביטים לשמר עבור כל ערך שאנו רוצים לשמר ואז נשמר $m \log m$ ביטים.

אפשרות אחרת היא אם מחוץ החיפוש באורך 30 ו k באורך 3. (השגיאה) ניתן ליעיל את שמירת מספרי השגיאות.

- Store the array A in a bitvector D , using l bits per value.
- Updating D can be done using constant number of operations.



- The values of D are $0, \dots, m$ so we need $l = \lceil \log(m+1) \rceil$.
- If $k \ll m$, it is more efficient to use $l = \lceil \log_2(k+1) \rceil + 1$, but the algorithm is slightly different.

באלגוריתם פשוט שומרים m ביטים לכל תא, כדי לחשב את הוקטור החדש מסויתים את הוקטור ימינה 3 תאים ואז מוסיפים 1 לכל תא או 0.

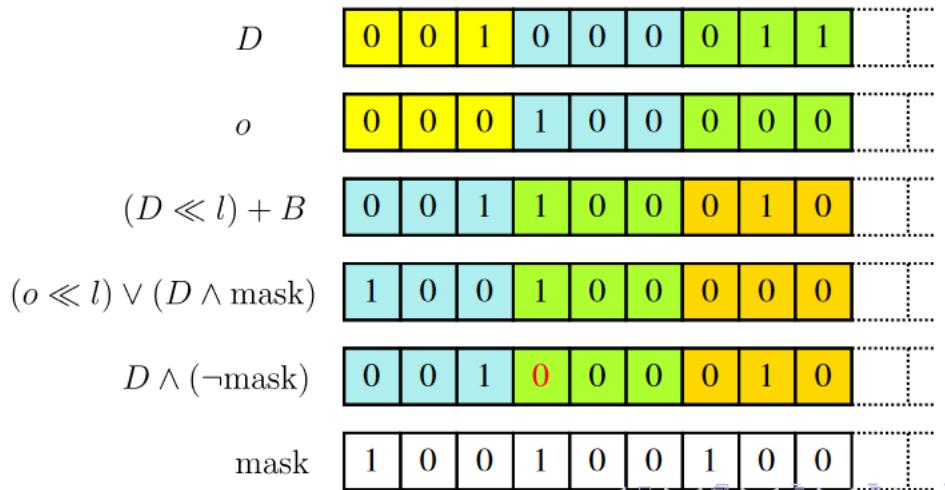
נניח ש W גדול ולכן מעוניין אותנו לשמר רק ערכים קטנים.

האלגוריתם עצמו:

נניח שמספריים 2 ביטים שמירת המספרים 0 לא.

אנו שומרים בדוגמה 3 ערכים וכל פעם מעדכנים את D , הבעה שככל איטרציה מעדכנים את מסויתים את הוקטור ומוציאים כמקודם 0/1.

- Use another word o . In a block of o , bit l is 1 iff the hamming distance is $\geq 2^{l-1} \geq k + 1$.
- Update D and o as follows
 - $D \leftarrow (D \ll l) + B[T[i]]$.
 - $o \leftarrow (o \ll l) \vee (D \wedge \text{mask})$
 - $D \leftarrow D \wedge (\neg \text{mask})$.



נשמר בית O נוסף האם ביצענו איפוס של הבית השמאלי ביותר.

בדוגמא הנ"ל בנוסף L שומרים את הווקטור האם ביצענו איפוס של הבית השמאלי ביותר בבלוק או לא.
בדגמא הزادת לא ולכן אנו יודעים שיש בדיק 3.

The Boyer-Moore-Horspool algorithm

P bbac

T  acbabbb

1st comparison window

P  bac

T  acbabbb

The last letter of the comparison window is 'b', and the last 'b' in $P[1..m - 1]$ is in position 2

P  bac

T  acbabbb

For the 2nd comparison window, we align the last 'b' in P with the 'b' at the end of the 1st window

P bbac

T  cbabbb

The last letter of the comparison window is 'c', and 'c' does not appear in $P[1..m - 1]$

P bbac
T bcb**bab****babb** 3rd comparison window

Definition

For $a \in \Sigma$, $\text{shift}[a] = \min \{s \geq 1 : s = m \text{ or } P[m-s] = a\}$.

- ➊ for every $a \in |\Sigma|$ do $\text{shift}[a] \leftarrow m$.
- ➋ for $i = 1, \dots, m-1$ do $\text{shift}[P[i]] \leftarrow m-i$.
- ➌ $i \leftarrow m$.
- ➍ while $i \leq n$ do
 - ➎ Compare $P[m-k]$ and $T[i-k]$ for $k = 0, \dots, m-1$ (stop comparing when mismatch is found).
 - ➏ $i \leftarrow i + \text{shift}[T[i]]$.

נחפש את bb במחוזת T , נשווהתו תוו. השאלה היא האם צריך להמשיך באינדקס 2 בדוגמה.

כמובן שלא, נסתכל על התו האחרון שבתורו החולון של אנו יודעים שהוא b ולכן נדרש להזיז את המחוזת עד שתשב שוב על רצף של bb . במקרה זה לא נדרש לבדוק את אינדקס 2 ולכן ניתן לקפוץ ישר ל.3.

כאשר מצאנו התאמה, עכשו לא תהיה התאמה באינדקס 4 כיון שהתו האחרון לא זהה לתו הראשון $c \neq b$.
לכן נוכל לזרז עד לאינדקס הכי ימני אחריו ולהשווות.

האלגוריתם עצמו פשוט, בניית מערך shift , כמה צעדים צריכים לקפוץ הלה כאשר בחולון ההתאמה האחרון נראתה התו a , בדוגמה שהייתה לנו $\text{shift}(b)=2$. האלגוריתם מחשב כל פעם מערך shift ואז מבצע הגדלה של / לפני המערך. האלגוריתמים פשוט יותר מ-KMP כיון שיש יותר עילום.

• Random text ($\sigma = 128$), $n \approx 5 \cdot 10^6$

m	2	4	8	16	32
Shift-or	16.8	16.9	16.8	16.8	16.8
Shift-or, $q = 2$	12.7	9.7	9.8	9.7	9.7
Shift-or, $q = 4$	-	8.0	5.1	5.1	5.1
BMH	24.2	12.6	6.7	3.8	2.5

• Natural language, $n \approx 2.5 \cdot 10^6$

m	2	4	8	16	32
Shift-or	8.1	8.1	8.1	8.1	8.1
Shift-or, $q = 2$	14.3	5.6	4.8	4.8	4.8
Shift-or, $q = 4$	-	11.9	3.1	2.5	2.5
BMH	13.7	7.4	4.3	2.6	1.8

אלגוריתמים למחוזות / סיכום מאת אורי שביט

השוואת זמני הריצות של האלגוריתמים שראינו היום, זמני ריצה של האלגוריתמים כפונקציה של אורך המחרוזת שמחפשים M . אין מנצח עדיף תמיד.

מצגת FFT 17

$B(x) = \sum_{k=0}^{n-1} b_k x^k$ | $A(x) = \sum_{k=0}^{n-1} a_k x^k$ - יהי $- Multiplication of polynomials$

אנו רוצים לחשב את $C(x) = A(x) * B(x) = \sum_{k=0}^{n+2n-2} c_k x^k$

דוגמא:

$$\begin{aligned} (2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) &= \\ 2x^5 + 3x^4 + &x^3 + 2x^2 + \\ 4x^4 + &6x^3 + 2x^2 + 4x + \\ 4x^3 + &6x^2 + 2x + 4 = \\ 2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4 \end{aligned}$$

$$A = (2, 3, 1, 2), B = (0, 1, 2, 2), C = (0, 2, 7, 11, 10, 6, 4).$$

האלגוריתם הנאייבי לוקח $O(n^2)$.

יהי $(x) A$ פולינום ממעלה קטנה מה.

הגדרה: $A(x) = \sum_{k=0}^{n-1} a_k x^k$ של A הוא coefficient representation

הגדרה: $\{(x_0, A(x_0)), \dots, (x_{n-1}, A(x_{n-1}))\}$ כאשר x_k ייחודי מ- A היא point-value representation והם ייחודיים/מובחנים.

דוגמא:

Let $A(x) = x^2 + 2x + 4$. Some point-value representations of A are

$$\{(0, 4), (1, 7), (2, 12)\}$$

and

$$\{(0, 4), (1, 7), (3, 19)\}$$

Properties of point-value representation

טענה: בהינתן קבוצה של נקודות $S = \{(x_0, y_0), \dots, (x_{n-1}, y_{n-1})\}$ כאשר x_k ייחודי, יש לבדוק פולינום ייחיד $A(x)$ מדרגה לכל היותר n כך ש $y_k = A(x_k)$ עבור $k = 0, \dots, n-1$.

הוכחה: אנו צריכים למצאו $A(x) = \sum_{k=0}^{n-1} a_k x^k$ כך שמתקיים:

$$A(x_0) = y_0$$

$$a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_{n-1} x_0^{n-1} = y_0$$

$$A(x_1) = y_1$$

$$a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_{n-1} x_1^{n-1} = y_1$$

⋮

$$A(x_{n-1}) = y_{n-1}$$

$$a_0 + a_1 x_{n-1} + a_2 x_{n-1}^2 + \dots + a_{n-1} x_{n-1}^{n-1} = y_{n-1} \quad 172$$

הפולינום המבוקש הוא $a_n x^n + \dots + a_1 x + a_0 = P(x) = y_i$ אמור לקיים: $P(x) = a_0 + a_1 x + \dots + a_n x^n$ ברישום מטריציוני:

$$\vec{a}X = \vec{f}$$

$$\vec{a} = (a_0, a_1, \dots, a_n)$$

$$X = (x_j^i)_{i,j=0}^n$$

$$\vec{f} = (f_0, f_1, \dots, f_n)$$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

זהו מטריצה ונדרמןד כאשר כל שורה היא סדרה הנדסית מטריצה זו מעריצה פולינום בנקודות: היא מעבירה את המקדמים של הפולינום לערכים שהפולינום מקבל בנקודות x_i .

ראויים כי המקדמים של הפולינום מקיימים מערכת משוואות בה מטריצת המקדמים היא ונדרמןד המבוססת על מספרים שונים בפרט זו מטריצה הפיכה ולכן הפתרון ייחידה.

הדרטמיננטה של מטריצת ונדרמןד היא $(x_k - x_j)_{0 \leq j < k \leq n-1}$,

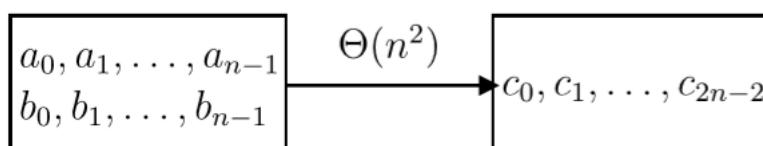
מהר ש- x מובנים המריצה הפיכה וקיים פתרון ייחודי. מש"ל

- Converting the coefficient representation to point-value representation takes $\Theta(n^2)$ time (**evaluation**).
- Converting a point-value representation to coefficient representation takes $\Theta(n^2)$ time (**interpolation**).

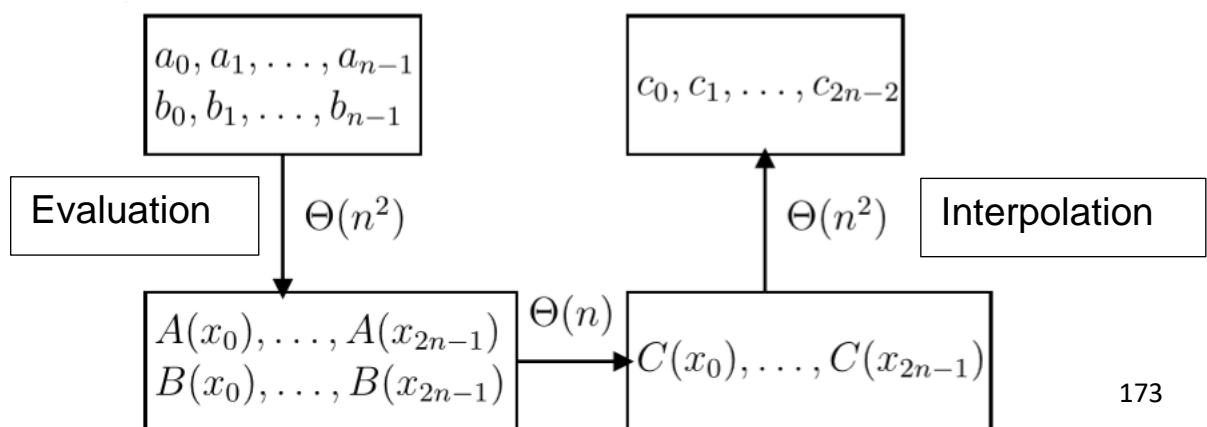
עובדת: יהו $\{A(x), B(x)\}$ פולינומים מדרגה לכל היותר n . יהו $C(x) = A(x) * B(x)$ יכול להיות מחושב בזמן $O(n)$. נבחן כי צריך לייצג עם מספר נקודות יותר גבוהה בגלל מעלה הפולינום C .

הוכחה: יהו $C(x) = A(x) * B(x)$ הם הייצוג בנקודות של $C(x)$. מש"ל.

יהו הפולינומים $A(x), B(x)$ מדרגה לכל היותר n , ב *coefficient representation*. אנו רוצים לחשב את $C(x) = A(x) * B(x)$.

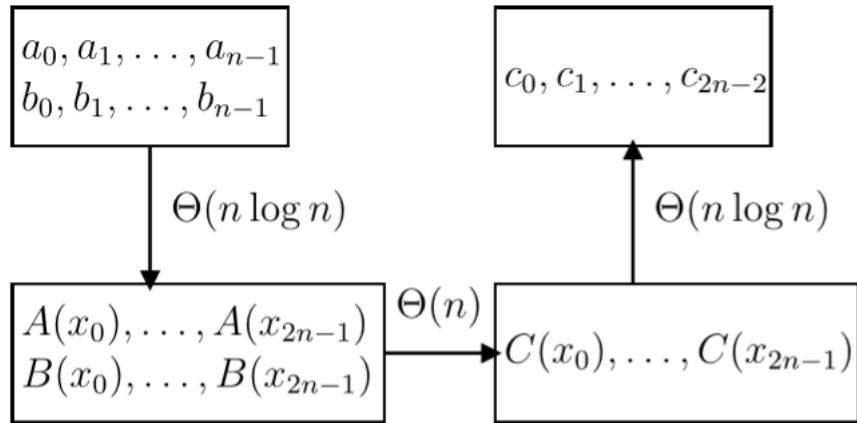


אנו רוצים להשתמש בעובדה שהכפלת פולינומים ב*point-value representation* הוא מהיר. אנו מעריכים את A ו- B בקבוצה מסוימת של נקודות x_0, \dots, x_{2n-1} .



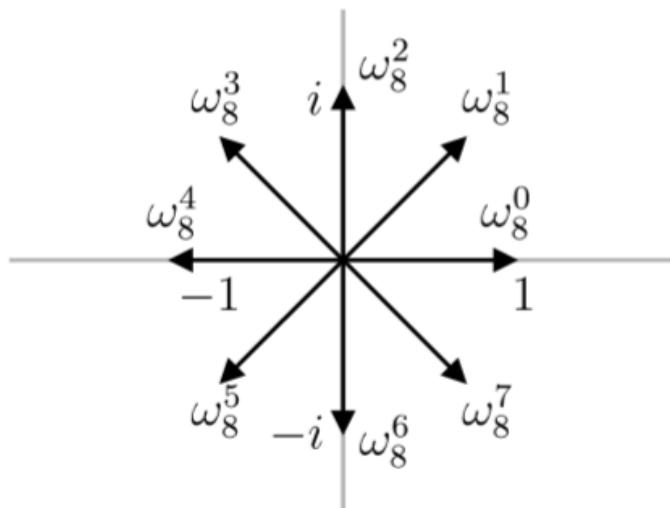
אלגוריתמים למחושבות / סיכום מאות אורי שביט

באופן כללי הערכה תיקח $\Theta(n^2)$. ע"י בחירה חכמה של הנקודות x אנו יכולים לבצע הערכה אינטראפולציה ב- $O(n \log n)$ זמן.



$$\omega_n = e^{\frac{2\pi i}{n}}$$

הגדרה: $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ הם הפתרונות למשוואה $z^n = 1$. בשדה המרוכבים לבועה יש לבדוק n פתרונות.



$$\omega_{dn}^{dk} = \omega_n^k$$

$$\omega_n^{dk} = \left(e^{\frac{2\pi i}{dn}} \right)^{dk} = \left(e^{\frac{2\pi i k}{n}} \right)^k = \left(e^{\frac{2\pi i}{n}} \right)^k = \omega_n^k$$

Discrete Fourier Transform

הכפלת הפולינומים A ו B תשוערך ע"י $\omega_{2n}^0, \omega_{2n}^1, \dots, \omega_{2n}^{2n-1}$. לכן אנו זקוקים לאלגוריתם מהיר לחישוב

$$A(x) = \sum_{k=0}^{n-1} a_k x^k \text{ on } \omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$$

Discrete Fourier Transform

הגדרה: יהי הווקטור $(y_0, \dots, y_{n-1}) = y$ כאשר $y_k = \sum_{j=0}^{n-1} a_j \omega_n^{jk}$. יהי הווקטור $(a_0, \dots, a_{n-1}) = a$. נקרא a נקראה *Discrete Fourier Transform*.

