Predecessor/Successor

תחזוקה של קבוצה $\supseteq U = [u]$, תמיכה בהכנסה/הוצאה/ ושאילתות קודם/עוקב.

van Emde Boas

הכנסה/הוצאה/קודם/עוקב: $\mathcal{O}(\log\log u)\ w.c$, מקום: $\mathcal{O}(u)$. מבנה vEB עבור עולם $[u]$ מכיל $\sqrt{u}$ vEB-ים לעולם $\left[\sqrt{u}\right]$: $S\left[0,\ldots,\sqrt{u}-1\right]$, ואת $S.summary$ מעולם $\left[\sqrt{u}\right]$. בכל vEB שומרים את הערך המינימלי (שלא מוכנס למבנה) והמקסימלי (שמוכנס כרגיל).

x-fast-trie

עדכון: $\mathcal{O}(\log u)\ w.h.p$, שאילתא: $\mathcal{O}(\log\log u)\ w.h.p$, מקום: $\mathcal{O}(n\log u)$ (אך תמיד $\leq 2u$). נשמור את הרישות של כל האיברים בטבלת hash, לכל צומת בעץ min / max בתת-עץ, העלים מהווים רשימה מקושרת. עבור **קודם/עוקב**: חיפוש בינארי על המסלול מהשורש ל-x למציאת הצומת העמוק ביותר שנמצא (מונוטוני: 1^i0^j). אם $x\in left(y)$ אז העוקב הוא $right(y).min$, אחרת $left(y).max$. **למחיקה**, נעיף מהרשימה המקושרת ומה-hash את כל הערכים עד שלאבא יש ילד נוסף. **להכנסה** נוסיף ל-hash את הערכים הנוספים ונעדכן את הרשימה המקושרת.

y-fast-trie

הכנסה/הוצאה/קודם/עוקב: $\mathcal{O}(\log\log u)$ $amortized$ $w.h.p,$ מקום: $\mathcal{O}(n)$. in-direction על x-fast-trie, נשמור את האיברים ב-BST-ים בגודל $\Theta(\log u)$, ולכל BST נשמור נציג ב-x-fast-trie (ולכן בגודל $n/\log u$). עבור **שאילתא** נחפש ב-x-fast-trie ואז בעץ המתאים. עבור **עדכון**, נעדכן את ה-BST המתאים, נפצל אם העץ גדול מ-$2\log u$ ונמזג עם אח אם קטן מ-$1/4\log u$ (יכול לגרור פיצול נוסף). כל פיצול/מיזוג גורר הוספה/הוצאה מה-x-fast-trie של הנציג, עולה $\mathcal{O}(\log u)$ וקורה אחת ל-$\Omega(\log u)$ עדכונים.

Wilber's Lower Bound

נגדיר בן מועדף של צומת y להיות הבן האחרון של y שניגשו אליו. מספר שינויי הבנים המועדפים מהווה חסם תחתון לעלות הכוללת **לכל** BST. $z(y)$, נק' המעבר של y הוא הצומת הגבוה ביותר בעץ הדינאמי T שהמסלול מהשורש ל-$z(y)$ מכיל צמתים משני תתי-העצים של y בעץ הסטטי P. תכונות: $z(y)$ יחיד ומוגדר היטב, ייחודי לכל y,

$z(y)=\text{depper}\left\{\text{LCA}[i,y],\text{LCA}[y+1,j]\right\}$

כאשר תת-העץ של y ב-P מכיל את $i,\ldots,j$. z ייחודי לכל y (חח"ע), ומשתנה רק כאשר מחפשים בתת-העץ של y. על כל שני שינויי בן מועדף של y בהכרח עברנו ב-$z(y)$, וזה החסם התחתון לכל עץ.