אלגוריתמים למחושבות / סיכום מאות אורי שביט

אנו רוצים להעריך את $A(x) = \sum_{k=0}^{n-1} a_k x^k$, נניח שהו חזקה של 2. (אחרת ניקח את החזקה הקדומה ביותר). האומגה יכנסו ככלט לפולינום ולכן לא מספיק לנו כי יתנו לנו סדרת נקודות.

נגדיר 2 פולינומים מדרגה קטנה מ- $\frac{n}{2}$:

$$A_0(x) = a_0 + a_2 x + a_4 x^2 + \dots + a_{n-2} x^{\frac{n}{2}-1}$$

$$A_1(x) = a_1 + a_3 x + a_5 x^2 + \dots + a_{n-1} x^{\frac{n}{2}-1}$$

בבירור ניתן לראות כי ניתן לקבל את הפולינום המקורי:

$$A_0(x^2) + x A_1(x^2) =$$

$$= a_0 + a_2 x^2 + a_4 x^{2*2} + \dots + a_{n-2} x^{n-2} + x(a_1 + a_3 x^2 + a_5 x^{2*2} + \dots + a_{n-1} x^{n-2}) = A(x)$$

אנו יכולים להעריך את A עבור $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ בצורה הבאה:

1. הערכת A_0 עבור $\omega_{n/2}^0, \omega_{n/2}^1, \dots, \omega_{n/2}^{n/2-1}$, מעריכים על סדרת הנקודות (חישוב ברקורסיה עליהם עד להגעה לפולינום מדרגה קבועה).

2. שילוב התוצאות עבור כל k :

$$A(\omega_n^k) = A_0(\omega_n^{2k}) + \omega_n^k A_1(\omega_n^{2k}) = A_0\left(\omega_{\frac{n}{2}}^k\right) + \omega_n^k A_1\left(\omega_{\frac{n}{2}}^k\right)$$

ניתן לחלק ב 2 לפי בחירת n חזקה של 2.

נוסחת הנסיגה לחישוב זמן הריצה היא $n + 2T\left(\frac{n}{2}\right)$ ולכן נקבל

סיבוכיות הזמן היא $\Theta(n \log n)$.

Properties of complex roots of unity

עובדיה: אם $0 < j \leq n-1$ לא מתחלק ב n אז $\omega_n^j = 1$ והוכחה:

$$\sum_{k=0}^{n-1} \omega_n^{jk} = \sum_{k=0}^{n-1} (\omega_n^j)^k = \frac{(\omega_n^j)^n - 1}{\omega_n^j - 1} = \frac{(\omega_n^n)^j - 1}{\omega_n^j - 1} = 0.$$

זו סדרה הנדסית סופית ולכן ניתן להשתמש בנוסחה לחישוב הסכום, $1 + \omega_n + \dots + \omega_n^{n-1} = 0$ ולכן נקבל 0.

אנו צריכים לחשב:

$$\begin{pmatrix} 1 & \omega_n^0 & \omega_n^0 & \dots & \omega_n^0 \\ 1 & \omega_n^1 & \omega_n^2 & \dots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \dots & \omega_n^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

כאשר המטריצה V היא ונדראונדה. $V_{j,k} = \omega_n^{kj}$, קלומר הנקודות x הם עבור אומגה.

היא W המטריצה המוגדרת $W = V^{-1}$.

עובדה: $W = V^{-1}$

הוכחה: אנו רוצים להראות ש $WV = I_n$

$$[WV]_{j,j'} = \sum_{k=0}^{n-1} \frac{\omega_n^{-kj}}{n} \cdot \omega_n^{kj'} = \sum_{k=0}^{n-1} \omega_n^{k(j'-j)} / n = \begin{cases} 1 & j = j' \\ 0 & j \neq j' \end{cases}$$

האיברים המתאים במטריצות ولكن ניתן לפי חוקי חזקות לעשות את ההכפלת.

ולכן מהטענה לעיל נקבל כי שווה ל-0 או 1 אם $j=j'$.

כלומר 0 אם לא באלכסון ו-1 אם באלכסון. לכן יש לנו נוסחה פשוטה למטריצה ההופכית.

יש לנו נוסחה פשוטה לחישוב

- DFA: $y_k = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$.
- Inverse DFA: $a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj}$.

To compute the DFA of y_0, \dots, y_{n-1} , modify the FFT algorithm to switch the roles of a and y , replace ω_n by ω_n^{-1} , and divide each element of the result by n .

האלגוריתם דומה לייצוג האלגוריתם לייצוג בנקודות, אלגוריתם הקודם בשינוי קל.

סיכום

Theorem

Two polynomials of degree $< n$ can be multiplied in $\Theta(n \log n)$ time, with both the input and output representations in coefficient form.

קיבלנו שאם אנו מקבלים 2 פולינומים אנו יכולים להכפילם בזמן $n \log n$

נדיר את הפעולה על 2 וקטורים – קונבולוציה של A ו- B :
הוא בדיקת הכפלת הפולינומים.

הפולינום A יהיה בעל סדרת המקדמים 2 3 1 2

הפולינום B יהיה המקסימים בצורה הפוכה של המקדמים של B .

כשאנו מכפילים את הפולינומים אך אנו שומרים את התוצאות של ההכפלות של התאים המתאים בנפרד לפני המקדים.

$$(2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) =$$

$$2x^5 + 3x^4 + x^3 + 2x^2 +$$

$$4x^4 + 6x^3 + 2x^2 + 4x +$$

$$4x^3 + 6x^2 + 2x + 4 =$$

$$2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4$$

<i>A</i>	<table border="1"><tr><td>2</td><td>3</td><td>1</td><td>2</td></tr></table>	2	3	1	2
2	3	1	2		
<i>B</i>	<table border="1"><tr><td>2</td><td>2</td><td>1</td></tr></table>	2	2	1	
2	2	1			

$$(2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) =$$

$$2x^5 + 3x^4 + x^3 + 2x^2 +$$

$$4x^4 + 6x^3 + 2x^2 + 4x +$$

$$\textcolor{red}{4x^3} + 6x^2 + 2x + 4 =$$

$$2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4$$

$$(2x^3 + \textcolor{red}{3x^2} + x + 2) \cdot (x^2 + \textcolor{red}{2x} + 2) =$$

$$2x^5 + 3x^4 + x^3 + 2x^2 +$$

$$4x^4 + \textcolor{red}{6x^3} + 2x^2 + 4x +$$

$$4x^3 + 6x^2 + 2x + 4 =$$

$$2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4$$

$$(2x^3 + 3x^2 + \textcolor{red}{x} + 2) \cdot (\textcolor{red}{x^2} + 2x + 2) =$$

$$2x^5 + 3x^4 + \textcolor{red}{x^3} + 2x^2 +$$

$$4x^4 + 6x^3 + 2x^2 + 4x +$$

$$4x^3 + 6x^2 + 2x + 4 =$$

$$2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4$$

$$\begin{aligned}
 (2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) = \\
 2x^5 + 3x^4 + x^3 + 2x^2 + \\
 4x^4 + 6x^3 + 2x^2 + 4x + \\
 4x^3 + 6x^2 + 2x + 4 = \\
 2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4
 \end{aligned}$$

$$\begin{aligned}
 (2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) = \\
 2x^5 + 3x^4 + x^3 + 2x^2 + \\
 4x^4 + 6x^3 + 2x^2 + 4x + \\
 4x^3 + 6x^2 + 2x + 4 = \\
 2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4
 \end{aligned}$$

$$\begin{aligned}
 (2x^3 + 3x^2 + x + 2) \cdot (x^2 + 2x + 2) = \\
 2x^5 + 3x^4 + x^3 + 2x^2 + \\
 4x^4 + 6x^3 + 2x^2 + 4x + \\
 4x^3 + 6x^2 + 2x + 4 = \\
 2x^5 + 7x^4 + 11x^3 + 10x^2 + 6x + 4
 \end{aligned}$$

29

Let $A = (a_1, \dots, a_n)$ and $B = (b_1, \dots, b_m)$ be two vectors of integers (with $m \leq n$).

The convolution of A and B (denoted $A \circ B$) is a vector $C = (c_1, \dots, c_{n-m+1})$, where

$$c_j = \sum_{i=1}^m a_{j+i-1} b_i.$$

המכפלות כל פעם של המספרים שאחד מעל השני:

<i>A</i>	2	1	3	1	0	0	1	2	3
----------	---	---	---	---	---	---	---	---	---

<i>B</i>	1	0	2
----------	---	---	---

$$C = (8,$$

<i>A</i>	2	1	3	1	0	0	1	2	3
----------	---	---	---	---	---	---	---	---	---

<i>B</i>	1	0	2
----------	---	---	---

$$C = (8, 3,$$

<i>A</i>	2	1	3	1	0	0	1	2	3
----------	---	---	---	---	---	---	---	---	---

<i>B</i>	1	0	2
----------	---	---	---

$$C = (8, 3, 3, \dots)$$

הkonvolוציה היא אותה פעולה כמו מכפלת פולינומיים.

כשהגדרנו את המכפלה של הפולינומים יכלנו להניח שהדרגה לכל היותר ℓ , ראיינו שהמכפלה שלהם .

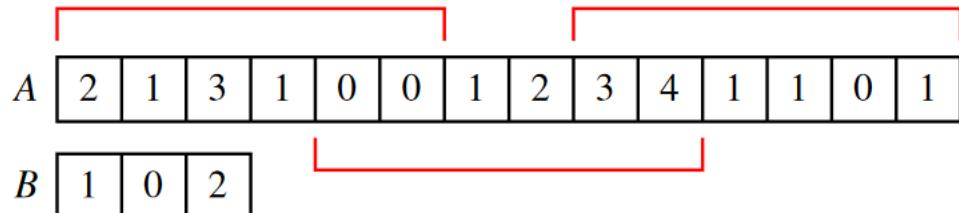
Using the FFT, the convolution of vectors of lengths n and $n/2$ can be computed in $O(n \log n)$ time.

Corollary

The convolution of vectors of lengths n and m can be computed in $O(n \log m)$ time.

Proof.

The vector of length n can be split into segments of length $2m$, with overlap of $m - 1$ between adjacent segments. □



נחלק את A לחלקי בגודל $2m$ כאשר בין 2 חלקי עוקבים יש חפיפה בגודל $m - 1$.

נחשב את הקונבולוציה של החלקים של A עם B . אם נחבר אותם נקבל את הקונבולוציה של A עם B . כל קונבולוציה לוקחת $m \log m$ פעמיים ולכן נקבל $O(m \log n)$.

Pattern Matching with Don't Cares

Input:

- A string T over an alphabet Σ .
- A string P over $\Sigma \cup \{\phi\}$.

Output: All the substrings of T (of length $|P|$) that match to P , where the character ϕ matches to any character from Σ .

Example

$$\begin{aligned} T &= \text{abcdadca} \\ P &= \text{a}\phi\text{c}a \end{aligned}$$

רוצים למצוא את כל המופיעים כאשר התו הלא משנה יכול להתאים לכלתו.

Example

$$\begin{aligned} T &= \text{abca}\text{d}\text{adca} \\ P &= \text{a}\phi\text{c}a \end{aligned}$$

רוצים לפתור בעיה זו:

Naive algorithm: $O(nm)$, where $n = |T|$ and $m = |P|$.

Can we do better?

האלגוריתם של KMP לא עובד כאן, כיס ההתאמה אינה טרנזיטיבי.

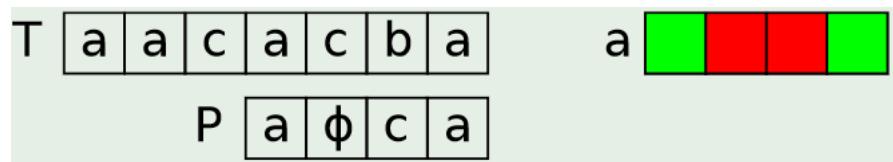
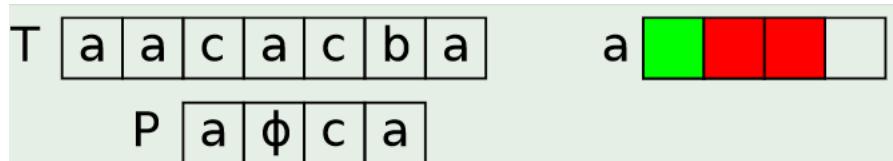
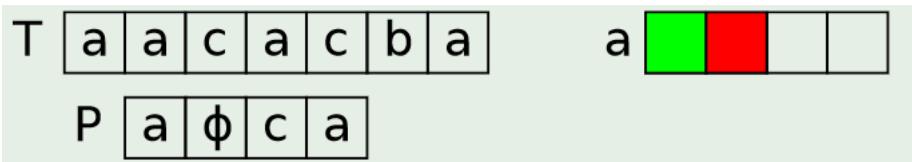
האלגוריתם

נחפש התאמות בין P ו- T כאשר כל פעם נבדוק לפניתו ספציפי בא"ב.

- For each $c \in \Sigma$, find the i -s such that $T[i..i+m-1]$ is consistent with P on c .
- Report the positions consistent with all c .

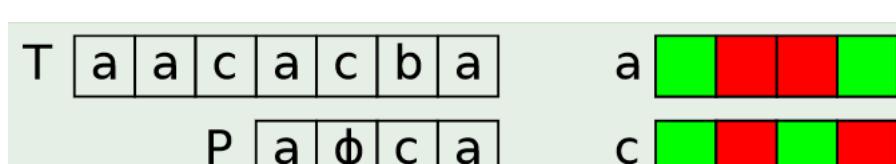
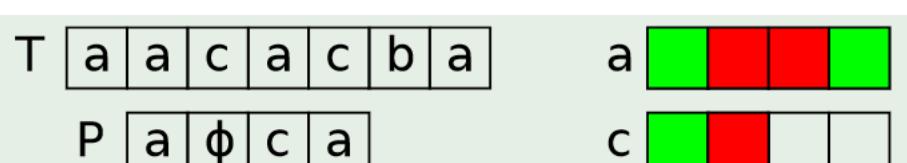
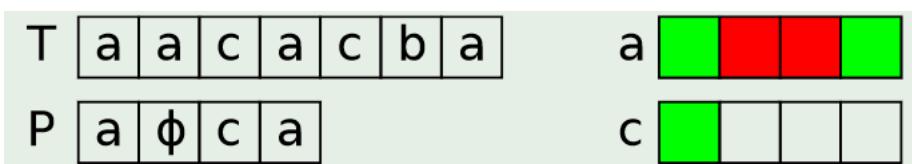
Example

T	a	a	c	a	c	b	a	a				
P	a	φ	c	a								

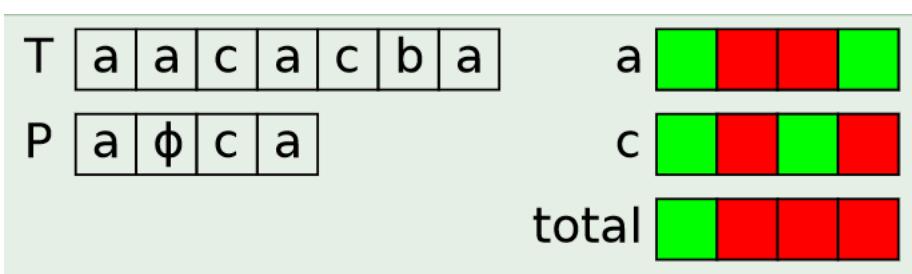


מתעלמים מכל התאים פרט ל a

עושים אותו דבר לו c



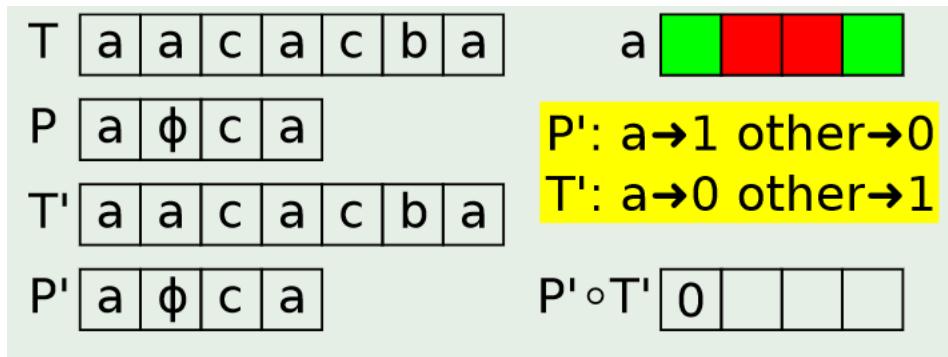
עושים and על כל הוקטורים וזה נקבל את התוצאה.