Splay Trees

פונקציית הפוטנציאל היא $\sum_{v\in T}\log(s(v))$. נעלה את אשר חיפשו לשורש ע"י zig-zig ו-zig-zag. עלות לשיעורין היא לכל היותר $3\cdot\left(r'(x)-r(x)\right)$, וכך העלות של splay היא $\mathcal{O}(\log n)$ לשיעורין (למעשה $\log n-\log s(x)$). splay הוא $\mathcal{O}(1)$-תחרותי עם static OPT, ואולי גם עם dynamic (השערה). **Working Set Theorem**: אם ניגשנו ל-x לפני t גישות, עלות החיפוש תהיה $\mathcal{O}(\log t)$. **Static Finger Theorem**: אם מתחילים מצומת מסוים f, זמן החיפוש של x יעלה לשיעורין $\mathcal{O}(\log\text{dist}(f,x))$. **Scanning Theorem**: ללא תלות בעץ ההתחלתי, גישה לאיברים $1,\ldots,n$ (בסדר הזה) תעלה $\mathcal{O}(n)$.

Hashing

Universal Hashing

קבוצה $\mathcal{H}$ של פונקציות hash היא אוניברסלית אם

$$\forall x\neq y: Pr_{h\in\mathcal{H}}(h(x)=h(y))=\frac{\mathcal{O}(1)}{m}$$

, כאשר m הוא גודל הטבלה. עבור $m=\Theta(n)$ נקבל שתוחלת אורך השרשרת היא $\mathcal{O}(1)$. למשל זו אוניברסלית:

$$\mathcal{H}_{p,m}=\{(ax+b)\mod p\mod m\ \mid a\in(0,p),b\in[0,p)\}$$

זמן בנייה $\mathcal{O}(n)$ w.c ו-$\mathcal{O}(1)$ זמן לשאילתא בתוחלת. נוכל להגיע ל-$\mathcal{O}(n)$ בנייה בתוחלת ו-$\mathcal{O}(1)$ w.c ע"י הגרלת פונקציות עד שהשרשרת הארוכה ביותר היא לכל היותר $2\times$ תוחלת אורך השרשרת (עובד מאחר וההתפלגות גיאומטרית).

FKS (Perfect Static Hashing)

מקום: $\mathcal{O}(n)$, שאילתא: $\mathcal{O}(1)\ w.c$. פתרון ראשון: $m=\Theta\left(n^2\right)$, ניתן למצוא ב-$\mathcal{O}(1)$ בתוחלת פונקציה ללא התנגשויות, סה"כ $\mathcal{O}\left(n^2\right)$ בתוחלת. פתרון שני: $m=n$ וכך נוכל להגיע ל-$\mathcal{O}(n)$ התנגשויות, לכן $\sum n_i^2=n$ והשרשרת הארוכה ביותר באורך $\sqrt{n}$. נפעיל פתרון שני, ועל השרשראות שקיבלנו נפעיל פתרון ראשון. עלות בנייה היא בתוחלת $\left(\substack{\mathcal{O}(n)+\sum\\=\mathcal{O}(n)}\right)$.

Tree Decompositions

Centroid Decomposition

לכל עץ יש centroid, שאם ננתקו מהעץ נקבל שכל אחד מהרכיבים בגודל $\leq n/2$. הכללה של אמצע המערך בחיפוש בינארי על עצים. ניתן למצוא פירוק centroid לעץ (ורקורסיבית לרכיבים שנותרו) ב-$\mathcal{O}(n\log n)$. נבנה עץ החלטה לפי רמות ה-centroids: centroid העץ המלא בשורש, והבנים שלו הם centroids-של הרכיבים, וכך רקורסיבית. עומק העץ הוא $\log n$: הרכיבים קטנים לפחות פי 2 בכל פעם.

Micro-Macro Decomposition

ניתן לפרק כל עץ עם דרגה $3\leq$ ל-$n/\log n$ עצי micro זרים, כל אחד בגודל $\log n$, כך שלכל עץ יש לכל היותר 2 צמתי גבול (יש להם שכנים מעצי micro אחרים). ניתן להפוך כל עץ לעץ (עם אותו סדר גודל של צמתים) עם דרגה $3\leq$, וכך מתבטלת ההנחה ההתחלתית.