רוצים לחשב את המערך הראשון :

$aXXX$

נייצר 2 מחuzzות חדשות :



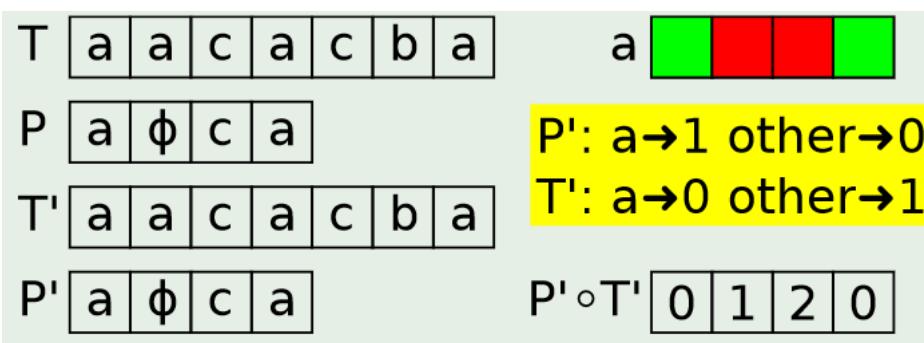
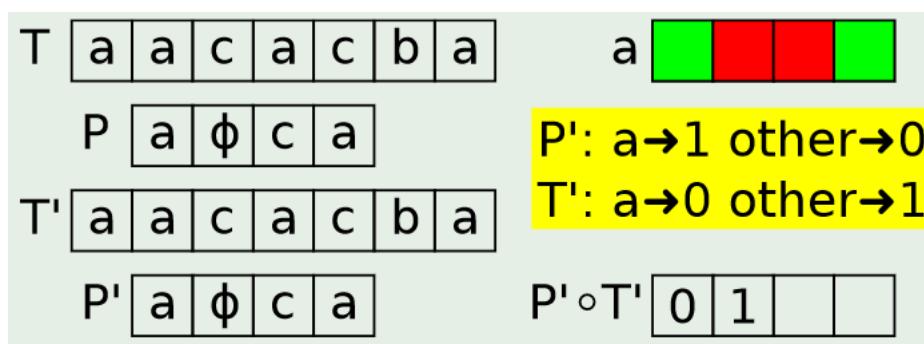
T כל a הופך ל0 ואחרת ל1

P' כל a הופך ל1 וכלתו אחר הופך ל0.

$$T' = 0010110$$

$$P' = 1001$$

ואז עושים קונבולוציה בניתם.



נקבל את הקונבולוציה בניתם

האלגוריתם

- $P_a = P$ where a is replaced by 1 and other chars by 0.
- $T_{\bar{a}} = T$ where a is replaced by 0 and other chars by 1.
- Compute $T_{\bar{a}} \circ P_a$.
- Report every index i such that $T_{\bar{a}} \circ P_a[i] = 0$ for all $a \in \Sigma$.

Example $T = \text{aacacba}$ $P = \text{a}\phi\text{ca}$ $T_{\bar{a}} = (0, 0, 1, 0, 1, 1, 0), P_a = (1, 0, 0, 1), T_{\bar{a}} \circ P_a = (\textcolor{red}{0}, 1, 2, 0)$ $T_{\bar{c}} = (1, 1, 0, 1, 0, 1, 1), P_c = (0, 0, 1, 0), T_{\bar{c}} \circ P_c = (\textcolor{red}{0}, 1, 0, 1)$

צריך ללקח את כל התאים שבהם יש 0.

- For each $a \in \Sigma$, the convolution takes $O(n \log m)$ time.
- Overall time complexity: $O(|\Sigma|n \log m)$.

עבור א"ב לא קבוע ניתן גם לעשות זאת, נראה אלגוריתם נוסף:

- $T = t_1 \dots t_n$ is a string over $\Sigma = \{1, \dots, n\}$.
- $P = p_1 \dots p_m$ is a string over $\{0, 1, \dots, n\}$ ($\phi = 0$).
- The substring $t_j \dots t_{j+m-1}$ matches P iff

$$\sum_{i=1}^m p_i (p_i - t_{j+i-1})^2 = 0.$$

Example

T	1	1	3	1	3	2	1	0			
-----	---	---	---	---	---	---	---	---	--	--	--

P	1	0	3	1
-----	---	---	---	---

$$(p_i - t_{j+i-1})^2 \quad 0 \quad 1 \quad 0 \quad 0$$
יכולים להיות ב- P תווים *don't care*.

T	<table border="1"><tr><td>1</td><td>1</td><td>3</td><td>1</td><td>3</td><td>2</td><td>1</td></tr></table>	1	1	3	1	3	2	1	<table border="1"><tr><td>0</td><td>16</td><td></td><td></td></tr></table>	0	16		
1	1	3	1	3	2	1							
0	16												
P	<table border="1"><tr><td>1</td><td>0</td><td>3</td><td>1</td></tr></table>	1	0	3	1								
1	0	3	1										
$(p_i - t_{j+i-1})^2$	0 9 4 4												

T	<table border="1"><tr><td>1</td><td>1</td><td>3</td><td>1</td><td>3</td><td>2</td><td>1</td></tr></table>	1	1	3	1	3	2	1	<table border="1"><tr><td>0</td><td>16</td><td>5</td><td></td></tr></table>	0	16	5	
1	1	3	1	3	2	1							
0	16	5											
P	<table border="1"><tr><td>1</td><td>0</td><td>3</td><td>1</td></tr></table>	1	0	3	1								
1	0	3	1										
$(p_i - t_{j+i-1})^2$	4 1 0 1												

T	<table border="1"><tr><td>1</td><td>1</td><td>3</td><td>1</td><td>3</td><td>2</td><td>1</td></tr></table>	1	1	3	1	3	2	1	<table border="1"><tr><td>0</td><td>16</td><td>5</td><td>3</td></tr></table>	0	16	5	3
1	1	3	1	3	2	1							
0	16	5	3										
P	<table border="1"><tr><td>1</td><td>0</td><td>3</td><td>1</td></tr></table>	1	0	3	1								
1	0	3	1										
$(p_i - t_{j+i-1})^2$	0 9 1 0												

עושים משהו שדומה לקונבולוציה, מזיזים את P לאורך T , כל זוג תאים ניקח את ההפרש ביניהם ונעלם אותו. בריבוע.

כעת אנו לוקחים כל ערך קיבלנו שם יש התאמה קיבלנו 0 – ככלומר אם נחסר אותם קיבל 0 – כל התווים המתאימים נקבל 0 וכן תא' *don't care* גם כן יהיה 0 בגלל ה嚗פהה.

אם אין התאמה התוצאה תהיה גדולה מ-0 – כמשמעותי התאמה אחרת יהיה שונה מ-0 בתווים מסוימים – שאר המכפלות : איבר זהה – הריבוע יהיה 0 (לא ישפייע על הסכום), איבר אחר של אי התאמה אך ריבוע ההפרש יהיה גדול מ-0 וכן קיבל מספר שגדל מ-0 ומגדילים את הסכום, *don't care* מקבל 0 ולא ישפייע על הסכום.

чисוב הקונבולוציה הנ"ל :

התוצאה שאנו רוצים לקבל – ניתן לפתח את הריבוע ולפרקו ל-3 סכומים שונים
чисוב הסכום הראשון הוא קבוע – חישוב פעם אחת כי לא תלוי ב-

чисוב הסכום השני – חישוב של P^2 פעם אחת וכן לעשות קונבולוציה $T \circ P^2$

чисוב הסכום השלישי – $T^2 \circ P$

סה"כ זמן הריצה = $\log m$

- The substring $t_j \cdots t_{j+m-1}$ matches P iff

$$\sum_{i=1}^m p_i(p_i - t_{j+i-1})^2 = 0.$$

- But

$$\sum_{i=1}^m p_i(p_i - t_{j+i-1})^2 = \sum_{i=1}^m p_i^3 - 2 \sum_{i=1}^m p_i^2 t_{j+i-1} + \sum_{i=1}^m p_i t_{j+i-1}^2.$$

- 1st sum is constant.
- 2nd sum is equal to $(T \circ P^2)[j]$, where $P^2 = p_1^2 \cdots p_m^2$.
- 3rd sum is equal to $(T^2 \circ P)[j]$.

Example $P = \text{ABCAAB}$ $T = \text{ABABCAABCABCABCABAA}$

Answer: 3, 7, 11

הרצאה 18 *Approximate String Matching*מקבילים מחוזות ומוצאים מופיעים של אחת בשניה. – *Exact string matching***Input:** A string T of length n , a string P of length $m \leq n$.**Output:** All locations i such that P appears in T starting at i .את בעיה זו ניתן לפתור בזמן לינארי $O(n)$ – אלגוריתם *Knuth-Morris-Pratt*cutet אנו רוצים למצוא את כל התת-מחוזות של T שהן "קרובות" ל P .הגדרה: Hamming distance בין 2 מחוזות S ו- R שווה לאורך המsoonן ע"י ($\text{Ham}(S, R)$) $. S[i] \neq R[i]$ עבורם i האינדקסים**Example** $S = \text{ABCABC}$ $R = \text{ACCCABB}$ $\text{Ham}(S, R) = 2$

הגדרת הבעיה:

קלט: מחוזת T באורך n ומחרוזת P באורך m פלט: עבור כל אינדקס i , את תוצאה($\text{Ham}(P, T[i..i + m - 1])$) $i = 0$ $i = 1$ $P = \text{ABCABA}$ $T = \text{ABABCAABCABCABCABAA}$

Answer: 4

 $P = \text{ABCAAB}$ $T = \text{ABABCAABCABCABCABAA}$

Answer: 4, 5

 $i = 2$ $P = \text{ABCAAB}$ $T = \text{ABABCAABCABCABCABAA}$

Answer: 4, 5, 0

*String matching with k mismatches*קלט: מחוזת T באורך n ומחרוזת P באורך m וקבוע k שלם כלשהו.פלט: עבור כל אינדקס i , את תוצאה($\text{Ham}(P, T[i..i + m - 1]) \leq k$)**Example** $P = \text{ABCAAB}$ $T = \text{ABABCAABCABCABCABAA}$ $k = 4$

Answer: 1, 3, ...

Simple algorithm

1. יהי $Votes$ מערך בגודל $1 \leq m - n$ המכיל אפסים

2. עבור $1 \leq l = 1, \dots, n - m + 1$ בצע:

3. עבור $m \geq j = 1, \dots, m$ בצע:

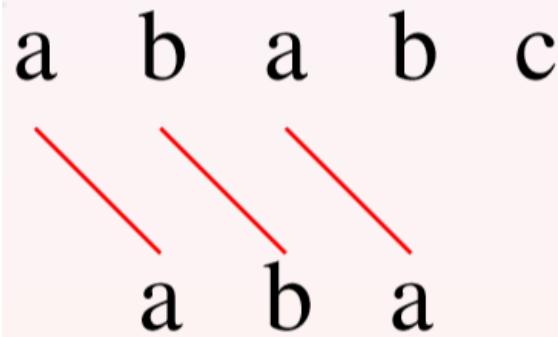
4. $Votes[l] + 1 \rightarrow Votes[l]$ ואם $P[j] = T[l + j - 1]$

בסוף האלגוריתם $Votes[l]$ מכיל את מספר אי ההתאמות בין P ו- $T[l \dots l + j - 1]$.

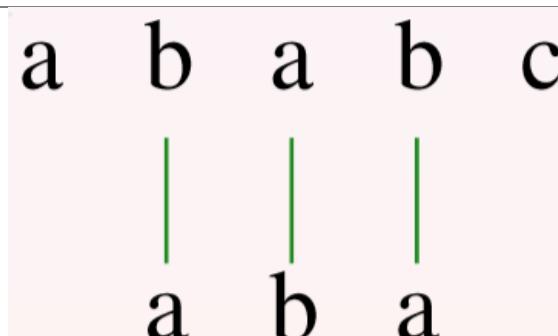
סיבוכיות זמן $O(mn)$.

0	0	0
---	---	---

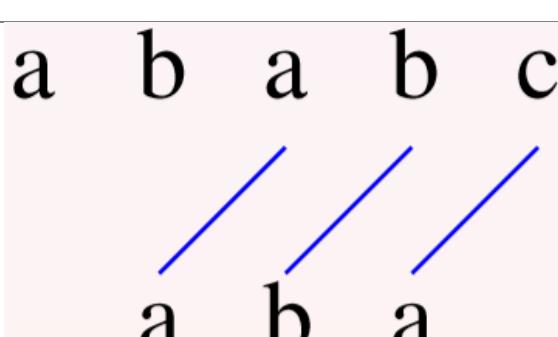
דוגמא עבור המחרוזות: $T = ababc$, $P = aba$. אתחול מערך $Votes$:



1	0	0
2	0	0
3	0	0
3	0	0
3	0	0

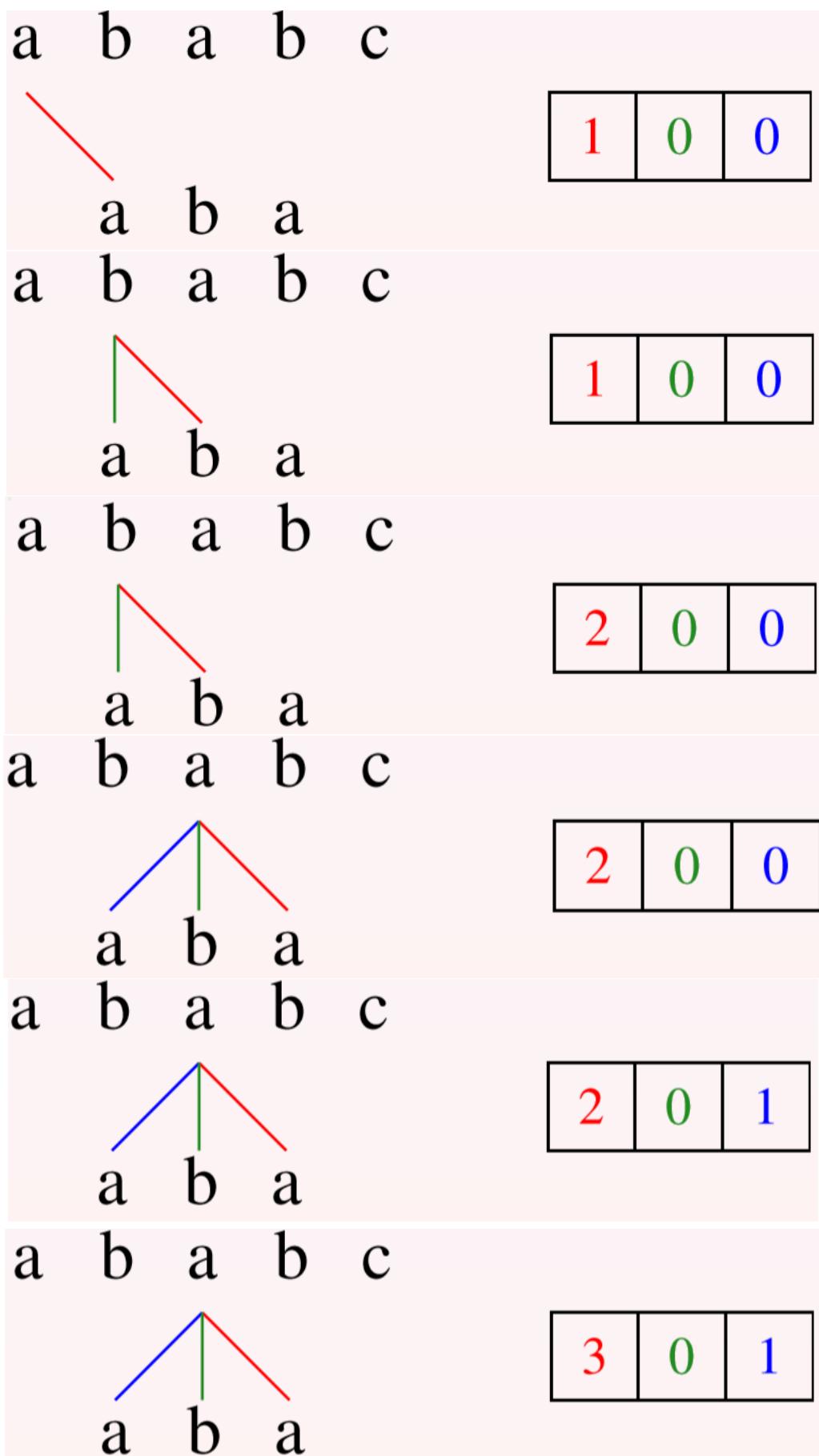


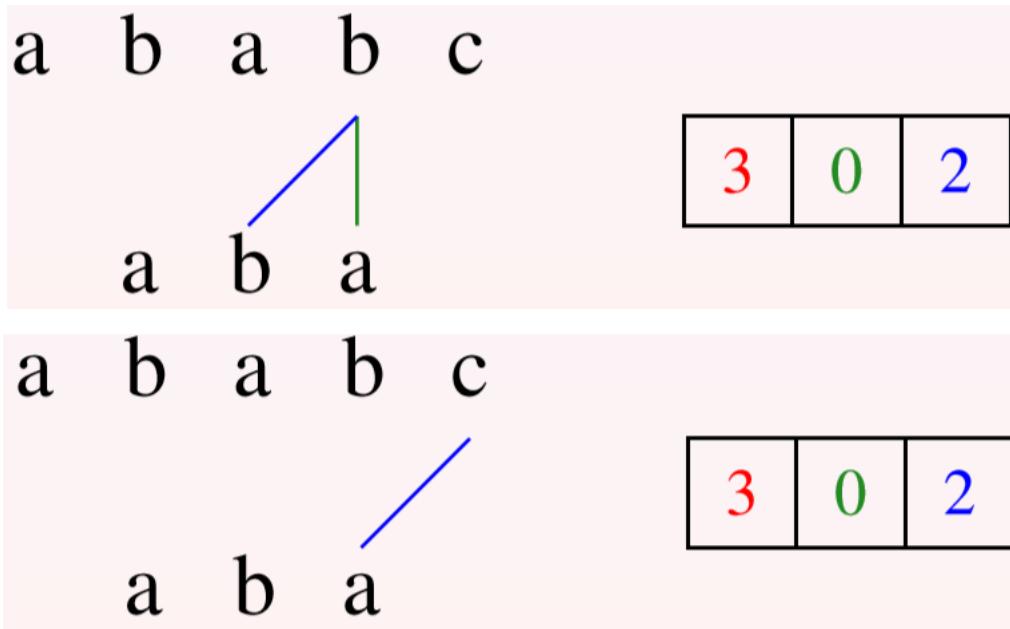
3	0	0
3	0	0
3	0	0
3	0	0



3	0	0
3	0	1
3	0	2

אנו יכולים לשנות את האלגוריתם ולקצת ביחס את כל ההשואות עבור $[i]T$.





האלגוריתם המחוදש

1. יהי $Votes$ מערך בגודל $1 + m - n$ המכיל אפסים
 2. עבור $n, 1, \dots, i = j$ בצע:
 3. עבור $m, \dots, j = 1, \dots, i$ בצע:
 4. אם $[i] = T[j]$ אז $P[j] = T[i] + 1 \rightarrow Votes[i - j + 1] + 1 \rightarrow Votes[i - j + 1]$
- נבחין כי אנו לא צריכים לעבור שוב עבור j .

Voting algorithm

1. יהי $Votes$ מערך בגודל $1 + m - n$ המכיל אפסים
 2. עבור כל $c \in \Sigma$ חשב $L[c] = \text{כל המופעים של } c \text{ ב-} P$.
 3. עבור $n, 1, \dots, i = j$ בצע:
 4. $\text{עבור } j \in L[T[i]] \text{ בצע:}$
- $$Votes[i - j + 1] + 1 \rightarrow Votes[i - j + 1]$$
- . $O(n * max_c |L[c]|)$

דוגמה ריצה עבור מחוזות : $T = babaacb, P = abba$

$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=0000$	$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=0100$
$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=1200$	$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=1200$
$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=2201$	$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=2201$
$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=2301$	$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=2301$
$T \downarrow$ $babaacb$ P $abba$ $L[a]=\{1,4\}$ $L[b]=\{2,3\}$ $L[c]=\{\}$ $\text{Votes}=2301$	

Abrahamson's algorithm

יהי b שלם כלשהו שיוגדר בהמשך.

הגדרה: יהי $\Sigma \in c$ תו *rare*(נדיר) אם הוא מופיע לכל היוטר b פעמיים ב- P .

האלגוריתם

1. נחשב את מספר ההתאמות בין P ו- $T[i..i + m - 1]$ עבור *rare symbols* לכל i
2. נחשב את מספר ההתאמות בין P ו- $T[i..i + m - 1]$ עבור *frequent symbols* לכל i

דוגמה:

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,1,

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,1,2

Matches on frequent symbols: 1,

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,1,2

Matches on frequent symbols: 1,

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,1,2

Matches on frequent symbols: 1,2,

$$P = \text{abbc}aa$$

$$T = \text{dbc}bbaab \quad b = 2$$

Matches on rare symbols: 1,1,2

Matches on frequent symbols: 1,2,1

Rare symbols — voting algorithm

1. יהי $Votes$ מערך בגודל $1 + m - n$ המכיל אפסים

2. עבור כל $\Sigma \in c$ חשב $[c]L =$ כל המופיעים של c ב- P .

3. עבור $n, \dots, i = 1$ כך ש- $T[i]$ הוא rare בצע:

 עבור $j \in L[T[i]]$ בצע: .4

$Votes[i - j + 1] + 1 \rightarrow Votes[i - j + 1]$.5

סיבוכיות זמן $O(bn)$

דוגמה:

$P = abbc\textcolor{red}{a}aa$ $T = \textcolor{red}{d}bcbbbaab$ $Votes = 0,0,0$	$P = ab\textcolor{red}{b}c\textcolor{red}{a}aa$ $T = dbcbbaab$ $Votes = 0,0,0$
$P = abbc\textcolor{red}{a}aa$ $T = \textcolor{red}{d}bcbbbaab$ $Votes = 1,0,0$	$P = ab\textcolor{red}{b}c\textcolor{red}{a}aa$ $T = db\textcolor{red}{c}bbbaab$ $Votes = 1,0,0$
$P = abbc\textcolor{red}{a}aa$ $T = dbc\textcolor{red}{b}bbbaab$ $Votes = 1,0,0$	$P = ab\textcolor{red}{b}c\textcolor{red}{a}aa$ $T = dbc\textcolor{red}{b}bbbaab$ $Votes = 1,1,1$
$P = abbc\textcolor{red}{a}aa$ $T = dbcb\textcolor{red}{b}baab$ $Votes = 1,1,1$	$P = ab\textcolor{red}{b}c\textcolor{red}{a}aa$ $T = dbcb\textcolor{red}{b}baab$ $Votes = 1,1,2$
	$P = ab\textcolor{red}{b}c\textcolor{red}{a}aa$ $T = dbcbbaab\textcolor{red}{b}$ $Votes = 1,1,2$

Frequent symbols

1. עבור כל $c \in \Sigma$ frequent symbol c בצע:

 2. כאשר c מוחלף ב-1 ושאר התווים ב-0 $P_c = P$

 3. כאשר c מוחלף ב-1 ושאר התווים ב-0 $T_c = T$

 4. חשב $T_c \circ P_c$

 5. לכל i פלט $\Sigma_c(T_c \circ P_c)[i]$

דוגמה:

$P = abcaa, T = bccbaab$
 $P_a = 10011$
 $T_a = 0000110$
 $T_a \circ P_a =$

$P = abcaa, T = bccbaab$
 $P_a = 10011$
 $T_a = 0000110$
 $T_a \circ P_a = 1,$

$P = abcaa, T = bccbaab$
 $P_a = 10011$
 $T_a = 0000110$
 $T_a \circ P_a = 1, 2,$

$P = abcaa, T = bccbaab$
 $P_a = 10011$
 $T_a = 0000110$
 $T_a \circ P_a = 1, 2, 1$

$P = abcaa, T = bccbaab$
 $P_a = 10011$
 $T_a = 0000110$
 $T_a \circ P_a = 1, 2, 1$

בהתוכן: יש לכל היותר $\frac{m}{b}$ frequent symbols

סיבוכיות זמן: $O(\frac{m}{b} * n \log m)$

Time complexity

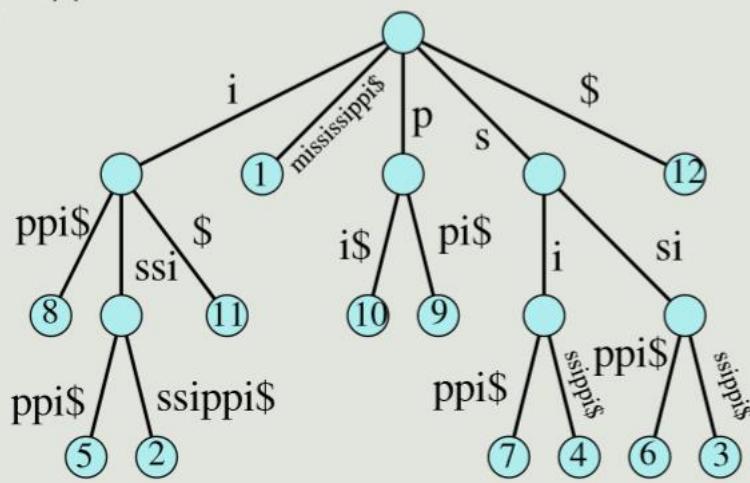
- Time complexity of the algorithm: $O(bn + \frac{m}{b} \cdot n \log m)$.
- The bound above is minimized when $b = \sqrt{m \log m}$.
- Time complexity: $O(n\sqrt{m \log m})$.

Suffix tree

� – עץ שומר קבוצה של מחוזות, בעץ יש על כל קשתתו בוודד כאשר כל מסלול מייצג מילה בודדת, יכולנו לתאר לכל מילה ענף בוודד בעץ אך רצינו לחסוך בזיכרון.

. הוא *trie* מכובץ שמכיל את כל הסופיות של מחוזות S .

$S = \text{mississippi\$}$.

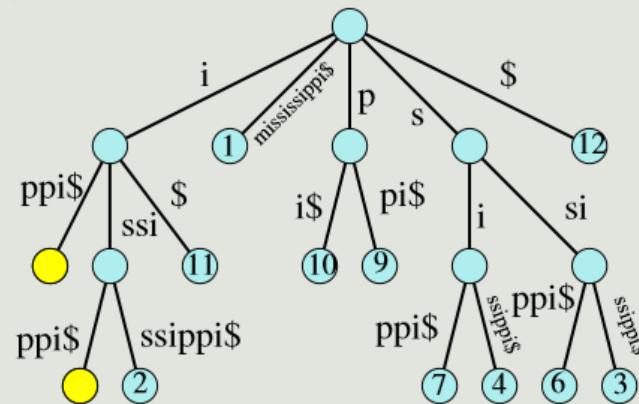


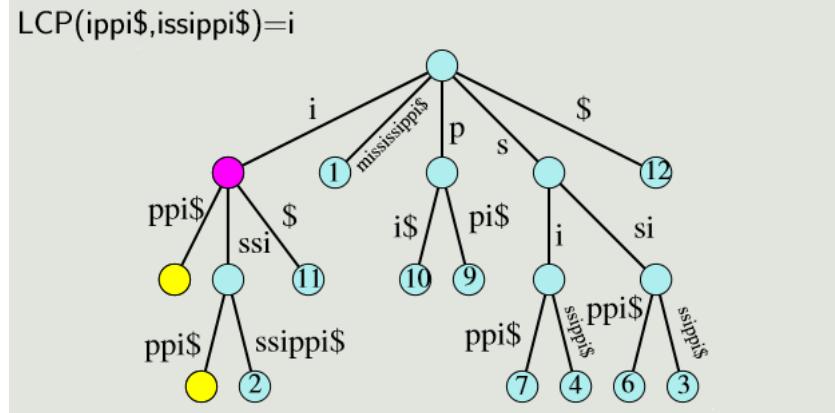
Computing LCP

– הרישא המשותפת הארוכה ביותר עבור 2 מחוזות, ע"י מציאת ה LCA של העלים המתאים ב*suffix tree*.

דוגמת מציאת LCP עבור 2 תת-מחוזות :

$$LCP(\text{ippi\$}, \text{issippi\$}) =$$





k -mismatches: The kangaroo method

Algorithm for the k -mismatches problem

1. נבנה עבור P ועבור T

2. עבור $i \leq n - m + 1$ בצע: (השוואת P ואת T)

(מספר התווים המשווים) $j = 0$.3

עבור $s = 1, \dots, k + 1$.4

$j = j + 1 + LCP(P[j + 1 \dots m], T[j + 1 \dots n])$.5

אם $j = m + 1$ דוח התאמה .6

אם $j \geq m$ צא .7

דוגמא : עבור $P = abcaabaccc$, $T = cabcdabbcccd$ $j = 4$

$P = \textcolor{red}{abca}abaccc$

$T = \textcolor{red}{cab}cdabbcccd \quad j = 4$

$P = \textcolor{red}{abca}aba\textcolor{green}{ccc}$

$T = \textcolor{red}{cab}cdabbcccd \quad j = 7$

$P = \textcolor{red}{abca}aab\textcolor{green}{accc}$

$T = \textcolor{red}{cab}cdabbcccd \quad j = 11$

ניתוח: סיבוכיות זמן $O(kn)$

- Building the suffix tree and preprocessing the tree for LCA queries takes $O(n + m) = O(n)$ time.
(assuming $\Sigma = \{1, \dots, n\}$)
- Checking one location i takes $O(k)$ time.
- Overall time complexity: $O(kn)$.

Faster algorithm

יהי b קבוע שלם כלשהו שנגדיר בהמשך.

הגדעה: $\Sigma \in c$ הוא *rare* אם הוא מופיע לכל היותר b פעמים ב- P .

נסתכל על 2 מקרים מרכזיים:

מקרה 1: יש לפחות $\left\lceil \frac{2k}{b} \right\rceil$ frequent symbols שונים.

מקרה 2: יש לפחות $\left\lceil \frac{2k}{b} \right\rceil$ frequent symbols שונים.

מקרה 1

1. נספור בצורה נפרדת את מספר ההתאמות של rare symbols על פני ה frequent symbols.

2. נספור את מספר ההתאמות של rare symbols: ע"י שימוש ב-*Voting algorithm* ב($O(bn)$).

3. נספור את מספר ההתאמות של frequent symbols: ע"י שימוש בקונבולוציה ב($O(\frac{k}{b} * n \log m)$).

4. סיבוכיות זמן כוללת ($O(bn + \frac{k}{b} * n \log m)$)

מקרה 2

1. נבחר $\left\lceil \frac{2k}{b} \right\rceil$ סמלים בעלי תדרות שונות $c_1, \dots, c_{\lceil \frac{2k}{b} \rceil}$.

2. לכל i נבחר b מופעים של c_i ב- P , תהי $L[c_i]$ קבוצה של מופעים נבחרים.

דוגמה:

$$P = \text{cabaacbcacdb } b = 3, k = 3, \hat{L}[a] = \{2, 4, 5\}$$

$$S = \text{cdbaaccacacdb } \text{Ham}(P, S) = 2, I(S) = 5 \geq 6 - 2$$

הגדעה: $I(S)$ = מספר ההתאמות בין P ו- S עבור האינדקסים הנבחרים.

עובדת: $I(S) \geq 2k - \text{Ham}(P, S)$

Voting algorithm

1. יהי *Votes* מערך בגודל $1 + m - n$ המכיל אפסים

2. לכל i כך ש- $T[i] \in \{c_1, \dots, c_{\lceil \frac{2k}{b} \rceil}\}$ בצע:

לכל $[i..j]$ בצע: .3

$\text{Votes}[i - j + 1] + 1 \rightarrow \text{Votes}[i - j + 1]$.4

לכל i כך ש- $\text{Votes}[i] \geq k$ בצע: .5

6. השווה את P ואת T ע"י שימוש בשיטת kangaroo. (אלגוריתם למעלה)

$$\text{Votes}[i] = I(T[i..i + m - 1]) \geq 2k - \text{Ham}(P, T[i..i + m - 1])$$

עובדה: אם $Votes[i] \geq k$ אז $Ham(P, T[i..i + m - 1]) \leq k$

ניתנות:

1. לפחות b , $b \leq vote$ פעמים.

2. סה"כ מספר ה ab $\leq bn$.

3. מספר ה a עבורם $\geq k$ $Votes[i]$ הוא $\frac{bn}{k}$.

סיבוכיות האלגוריתם כולם:

סיבוכיות זמן: $O(bn + \frac{k}{b} * n \log m)$

נבחר את b להיות $\sqrt{k \log m}$ ($n \sqrt{k \log m} = b$ עבורי הסיבוכיות היא $O(n \sqrt{k \log m})$)

Edit Distance with Moves

- מספר פעולות העריכה(הוספה/מחיקה/החלפה של תו/הזזה של תת מחוזה) המינימלי הנדרש כדי להגיע מ- S ל- T . פעולות העריכה הן החלפה, הורדה והוספה של אות. ($ED(S, T)$)

דוגמה:

$S = abcdef$, $T = deabcfg$, $ED(S, T) = 2$.

$abcdef \rightarrow deabcf \rightarrow deabcfg$

чисוב ($ED(S, T)$) היא בעיה $NP-hard$

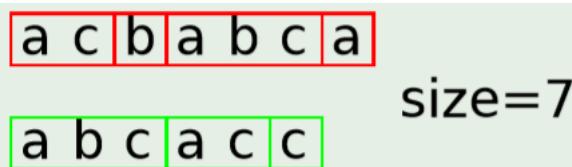
למרות זאת אנו רואים אלגוריתם המעריך את $ED(S, T)$ ב(n) O .

Common partition

הגדרה: common partition עבור 2 מחוזות S ו- T הוא חלוקה $S = S_1S_2\dots S_a$ ו- $T = T_1T_2\dots T_b$ כך שהsets של כל החלקים S_i באורך גדול ממש מ-1 שוים לsets של כל החלקים T_i באורך גדול ממש מ-1.

גודל החלוקה הוא $a + b$.

דוגמה:



הגדרה: ($CP(S, T)$) הוא הגודל המינימלי של common partition עבור 2 מחוזות S ו- T .

למה: $ED(S, T) \leq CP(S, T)$

הוכחה: יהיו $S_1S_2\dots S_a$ ו- $T_1T_2\dots T_b$ common partition $T = T_1T_2\dots T_b$ ו- $S = S_1S_2\dots S_a$ מוגדל מינימלי.

היא c מספר ה S_i שגדלים גדול ממש מ-1.

ניתן להפוך את S ל- T בתהליך הבא:

אלגוריתמים למחוזות / סיכום מאות אורי שביט

1. מחק את כל האותיות מ-S ששווים לחלק מ-S שגדלים 1.
2. סדר מחדש של החלקים.
3. הוסף את כל התווים של 7 ששווים לחלקים מ-S שגדלים 1. מספר הפעולות הוא לכל היותר $(c - 1) + (b - c) + (a - b)$ שכן קטן ממש $a + b + c$.

עבור S ו-T:

דוגמת הרצה:

a c b a b c a

a b c a c c

$$a = 4, b = 3, c = 2$$

a c b a b c a → a c a b c → a b c a c → a b c a c c

Iterated parsing scheme

בנייה את המחרוזות S_2, T_2 מ-S ו-T בצורה הבאה:

1. נחלק את התווים של S ו-T לפסוקיות (phrases). פסוקית (phrase) היא כל ריצה מקסימלית לא טריוויאלית. שאר האותיות מוחולקות לפסוקיות מגודל 1 או 2.
2. נחליף את כל הפסוקיות הגדולות ממש מ-1 באות חדשה. אותה אות חדשה משומשת בכל מופע חדש של פסוקית. (בהתאם בין המחרוזות ואותה פסוקית בתוך המחרוזת)
3. נמשיך בתהליך על T_2, S_2 עד שנגיע למחרוזות T_h, S_h באורך 1.

הגדרה: $V(S) = \text{וקטור המכיל את כל אות } c, \text{ מספר המופיעים של } c \text{ במחוזת הראשונה מ...}, S, S_2, S_3, \dots$.
את c .

S_5 m

S ₄	g	k						
S ₃	g	b	h					
S ₂	d	e	b	a	c			
S	a	a	a	b	c	b	a	c

T_5 n

T ₄	i		l					
T ₃	i	j	j					
T ₂	e	f	b	f	b	f		
T	b	c	a	a	b	a	a	a

	$V(S)$	$V(T)$	Δ
a	4	6	-2
b	2	3	-1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	-3
g	1	0	1
h	1	0	1
i	0	1	-1
j	0	2	-2
k	1	0	1
l	0	1	-1
m	1	0	1
n	0	1	-1

דוגמת הרצה: (Iterated parsing scheme)

S a a a b c b a c

S a a a b c b a c

T b c a a b a a b a a	S ₂ d e b a c	S a a a b c b a c
-----------------------	--------------------------	-------------------

T b c a a b a a b a a	S ₂ d e b a c	S a a a b c b a c
-----------------------	--------------------------	-------------------

T ₂ e f b f b f	T b c a a b a a b a a
S ₃ g b h	S ₂ d e b a c
S ₂ d e b a c	S a a a b c b a c

T ₂ e f b f b f	T b c a a b a a b a a
S ₅ m	S ₄ g k
S ₄ g b h	S ₃ g b h
S ₂ d e b a c	S ₂ d e b a c
S a a a b c b a c	S a a a b c b a c

T ₃ i j j	T ₂ e f b f b f
T b c a a b a a b a a	T i l
T ₃ i j j	T ₂ e f b f b f
T b c a a b a a b a a	T b c a a b a a b a a

T ₅ n	T ₄ i l
T ₄ i l	T ₃ i j j
T ₃ i j j	T ₂ e f b f b f
T ₂ e f b f b f	T b c a a b a a b a a

Upper bound

הגדרה: לכל וקטור (x_1, \dots, x_k) יהי $X = (x_1, \dots, x_k)$.

טענה: לכל איטרציה *parsing scheme* של S ו T $|V(S) - V(T)| \leq 4|V(S) - V(T)|$.

על מנת להוכיח את הлемה אנו צריכים להראות $|V(S) - V(T)| \leq 4|V(S) - V(T)|$.

ההוכחה משתמשת במתודה הבהא: אנו מזמנים $\$$ בתווים של המוחוזות $S_h \dots S_1 S_2 \dots T_h \dots T_1$. ה-

מכסם אלמנטי $V(T) - V(S)$.

נגידר את החלקים של S ו T , חלקיים אלו "משולמים" לפי החלפת $\$$.

m מופיע פעם ראשונה ברמה 5 ומופיע רק ב- S_5 , נשים $\$4$ במופיע של m . נחיב ב- $[m](V(S) - V(T))$

n מופיע פעם ראשונה ברמה 5 ומופיע רק ב- T_5 , נשים $\$4$ במופיע של n . נחיב ב- $[n](V(T) - V(S))$

g מופיע פעם ראשונה ברמה 4. נשים $\$2$ במופיע של g , שנלקח $\$4$ של m .

k מופיע פעם ראשונה ברמה 4 והוא מופיע ב- S_4 בלבד, נשים $\$4$ במופיע שלו, נחיב ב- $[k](V(S) - V(T))$

l מופיע פעם ראשונה ברמה 4 והוא מופיע ב- T_4 בלבד, נשים $\$4$ במופיע שלו, נחיב ב- $[l](V(T) - V(S))$

נמשיך באותו תהליך פעוף כלפי מטה עד לרמה התחתונה.

נשים $\$2$ במופיע של b ב- S_2 , ניקח $\$2$ מ- b_3 .

f מופיע פעם ראשונה ברמה 2 והוא מופיע רק ב- T_2 . נשים $\$4$ ב- f המופיעים של f . נחיב ב- $[f](V(S) - V(T))$

e מופיע פעם ראשונה ברמה 2 ומופיע ב- T_2 וב- S_2 . נסמן את המופיע של e ב- T_2 וב- S_2 .

ההרוחבות של e ב- S ו T הן *common partition*. העלות של חלקיים אלו משולמים ע"י $\$$ מהמופיע של e .

ההרוחבה של התווים המסומנים מסומנת, נתעלם מתווים אלו.

a מופיע 4 פעמים ב- S_1 ו-6 פעמים ב- T_1 , נחישב את 4 המופיעים של a ב- S_1 וב- T_1 .

נשים $\$2$ לכל ריצה של a (מחשיבים רק 4 מופיעים ב- T). ה-\$-ים הללו נלקחים מ- S_2, T_2 .

נחלק את הריצות הללו לחלקיים באורכים מתאימים, נסמן את התווים של הריצות הללו.

מספר החלקיים הוא לכל היותר פעמיים ממספר הריצות, כלומר $\$$ המאוחסנים בrizות אלו מכוסות את עלות החלקיים.

נשים $\$2$ בכל אחד מ-2 המופיעים של a ב- T_1 . נחיב ב- $[a](V(S) - V(T))$.

ונעשה את אותו תהליך עברו b ו c .

לבסוף כל אות לא מסומנת ב- T_1, S_1 מגדיר חלק. $\$$ שמאוחסן בתווים אלו מכוסה את עלות החלק.

S_5 m4

S_4	g	k
S_3	g	b h
S_2	d e	b a c
S	a a a b c b a c	

T_5 n4

T_4	i	j
T_3	i	j j
T_2	e f	b f b f
T	b c a a b a a b a a	

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4	g 2	k
S_3	g	b h
S_2	d e	b a c
S	a a a b c b a c	

T_5 n4

T_4	i	j
T_3	i	j j
T_2	e f	b f b f
T	b c a a b a a b a a	

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4	g 2	k 4
S_3	g	b h
S_2	d e	b a c
S	a a a b c b a c	

T_5 n4

T_4	i	j
T_3	i	j j

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1

S_5 m2

S_4 g 2 k 4

S_3 g b h

S_2 d e b a c

S a a a b c b a c

T_5 n 2

T_4 i 2 l 4

T_3 i j j

T_2 e f b f b f

T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g 2 k 2

S_3 g 4 2 b h 4

S_2 d e b a c

S a a a b c b a c

T_5 n 2

T_4 i 2 l 4

T_3 i j j

T_2 e f b f b f

T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g 2 k 2

S_3 g 4 2 b h 4

S_2 d e b a c

S a a a b c b a c

T_5 n 2

T_4 i 2 l 4

T_3 i 4 j 4 j 4

T_2 e f b f b f

T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g 2 k 2

S_3 g 2 0 b h 0

S_2 d 4 2 e 2 b a 2 c 2

S a a a b c b a c

T_5 n 2

T_4 i 2 l 4

T_3 i 4 j 4 j 4

T_2 e f b f b f

T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g2 k2

S_3 g2 0b h0

S_2 d4 2e 2b a2c 2

S a a a b c b a c

T_5 n2

T_4 i2 l4

T_3 i2 j2 j2

T_2 e2 f4 2b f4 2b f4

T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g2 k2

S_3 g2 0b h0

S_2 d4 e2 b a2c 2

S a a a b c b a c

T_5 n2

T_4 i2 l4

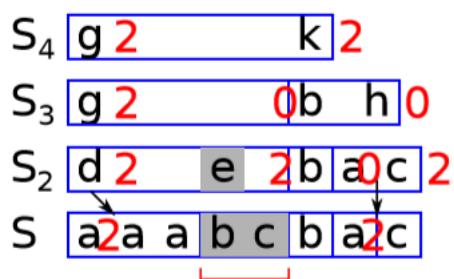
T_3 i2 j2 j2

T_2 e f4 2b f4 2b f4

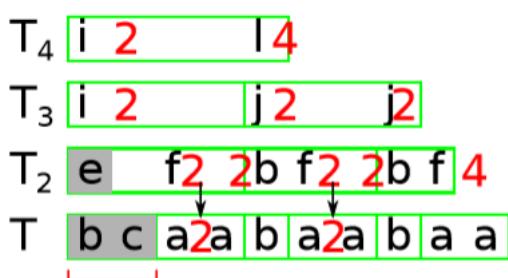
T b c a a b a a b a a

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

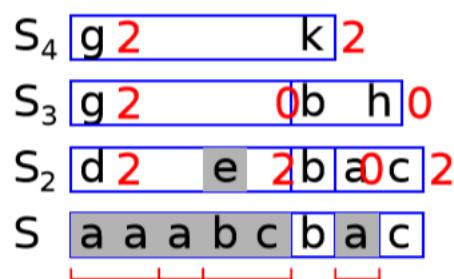


T_5 n2

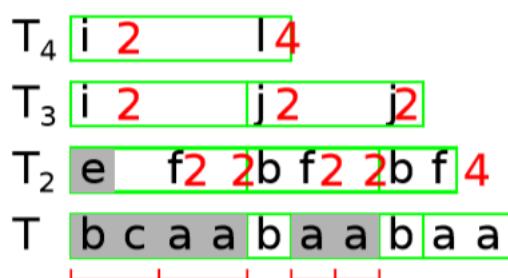


	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2



T_5 n2



	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g2 k2

S_3 g2 0b h0

S_2 d2 e2b2a0c2

S aaaa bcbac
 ↓ ↓

T_5 n2

T_4 i2 l4

T_3 i2 j2 j2

T_2 e f2 2b f2 2b f4

T bcaaa bbaab a4a4
 ↓ ↓ ↓ ↓

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g2 k2

S_3 g2 0b h0

S_2 d2 e0b2a0c2

S aaaa b2baca
 ↓ ↓

T_5 n2

T_4 i2 l4

T_3 i2 j2 j2

T_2 e f2 0b f2 2b f4

T bcaaa b2bab a4a4
 ↓ ↓ ↓

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g 2 k 2

S_3 g 2 0 b h 0

S_2 d 2 e 0 b a 0 c 2

S a a a b c b a c

T_5 n 2

T_4 i 2 l 4

T_3 i 2 j 2 j 2

T_2 e f 2 0 b f 2 2 b f 4

T b c a a b a a b 4 a 4

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

S_5 m2

S_4 g 2 k 2

S_3 g 2 0 b h 0

S_2 d 2 e 0 b a 0 c 2

S a a a b c b a c 4

T_5 n 2

T_4 i 2 l 4

T_3 i 2 j 2 j 2

T_2 e f 2 0 b f 2 2 b f 4

T b c a a b a a b 4 a 4 a 4

	$V(S)$	$V(T)$	$ \Delta $
a	4	6	2
b	2	3	1
c	2	1	1
d	1	0	1
e	1	1	0
f	0	3	3
g	1	0	1
h	1	0	1
i	0	1	1
j	0	2	2
k	1	0	1
l	0	1	1
m	1	0	1
n	0	1	1

Lower bound

$$.ED(S, T) \leq 4 \left| |V(S) - V(T)| \right|_1$$

אם אנחנו יכולים להראות גם $|V(S) - V(T)| \leq ED(S, T) * f(n)$

ע"י שוויון המשולש זה מראה את המקרה ש $ED(S, T) = 1$ וראה ש $f(n) = 1$

Fixed parsing

נניח שאנו מחלקים כל מחוזת לפסוקיות בגודל 2.

אם T מושג מס ע"י החלפה(*substitution*) אז $.||V(S) - V(T)|| = O(\log n)$

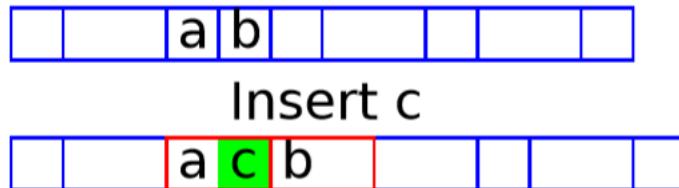
S_4	k	T_4	$ $
S_3	$h \quad i$	T_3	$j \quad i$
S_2	$d \quad e \quad f \quad e$	T_2	$d \quad g \quad f \quad e$
S	$b \quad a \quad b \quad c \quad a \quad c \quad b \quad c$	T	$b \quad a \quad c \quad c \quad a \quad c \quad b \quad c$

למרות זאת, אם T מושג מס ע"י הכנסה(*insertion*) אז $.||V(S) - V(T)|| = \Theta(n)$

S_4	m	T_4	$n \quad c$
S_3	$i \quad j$	T_3	$k \quad \quad c$
S_2	$d \quad e \quad f \quad e$	T_2	$f \quad f \quad g \quad h \quad c$
S	$b \quad a \quad b \quad c \quad a \quad c \quad b \quad c$	T	$a \quad b \quad a \quad b \quad c \quad a \quad c \quad b \quad c$

Locally consistent parsing

אנו רוצים חלוקה של המחרוזות S ולפסוקיות בגודל 1 או 2 (ריצות לא טריוויאליות) כגון פעולה עERICA אחת המשנה פסוקיות ב(1).



חלוקת $\Sigma_1 \cup \Sigma_0 = \Sigma$ של Σ מגדירה את ה *parsing* של מחרוזת S :

1. כל ריצה לא טריוויאלית היא פסוקית.
2. כל תת מחרוזת ab כך ש $a \in \Sigma_0, b \in \Sigma_1$ ולא שייכים לריצה לא טריוויאלית היא פסוקית.

Example

$a \quad c \quad b \quad a \quad b \quad b \quad a \quad b \quad c \quad b \quad \Sigma_0 = \{a\} \quad \Sigma_1 = \{b, c\}$

למה: בהינתן T, S אם חלוקה של S נבחרת רנדומלית אז צפוי מספר פסוקיות *parsing* של S ו T לכל היותר $\frac{3}{4}(|T| + |S|)^2 + \frac{1}{2}$.

הוכחה: נניח שאין ריצות לא טריוויאליות. המספר הצפוי של פסוקיות מגודל 2 הוא $\frac{|S|+|T|-2}{4}$.

המספר הצפוי של פסוקיות הוא $\frac{|S|+|T|-2}{4} \cdot |S| + |T|$.

ההוכחה של המקרה שאין פסוקיות לא טריויאליות לא יוכח פה. מש"ל.

המספר הצפוי של פסוקיות עבור parsing של חלוקה רנדומלית של S הוא לפחות $\frac{1}{2}(|S| + |T|) + \frac{3}{4}(|S| + |T|)$.

חלוקת של S שהינה parsing כולל לפחות $\frac{1}{2}(|S| + |T|)$ פסוקיות יכול להיות חסום בזמן לינארי ע"י שימוש במתודת conditional probability.

בහינת S ו T אנו יכולים להרכיב בזמן לינארי $S_1, T_1 \dots S_h, T_h$ - iterated parsing.

Method of conditional probability

יהי X משתנים רנדומליים כך ש $X = f(x_1, \dots, x_n)$, כאשר x_1, \dots, x_n הם משתנים בליאניים רנדומליים.

אנו רוצים למצוא השמה a_1, \dots, a_n ל x_1, \dots, x_n עבורו הערך של X הוא לפחות $E[X]$.

נחשב $E[X|x_n = 0] \leq E[X|x_n = 1]$ אם $E[X|x_n = 0] = 0$ ונגדיר $a_n = 1$ אחרת.

נמשיך רקורסיבית על $X' = f(x_1, \dots, x_{n-1}, a_n)$.

Locally consistent parsing

פעולות ערך ייחידה משנה ($O(1)$ פסוקיות).

$S \quad \boxed{\text{b}} \text{ a } \boxed{\text{b}} \text{ c } \boxed{\text{a}} \text{ c } \boxed{\text{b}} \text{ c}$ $\Sigma_0 = \{\text{a}\}$ $\Sigma_1 = \{\text{b}, \text{c}\}$

$T \quad \boxed{\text{b}} \text{ b } \boxed{\text{a}} \text{ b } \boxed{\text{c}} \text{ a } \boxed{\text{c}} \text{ b } \boxed{\text{c}}$



הגדרה: $I_k =$ אורך של פסוקיות לא מתאימות $.S_k, T_k$

הגדרה: $V(S) =$ וקטור המכיל את כל אות c , מספר המופיעים של c במחuzeות הראשונה מ... S, S_2, S_3, \dots, S_k שמכיל את c .

$S_5 \text{ m}$	$V(S)$
$S_4 \quad \boxed{\text{g}} \quad \boxed{\text{k}}$	a 4
$S_3 \quad \boxed{\text{g}} \quad \boxed{\text{b}} \quad \boxed{\text{h}}$	b 2
$S_2 \quad \boxed{\text{d}} \quad \boxed{\text{e}} \quad \boxed{\text{b}} \quad \boxed{\text{a}} \quad \boxed{\text{c}}$	c 2
$S \quad \boxed{\text{a}} \quad \boxed{\text{a}} \quad \boxed{\text{a}} \quad \boxed{\text{b}} \quad \boxed{\text{c}} \quad \boxed{\text{b}} \quad \boxed{\text{a}} \quad \boxed{\text{c}}$	d 1
	e 1
	f 0

הגדרה: $V_k(S) = V(S)$ המתאים/מקביל/מוגבל (restricted) לאותיות שמופיעות בא הרמות הראשונות.

$$I_k + \|V_k(S) - V_k(T)\|_1 = I_{k-1} + \|V_{k-1}(S) - V_{k-1}(T)\|_1 + O(1).$$

$$\implies I_k + \|V_k(S) - V_k(T)\|_1 = O(k).$$

$$\implies \|V(S) - V(T)\|_1 = O(\log n).$$

$$\begin{aligned}
 l_k + \|V_k(S) - V_k(T)\|_1 &= l_{k-1} + \|V_{k-1}(S) - V_{k-1}(T)\|_1 + O(1). \\
 \implies l_k + \|V_k(S) - V_k(T)\|_1 &= O(k). \\
 \implies \|V(S) - V(T)\|_1 &= O(\log n).
 \end{aligned}$$

S_4  $\Sigma_0 = \{h\}$ $\Sigma_1 = \{b, c, d, e\}$

S_3  $\Sigma_0 = \{g\}$ $\Sigma_1 = \{b, c, d, e\}$

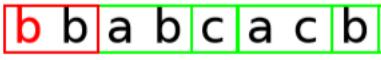
S_2  $\Sigma_0 = \{f\}$ $\Sigma_1 = \{b, c, d, e\}$

S  $\Sigma_0 = \{a\}$ $\Sigma_1 = \{b, c\}$

T_4  $|l_4 + \|V_4(S) - V_4(T)\|_1| = 6 + 4$

T_3  $|l_3 + \|V_3(S) - V_3(T)\|_1| = 5 + 3$

T_2  $|l_2 + \|V_2(S) - V_2(T)\|_2| = 4 + 2$

T  $|l_1 + \|V_1(S) - V_1(T)\|_1| = 3 + 1$

Approximation algorithm

בහינתן S ו T אנו מרכיבים ב(n) O וקטוריים $V(S), V(T)$ כאשר:

$$ED(S, T) \leq 4 \cdot \|V(S) - V(T)\|_1 \quad .1$$

$$\|V(S) - V(T)\|_1 \leq ED(S, T) * O(\log n) \quad .2$$

לכן, יש אלגוריתם לינארי להערכתה עם יחס הערכה של $O(\log n)$.

אלגוריתמים למחוזות / סיכום מאות אורי שביט

הרצאה 20 ואחרונה תודה לאל – Two-Dimensional Matching

P	<table border="1"> <tr><td>a</td><td>b</td></tr> <tr><td>c</td><td>a</td></tr> </table>	a	b	c	a	T	<table border="1"> <tr><td>a</td><td>a</td><td>b</td><td>a</td></tr> <tr><td>b</td><td>c</td><td>a</td><td>b</td></tr> <tr><td>b</td><td>a</td><td>c</td><td>a</td></tr> <tr><td>a</td><td>a</td><td>a</td><td>a</td></tr> </table>	a	a	b	a	b	c	a	b	b	a	c	a	a	a	a	a
a	b																						
c	a																						
a	a	b	a																				
b	c	a	b																				
b	a	c	a																				
a	a	a	a																				

הגדרת הבעיה:

קְלֻט: $m \times m$ string $P, n \times n$ string T

פלט: כל המופעים של P בט'

ליגמאנ

$$P = \begin{pmatrix} a & b \\ c & a \end{pmatrix} \quad T = \begin{pmatrix} a & a & b & a \\ b & c & a & b \\ b & a & c & a \\ a & a & a & a \end{pmatrix}$$

מיקום (x,y) ב**T** מתאים אם $x..x + m - 1, y..y + m - 1] = P$:

בדוגמא לעיל המיקומים: (1,2), (2,3) מתאימים.

Witness table

ביקח 2 עותקים של C ובדיז את העותק השני, בעתיק \hat{c} שוררת למטה ועומדות ימינה. (\hat{c}, b יכולים להיות > 0).

(מתאים הזזה) אם סמלים החלק החופף מתאים.

a	b	c
b	a	a
a	b	c

a	b	c
b	a	a
a	b	c
b	a	a
a	b	c

$$\begin{aligned}a &= 2 \\b &= 0\end{aligned}$$

a	b	c
b	a	c
a	b	c
a	b	c

$$\begin{aligned}a &= 1 \\b &= 0\end{aligned}$$

הגדירה: עבור הuzzת א' התאימה a, b , witness הוא מיקום באזור החופף שבו הסמלים לא מתאימים:

pair (x, y) s.t. $P[x, y] \neq P[x + a, y + b]$

הגדירה: $witness = W[a,b]$ עבור הZZZ a, b

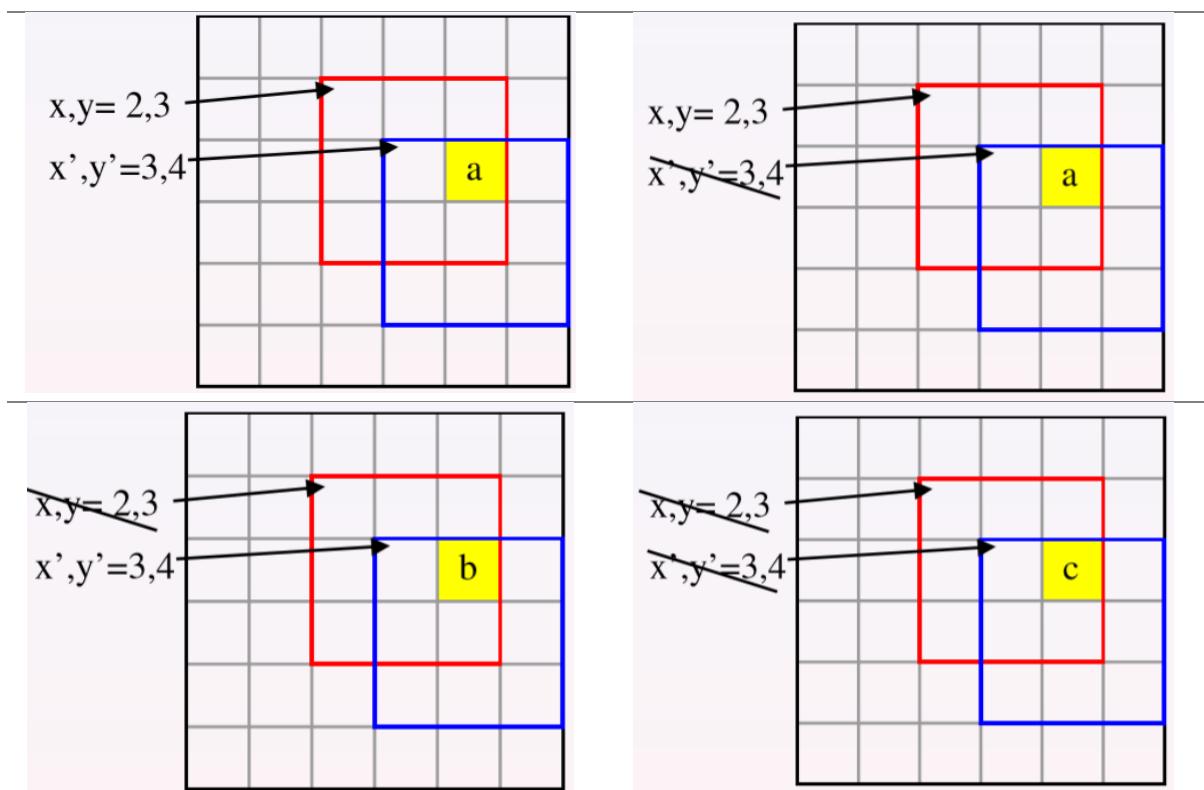
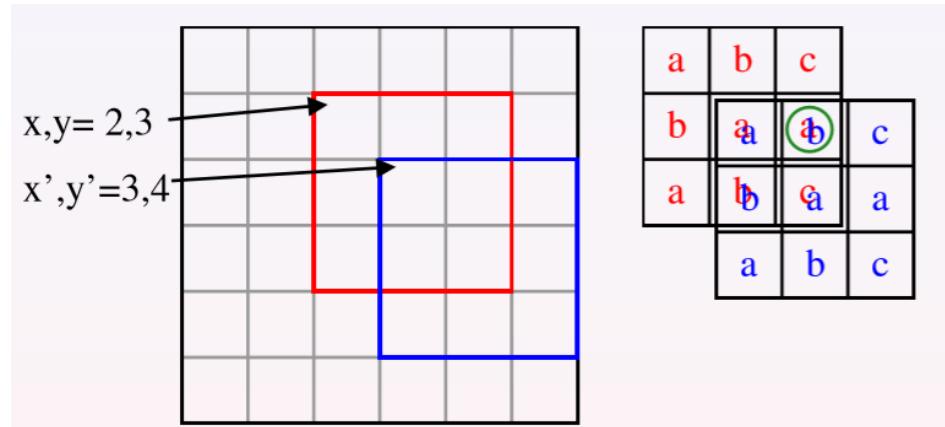
דוגמאות

$a = 1$ $b = 0$	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td>a</td><td>b</td><td>c</td></tr> <tr><td>b</td><td>a</td><td>c</td></tr> <tr><td>a</td><td>b</td><td>a</td></tr> <tr><td>a</td><td>b</td><td>c</td></tr> </table>	a	b	c	b	a	c	a	b	a	a	b	c	W	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>(1,1)</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>													(1,1)									-2 -1 0 1 2							
a	b	c																																										
b	a	c																																										
a	b	a																																										
a	b	c																																										
(1,1)																																												
$a = 1$ $b = 1$	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td>a</td><td>b</td><td>c</td></tr> <tr><td>b</td><td>a</td><td>c</td></tr> <tr><td>a</td><td>b</td><td>a</td></tr> <tr><td>a</td><td>b</td><td>c</td></tr> </table>	a	b	c	b	a	c	a	b	a	a	b	c	W	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>(1,1)</td><td>(1,2)</td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>													(1,1)	(1,2)								-2 -1 0 1 2							
a	b	c																																										
b	a	c																																										
a	b	a																																										
a	b	c																																										
(1,1)	(1,2)																																											
$a = 2$ $b = 0$	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td>a</td><td>b</td><td>c</td></tr> <tr><td>b</td><td>a</td><td>a</td></tr> <tr><td>a</td><td>b</td><td>c</td></tr> <tr><td>b</td><td>a</td><td>a</td></tr> <tr><td>a</td><td>b</td><td>c</td></tr> </table>	a	b	c	b	a	a	a	b	c	b	a	a	a	b	c	W	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td>(1,1)</td><td>(1,2)</td><td></td></tr> <tr><td>None</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>													(1,1)	(1,2)		None						-2 -1 0 1 2				
a	b	c																																										
b	a	a																																										
a	b	c																																										
b	a	a																																										
a	b	c																																										
(1,1)	(1,2)																																											
None																																												
$a = -1$ $b = 1$	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td></td><td>a</td><td>b</td><td>c</td></tr> <tr><td>a</td><td>b</td><td>c</td><td>a</td></tr> <tr><td>b</td><td>a</td><td>a</td><td>c</td></tr> <tr><td>a</td><td>b</td><td>c</td><td></td></tr> </table>		a	b	c	a	b	c	a	b	a	a	c	a	b	c		W	<table border="1" style="border-collapse: collapse; font-size: 1.5em;"> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> <tr><td></td><td>(2,2)</td><td></td></tr> <tr><td>(1,1)</td><td>(1,2)</td><td></td></tr> <tr><td>None</td><td></td><td></td></tr> <tr><td></td><td></td><td></td></tr> </table>														(2,2)		(1,1)	(1,2)		None						-2 -1 0 1 2
	a	b	c																																									
a	b	c	a																																									
b	a	a	c																																									
a	b	c																																										
	(2,2)																																											
(1,1)	(1,2)																																											
None																																												

Duel

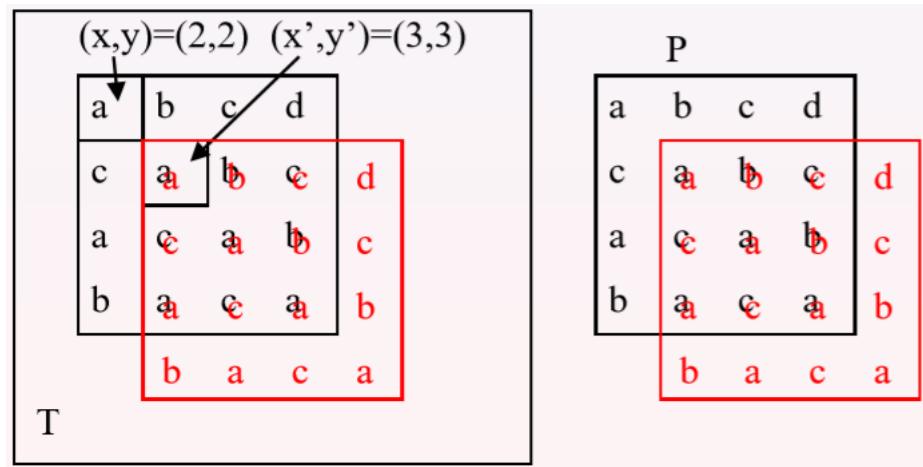
יהיו $(x, y), (x', y')$ מיקומים ב \mathcal{C} כאשר $x \leq x', y \leq y'$. אם $x - x', y - y' \leq 1$ הוא הצעת חוסר התאמה \mathcal{C} לא יכול להופיע גם ב (y, x) וגם ב (y', x') .

ע"י קרייתתו אחד ב \mathcal{C} , אנו יכולים לשולב מופע של \mathcal{C} בלפחות אחד המיקומים $(x, y), (x', y')$.



Consistency

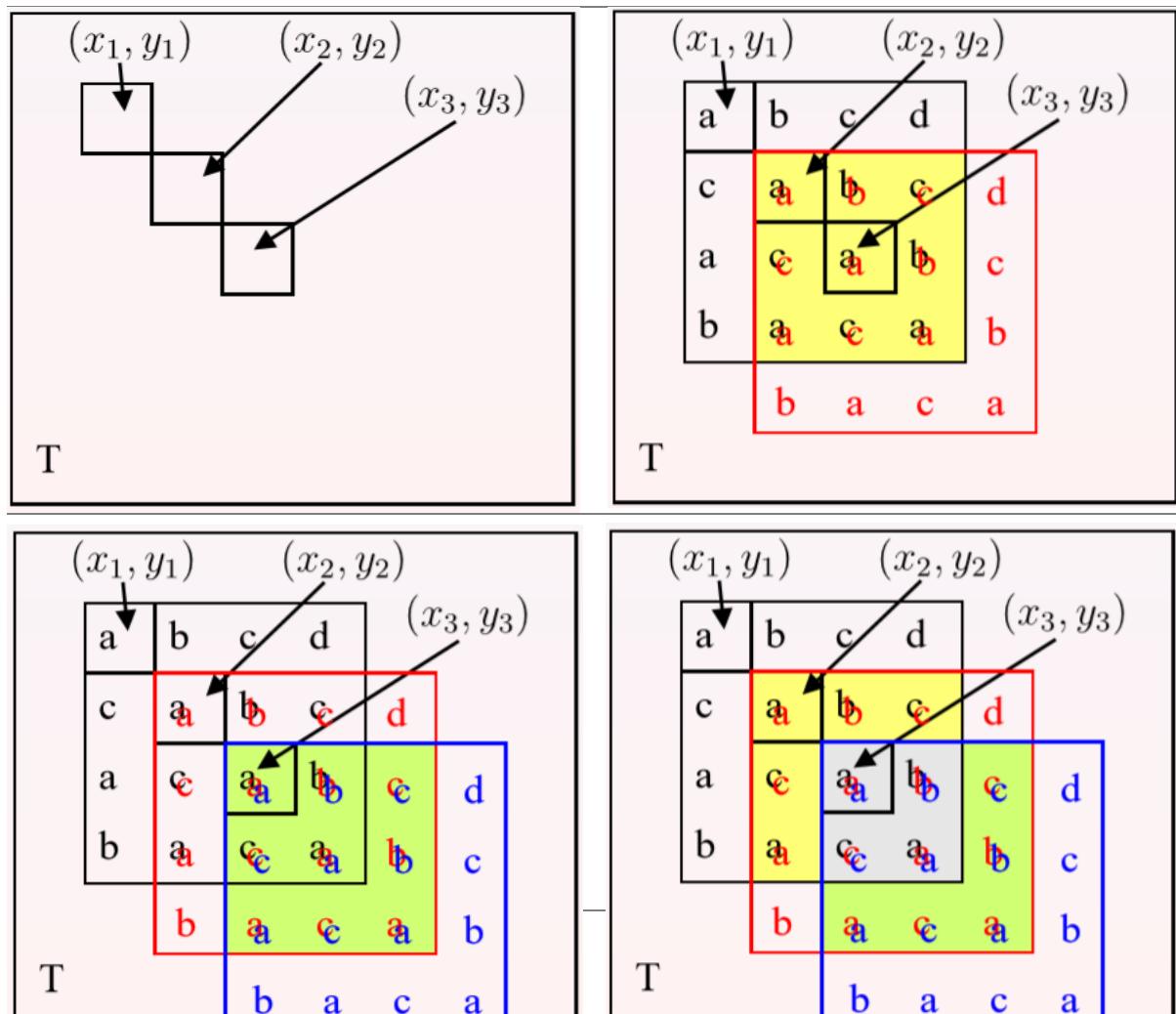
הגדעה: 2 מיקומים (x', y') , (x'', y'') נקראים **consistent** אם $x' - x, y' - y$ היא התאמה בהזזה.



בambilim achrotot $(x', y'), (x, y)$ נקראים **consistent** אם נשים את P על (x', y') ועל (y, x) הסמלים יהיו איזור חופף.

Semi-transitivity of the consistency relation

למה: נניח ש (x_1, y_1) **consistent** (x_2, y_2) . אם $x_1 \leq x_2 \leq x_3, y_1 \geq y_2 \geq y_3$ או $x_1 \leq x_2 \leq x_3, y_1 \leq y_2 \leq y_3$ אז (x_1, y_1) **consistent** (x_3, y_3) .



מקנות:

יחוך הוא חצי טרנסיציטיבי.

אם 2 מקומות הם לא *consistent*, אז' שניים לא יכולים להיות התאמה. בשימוש *duel* אנו יכולים לשולב לפחות אחת אחד המיקומים ב(1)0.

Duel – ה'ו) (x', y'), (y, x) מיקומיים ב T כאשר $y \leq x'$, $y' \leq x$. אם $y - x' = y' - x$ הוא הצעת חוסר התאמה אז C לא יכול להופיע גם ב(y, x) וגם ב(y', x'). ע"י קרייתתו אחד ב T , אנו יכולים לשולב מופיע של C ב לפחות אחד המיקומיים (y', y), (x, x').

Algorithm's outline

1. תהא L קבוצת כל המיקומים (y, x) כאשר $1 \leq x \leq n - m + 1, 1 \leq y \leq n - m + 1$.
 2. בצע אלימינציה למיקומים ב- L לפי *duels* עד שgal המיקומים ב- L הם זוגות *consistent*.
 3. אמת את המיקומים ב- L .

* אם בשלב 2 אנו מבצעים את האלגוריתם נאיבי בעבר כל 2 מיקומים, האם הם *consistent* () ואם לא לבצע duel). תחילה זה עשוי לדרוש $(^4n)\theta$. על כן נראה איך מבצעים זאת ב- $(n^2)\theta$.

Column consistency

- .1. $C_y = \emptyset$ (רישימה ריקה)

.2. עבור כל $L \in (y, x)$ בסדר יורד עבור x :

: $head(C_y) \neq \emptyset$ ולא (x, y) עם $consistent(C_y)$

.3. כל עוד $\emptyset \neq C_y$ ו (x, y) לא $consistent(C_y)$

.4. בצע השוואת duel בין (x, y) ו- $head(C_y)$

.5. אם $head(C_y)$ אבד אז מחק אותו מ- C_y

.6. אם (x, y) אבד את צא.

.7. אם (x, y) לא אבד אז הוסף אותו לראש C_y .

T							
P	a	b	a	a	b		
	a	a	b	c	c		
	a	b	a	a	b		
	a	a	b	c	c		
	a	b	a	a	b		

דוגמת הרצה:

T		
	(5,5)	

P	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b

T		
	(4,5)	

P	a	b	a	a	b
	a	a		c	c
	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b

$$W[1,0]=(2,3)$$

(5,5)

consistent

T		
	(3,5)	

T		
	(3,5)	

P	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b

T		
	(2,5)	

P	a	b	a	a	b
	a	a		c	c
	a	b	a	a	b
	a	a	b	c	c
	a	b	a	a	b

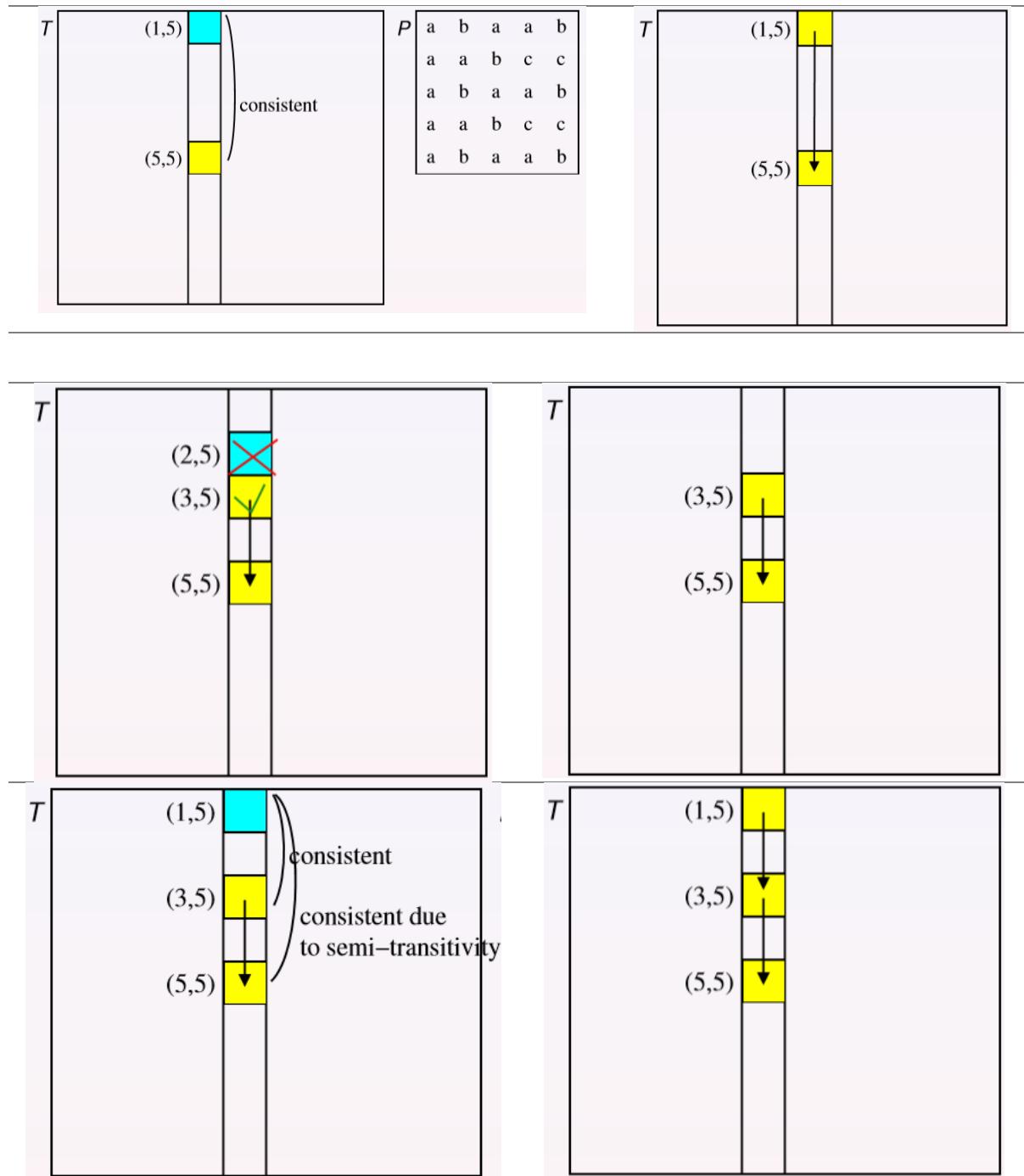
$$W[1,0]=(2,3)$$

not consistent

(5,5)

not consistent

אלגוריתמים למחuzeות / סיכום מאות אורי שביט



Correctness

למה: בכל שלב באלגוריתם המיקומיים ב- C_y הם זוגות consistent (עקביים).

הוכחה: נניח שאנו בוחנים מיקום (y, x) . אם (y, x) מפסיד duel או \emptyset , אז בוודאי C_y מקיימים את הלמה אחרת, (y, x) הוא consistent עם האלמנט הראשון של C_y . לפי טרנסיטיביות של יחס עקביות, אז עקיי עם כל האלמנטים ב- C_y . **מש"ל.**

סיבוכיות זמן: הזמן לטיפול בעמודה אחת הוא $O(1)$, בכל עמודה יש $n - 1$ duels.

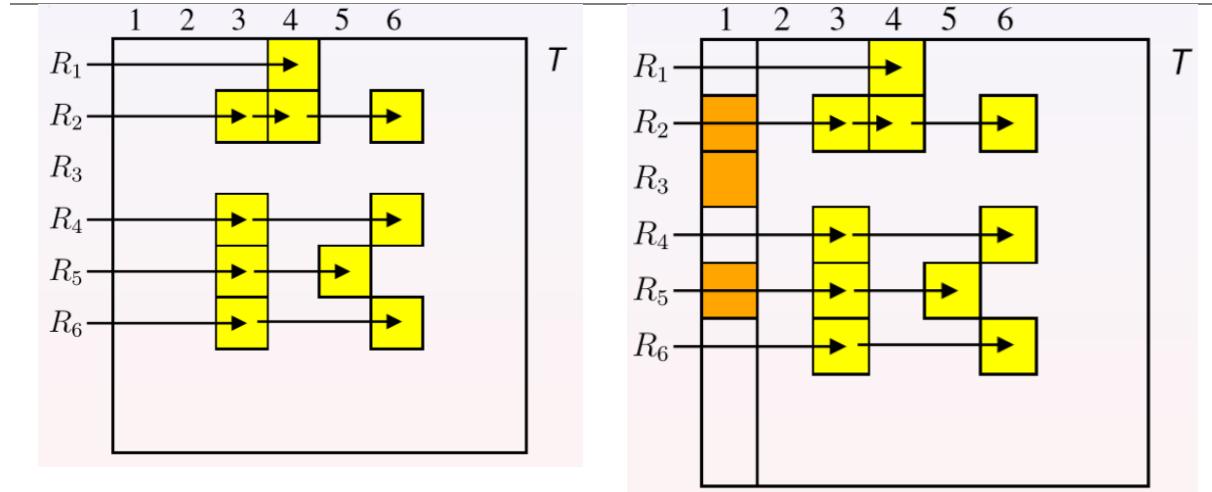
סיבוכיות זמן כוולת: הסיבוכיות הכלולת של שלב זה היא $(^2n)O$.

Pairwise consistency

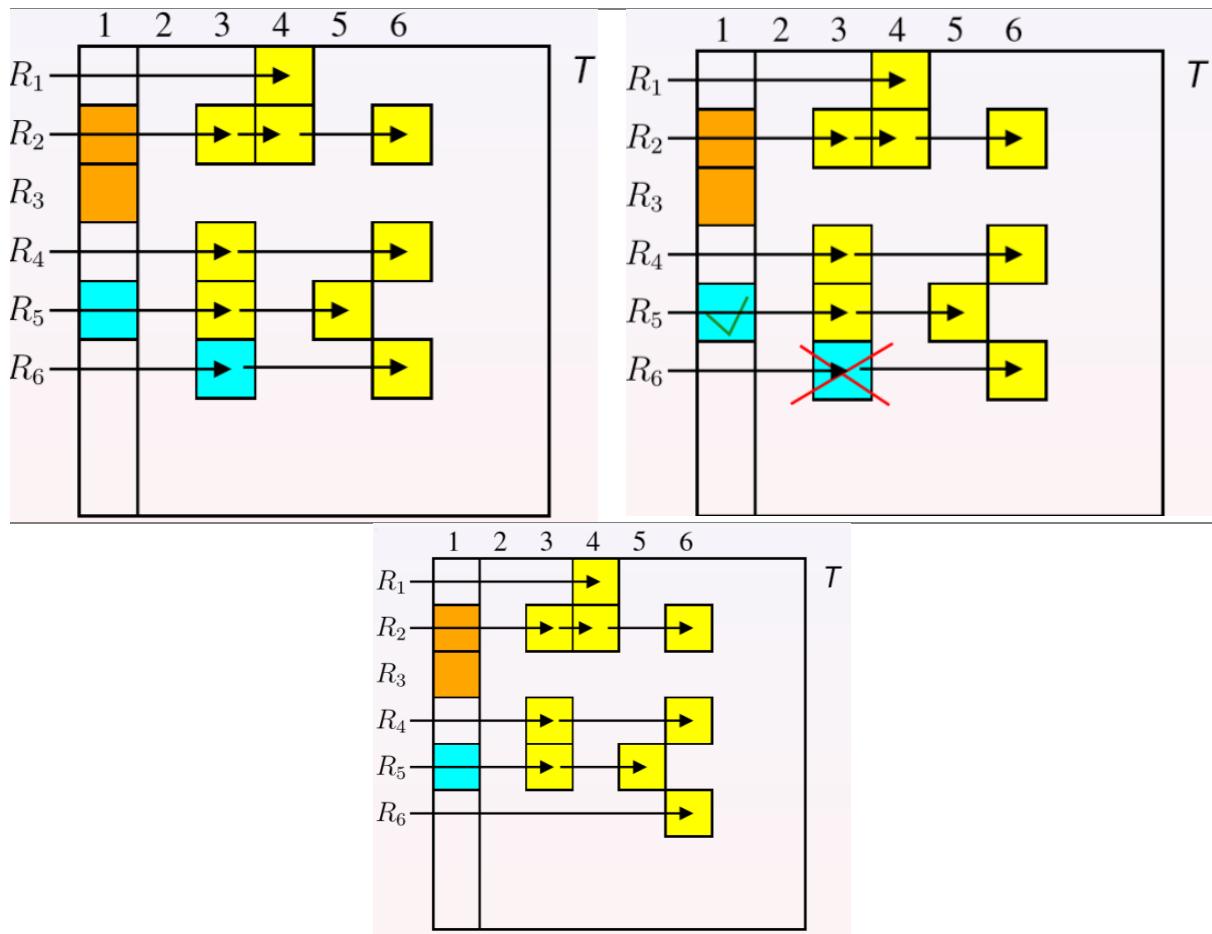
אחרי הצעד הראשון יש לנו את כל המיקומים (y, x) ברשימה כלשהי C_y שהם זוגות עקבאים. המטרה היא למצוא את תת הקבוצה של C_y שבה כל הזוגות הם זוגות עקבאים.

נשמר רשימה $\{R_i\}_{i=1}^{(n-m+1)}$ של הזוגות השורדים בכל שורה. מועמדים אלו הם זוגות עקבאים.

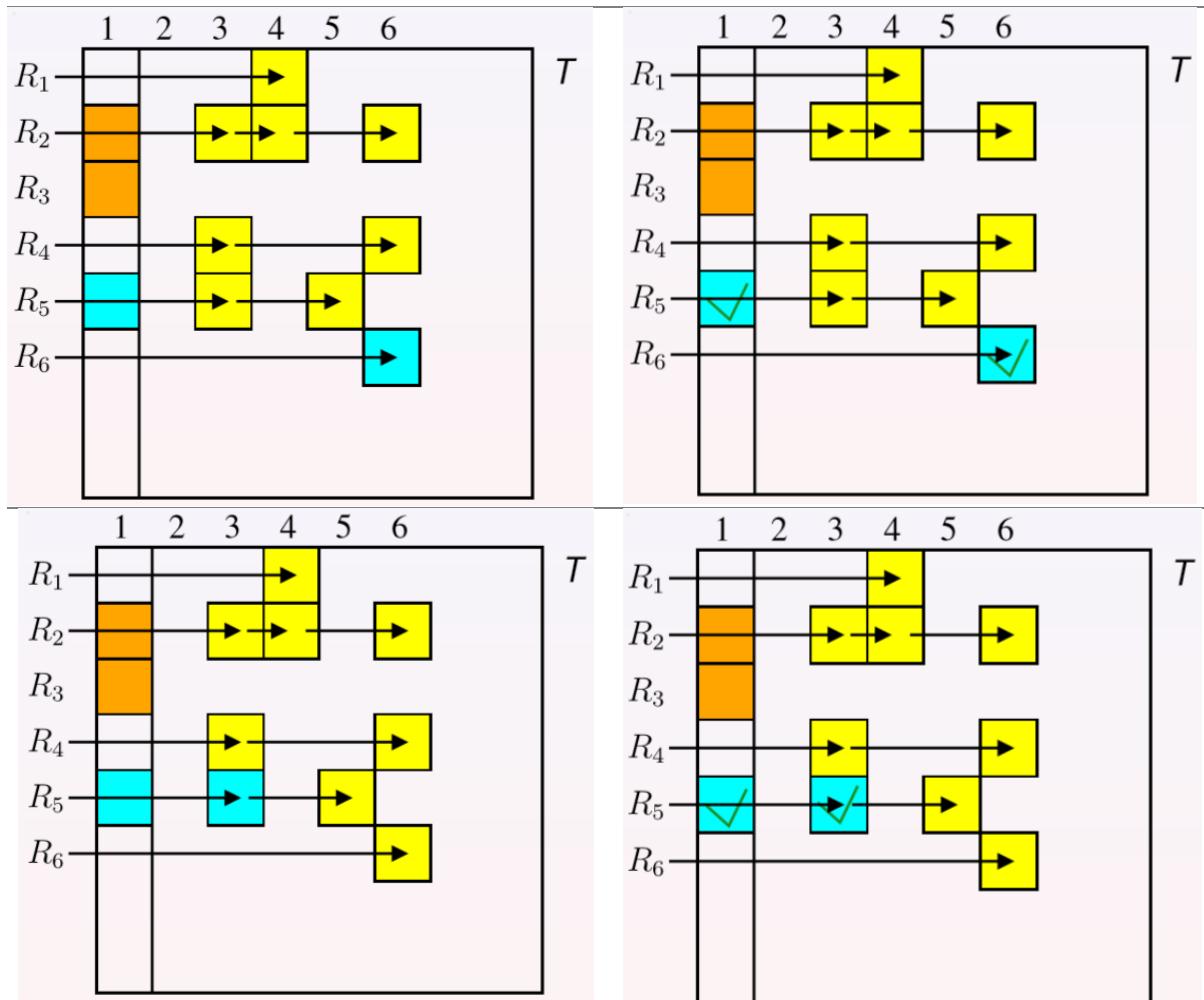
מעבר על הרשימות C_1, \dots, C_{n-m+1} מימין לשמאל. מעבר ראשון על עמודה חדשה מלמטה למעלה.



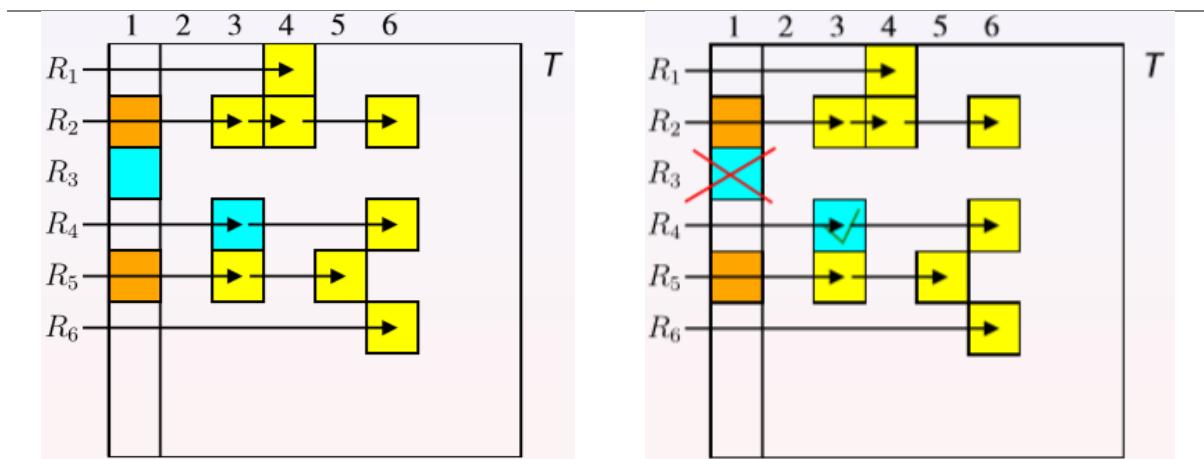
נשווה כל מועמד בשורה חדשה למועמד מתחתיו או משמאלו ובכע *duel*.



אלגוריתמים למחוזות / סיכום מאות אורי שביט



מוצע בעומדה חדשה שרד לא נכנס עוד לרשימה R_i , כיוון שעקביות המועמדים מעליו לא נבדקה עוד.



האלגוריתם

- $R_1 = \emptyset, \dots, R_{n-m+1} = \emptyset$
 - נעבור על הרשימות C_{n-m+1}, \dots, C_1 מימין לשמאל.
 - נעבור על C_y בשני שלבים. שלב ראשון:

$$x' = n \cdot 1$$

2. עבור כל $(x, y) \in C_y$ מלמטה מעלה בצע:

3. כל עוד $x' \geq x$:

כל עוד $0 \neq R_x \cup R_y$ לא עקי עם $head(R_x \cup R_y)$ בצעו. 4.

. $head(R_{x'})$) | (x, y) | duel ב' | .5

$.head(R_{x'})$ הופיע **אך** מחק את $head(R_{x'})$ **ונ** .6

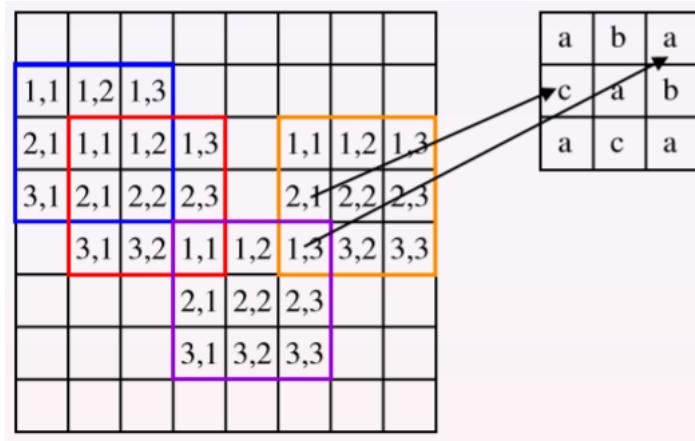
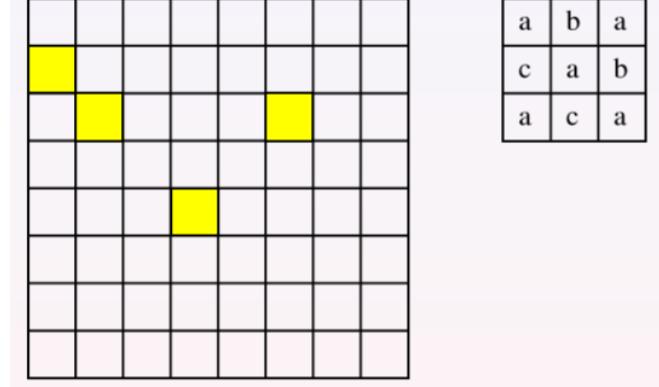
אם (x, y) הפסיד צא.

. $x' = x$ הפסיד את צא אחרת 1 – y .8

בשלב 2 משווים את כל $C_y \in \{x, y\}$ עם האלמנטים של R_1, \dots, R_{x-1} .

מוסיפים את האלמנטים השורדים מ- C_γ ל-

Candidate verification



אלגוריתמים למחוזות / סיכום מאות אורי שביט

1. עבור כל מיקום (y, x) בטקסט שבתוֹר מועמד, נבחר מיקום $(', y')$ בדףו וنبזק אם $T[x, y] = P[x', y']$

2. נדוח על מועמדים עבור כל הרשאות שהן 'True'

T	T	T			
T	T	T	T	T	F
T	T	T	T	F	T
T	F	T	T	T	T
	T	T	T		
T	T	T			

a	b	a
c	a	b
a	c	a

T	T	T					
T	T	T	T	T	F	F	
T	T	T	T	T	F	T	T
T	F	T	T	T	T	T	T
	T	F	T	T	T	T	T
T	T	T					

Step 1

1. לכל מועמד (y, x) נסמן מיקום (x, y) ע"י $(1,1)$.

2. סרוק כל שורה משמאלי לימין. אם $T[x, y - 1]$ לא מסומן ו- $T[x, y]$ מסומן ע"י (j, i) כאשר $m < j$, אז סמן $T[x, y]$ with $(i, j + 1)$

3. סרוק כל עמודה מלמعلיה למטה. אם $T[x - 1, y]$ לא מסומן ו- $T[x, y]$ מסומן ע"י (i, j) כאשר $m < i$ אז

mark $T[x, y]$ with $(i + 1, j)$

1,1							
	1,1						
		1,1					
			1,1				
				1,1			
					1,1		
						1,1	

1,1	1,2						
	1,1						
		1,1					
			1,1				
				1,1			
					1,1		
						1,1	

1,1	1,2	1,3					
	1,1						
		1,1					
			1,1				
				1,1			
					1,1		
						1,1	

1,1	1,2	1,3					
	1,1	1,2	1,3				
		1,1	1,2	1,3			
			1,1	1,2	1,3		
				1,1	1,2	1,3	
					1,1	1,2	1,3
						1,1	1,2

1,1	1,2	1,3	
2,1	1,1	1,2	1,3

1,1	1,2	1,3	
2,1	1,1	1,2	1,3
3,1			

1,1	1,2	1,3	
2,1	1,1	1,2	1,3
3,1	2,1	2,2	2,3

1,1	1,2	1,3	
2,1	1,1	1,2	1,3
3,1	2,1	2,2	2,3

T	T	T	
T	T	T	T
T	T	T	T

T	F	T	T
T	F	T	T
T	T	T	T

T	T	T	
T	T	T	
T	T	T	

T	T	T	
T	T	T	
T	T	T	

Step 2

1. עבור כל pari התאמה (y, x) נסמן את המיקום (y, x) ב $(1,1)$.
2. סמן כל שורה מימין לשמאלי. אם $[y, x] T [y, x+1]$ (i, j) לא מסומן ו $[y, x+1] T [y, x]$ ($i, j < j$) אז סמן $T[x, y]$ with $(i, j+1)$
3. סרוק כל עמודה מלמטה למעלה. אם $[x, y] T [x+1, y]$ (i, j) לא מסומן ו $[x+1, y] T [x, y]$ ($i < m, i < m$) אז סמן $T[x, y]$ with $(i+1, j)$
4. הדפס את המועמדים הלא מסומנים.

סיבוכיות זמן:

1. טבלת העדים יכולה להיבנות ב $O(m^2)$
2. ה due לוקח $O(n^2)$.
3. שלב ההוידוא לוקח $O(n^2)$

סיבוכיות הזמן הכלולית $O(n^2)$

שלבים של שלב 2:

T	T	T						
T	T	T	T		T	F	F	
T	T	T	T		F	T	T	
T	F	T	T	T	T	T	T	
	T	T	T					
	T	T	T					
	T	T	T					

Two-Dimensional Parameterized Matching

Parameterized matching

הגדרה: עבור 2 מחרוזות בגודל $n \times n$ T אם יש פונקציית הסתגלות($P - matches S$) אם $f: \Sigma \rightarrow \Sigma$ מחרוזת בגודל $n \times n$ T אם i, j יושב בזיהוי $T[i, j] = f(S[i, j])$. דוגמא:

S	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>a</td><td>b</td></tr> <tr><td>c</td><td>a</td></tr> </table>	a	b	c	a	T	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>b</td><td>c</td></tr> <tr><td>d</td><td>b</td></tr> </table>	b	c	d	b
a	b										
c	a										
b	c										
d	b										

$$f(a) = b, f(b) = c, f(c) = d$$

הגדרת הבעה

קלט: מחרוזת P בגודל $m \times m$ ומחרוזת T בגודל $n \times n$.

פלט: כל המיקומים (x, y) ב T כך ש $P - matches P[x..x + m - 1, y..y + m - 1]$ עבור $P - matches P$.

P	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>a</td><td>b</td></tr> <tr><td>c</td><td>a</td></tr> </table>	a	b	c	a	T	<table border="1" style="border-collapse: collapse; width: 100px; height: 100px;"> <tr><td>b</td><td>a</td><td>b</td><td>b</td></tr> <tr><td>d</td><td>b</td><td>a</td><td>d</td></tr> <tr><td>b</td><td>a</td><td>c</td><td>a</td></tr> <tr><td>a</td><td>a</td><td>a</td><td>a</td></tr> </table>	b	a	b	b	d	b	a	d	b	a	c	a	a	a	a	a
a	b																						
c	a																						
b	a	b	b																				
d	b	a	d																				
b	a	c	a																				
a	a	a	a																				

In the example above, (1, 1) and (2, 3) are p-matches.

Function matching

הגדרה: עבור 2 מחרוזות בגודל $n \times n$ T אם יש פונקציה $f: \Sigma \rightarrow \Sigma$ כך ש $f(S) - matches T$ אם i, j יושב בזיהוי $T[i, j] = f(S[i, j])$. דוגמא:

S	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>a</td><td>b</td></tr> <tr><td>c</td><td>a</td></tr> </table>	a	b	c	a	T	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>b</td><td>b</td></tr> <tr><td>d</td><td>b</td></tr> </table>	b	b	d	b
a	b										
c	a										
b	b										
d	b										

עובדת: T אם $T - matches S$ אם T מושב בזיהוי T מושב בזיהוי S ומספר התווים T שווה למספר התווים S . T ב- $distinct$.

דוגמה

The number of distinct characters in S is 3.
The number of distinct characters in T is 2.

S	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>a</td><td>b</td></tr> <tr><td>c</td><td>a</td></tr> </table>	a	b	c	a	T	<table border="1" style="border-collapse: collapse; width: 50px; height: 50px;"> <tr><td>b</td><td>b</td></tr> <tr><td>d</td><td>b</td></tr> </table>	b	b	d	b
a	b										
c	a										
b	b										
d	b										

Algorithm outline

1. ניצור רשימה של מועמדים $(m \times m \text{ substrings of } T)$
- i. נתחל על קבוצה L של כל המיקומים (y, x) .
- ii. נבצע *duels* כדי להדיח את כל המיקומים מ- L עד אשר כל המיקומים הם עקבים בזוגות.
- iii. פודא את המועמדים:

- . מחק את כל המועמדים שהם לא $f - \text{match}$
- ii. מחק את כל המועמדים שהם לא $P - \text{match}$ ע"י חישוב מספר התווים *distinct* בכל מועמד.

Witness table

ניקח 2 עותקים של P ונדיז את העותק השני ע"י הuzzת השורות למטה, $0 \leq b \leq a$ עמודות ימינה. (יכול להיות > 0)
 p -match אם הסמלים חופפים באזורי *match shift* a, b

$a = 2$ $b = 0$	a	b	a	W None	-2 2
	b	a	a		
	c	b	a		
	b	a	a		
	c	b	c		
	c	b	c		

עבור הuzzת אי התאמה a, b עד witness ($x, y), (x', y')$ הוא זוג של מיקומים ($', y'$) כאשר אונ:

$$P[x, y] = P[x', y'] \& P[x + a, y + b] \neq P[x' + a, y' + b] .1$$

$$P[x, y] \neq P[x', y'] \& P[x + a, y + b] = P[x' + a, y' + b] .2$$

$a = 1$ $b = 0$	a	b	a	W None	-2 2
	b	a	a		
	c	b	a		
	b	a	a		
	c	b	c		
	c	b	c		

$a = 1$ $b = 1$	a	b	a	W None	-2 2
	b	a	a		
	c	b	a		
	b	a	a		
	c	b	c		
	c	b	c		

Verifying candidates

חולק את T ל $2m \times 2m$ חלקים (עם חפיפה של $1-m$ שורות או עמודות בין החלקים) וטיפול בכל חלק בנפרד.
כלומר, אנו מניחים שמעכשוו $m=2n$.

P	a	b		T	b	a	b	b	b	b	b	b
	c	a		d	b	a	d	d	d	d	d	
				b	a	c	a	a	a	a	a	
				a	a	c	a	a	a	a	a	
				a	a	d	a	a	b	b		
				a	a	b	d	d	a	a		
				a	a	b	d	d	a	a		

T	b	a	b	b	b	b	b	b
	d	b	a	d	d	d	d	d
	b	a	c	a	a	a	a	a
	a	a	c	a	a	a	a	a
	a	a	d	a	a	b	b	
	a	a	b	d	d	a	a	
	a	a	b	d	d	a	a	
	a	a	b	d	d	a	a	

T	b	a	b	b	b	b	b	b
	d	b	a	d	d	d	d	d
	b	a	c	a	a	a	a	a
	a	a	c	a	a	a	a	a
	a	a	d	a	a	b	b	
	a	a	b	d	d	a	a	
	a	a	b	d	d	a	a	
	a	a	b	d	d	a	a	

התאים בצעב טורקייז ב- T הם מועמדים כאשר אנו רוצים לוודא אותם. אנו צריכים לוודא אם יש *f-match*

P	a	a	b	b								
	b	b	c	c								
	c	c	d	d								
	d	d	e	e								
T	b	c	b	b	b	b	b	b	b	b	b	b
	d	b	a	d	d	d	d	d	d	d	d	d
	b	b	c	c	c	c	c	c	c	c	c	c
	b	a	c	a	c	c	c	c	c	c	c	c
	b	d	d	c	c	a	b	b	b	b	b	b
	d	b	b	d	a	c	a	c	a	c	a	c
	c	c	a	d	d	b	b	b	b	b	b	b
	a	c	b	a	d	b	b	b	b	b	b	b

אלגוריתמים למחוזות / סיכום מאות אורי שביט

נטפל בכל אות של הא"ב בנפרד. עכשו נטפל באותיות מסוג a :

		a					
	a			a			
					a		
						a	
a				a			

בניהם שמו עמד (y, x) , יש לנו זוג של a -ים ב- \mathcal{C} , כאשר התווים המתאימים ב- P שונים (כלומר $[x' + a, y' + b] \neq P[x + a, y + b] = P[x', y'] = a \& P[x + a, y + b] \neq P[x' + a, y' + b]$)

P	a	a	b	b
	b	b	c	c
	c	c	d	d
	d	d	e	e

		a				
		a		a		
					a	
					a	a
		a				
a				a		

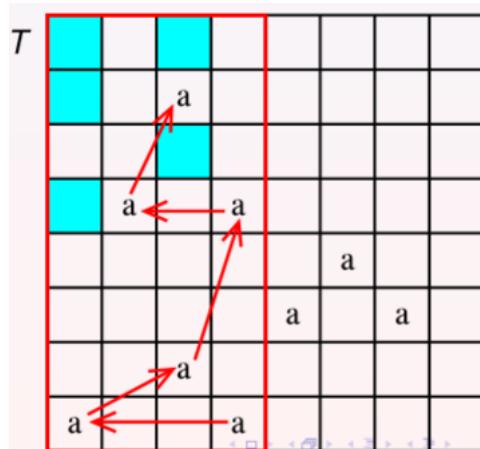
P	a	a	b	b
	b	b	c	c
	c	c	d	d
	d	d	e	e

המודען הוא עקיבי בזוגות, וכך יכולם לקבע עבור כל המועמדים שמקבילים את הזוג הב"ל של ס"ט.

air nbe'au at b'dikot al? anu la ro'zim le'ubor ul kol zogot ha'im hullo?

אלגוריתמים למחוזות / סיכום מאת אורי שביט

ניצור רשימה של כל המ-ים שופיעים ב- m העמודות הראשונות של T . הרשימה ממוינת לפי סריקה מימין לשמאל/ מלמטה למעלה.



לכל זוג של a -ים סומכים ברשימה, נבחר את המועמד העליון ביותר שמכיל את הזוג, ונבדק את התווים המתאימים ב- P . במקרה של אי-שוויון נשולב את כל המועמדים שמכילים את הזוג.