ART Decomposition

מפרקים את העץ לעצים תחתונים ועץ עליון יחיד. העצים התחתונים יושרשו בצמתים הגבוהים ביותר עם $x\leq$ עלים בתת-עץ. מכאן, בעץ העליון יהיו $\frac{n}{x}\leq$ עלים, נפרק אותו באופן רקורסיבי.

Link-Cut Trees

$cut(v), link(v,w), findroot(v)$, עלות כל פעולה היא $\mathcal{O}(\log n)$ לשיעורין. בשונה מ-tango ייתכן שלצומת אין בן מועדף כלל. נפרק למסלולים מועדפים לפי הגדרה זו ונשמור כל מסלול מועדף בעץ splay. הפעולה הבסיסית, $access(v)$, הופכת את המסלול מהשורש ל-v למועדף, ואז עושה splay ל-v.

1. splay ל-v בעץ המייצג שלו. **2.** חותכים את המסלול אחרי v: $right(v)=none$, $path-parent(right(v))=v$. **3.** מחברים למסלול שמעל v, כל עוד v לא השורש: **a.** $w=path-parent(v)$; **b.** $splay(w)$; **c.** $path-parent(right(w))=w$; **d.** $right(w)=v$; **e.** $path-parent(v)=none$, ו-**f.** $v=w$. **4.** $splay(v)$.

עבור $findroot(v)$, נבצע $access(v)$, נמצא את המינימלי ב-splay ונבצע splay למינימלי. עבור $link(v,w)$, בצע $access(v)$, אז $access(w)$ ואז חבר את w כבן שמאלי של v ב-splay. עבור $cut(v)$, נבצע $access(v)$ ואז ננתק את v מבנו השמאלי **בעץ המייצג**.

ניתוח $access$: פונקציית הפוטנציאל היא $\sum_{v\in T}\log w(v)$, כאשר w הוא גודל תת-העץ וכל ה-w-של הצמתים שנתלים על הצומת. באופן שקול לפוטנציאל של splay, הסכום הטלסקופי מסתכם ב-$\mathcal{O}(\log n)$ משוערך. בנוסף, מספר שינויי בן מועדף הוא $\mathcal{O}(\log n)$ משוערך: נובע מ-Heavy-Light Decomposition, מספר הקשתות הקלות במסלול מהשורש ל-v הוא לכל היותר $\log n$, ולכן לכל היותר $\log n$ קשתות קלות הופכות למועדפות. מכאן, $\mathcal{O}(\log n)$ קשתות כבדות הופכות ללא-מועדפות ($\pm n$), ולכן יהיו $\mathcal{O}(\log n)$ שינויי העדפות.

שימושי בשאילתות על מסלולים, למשל מינימום במסלול מהשורש: הוספת שדה מינימום בתת-עץ (למשקלים, לא מפתחות בעץ) בעצי ה-splay ותחזוקה שלו תוך כדי פיצולים/מיזוגים. כך לאחר $access$ עץ ה-splay של השורש מכיל את המסלול מהשורש ל-v, ונוכל לגשת לשדה הנוסף שלו לקבלת המענה לשאילתא. ניתן לעדכן תוך כדי ה-$access$ וה-splay-ים השונים בחינם.

Euler-Tour Trees

$cut(v), link(v,w), findroot(v)$, עלות כל פעולה היא $\mathcal{O}(\log n)\ w.c$.

מסיירים על העץ ומבקרים בכל צומת פעמיים: לפני ואחרי תת-העץ שלו. נשמור BST שיכיל את זמני הכניסה והיציאה של הצמתים, ומצביעים דו-כיווניים מכל צומת מייצג לצומת המתאים שלו בעץ המקורי.

עבור $findroot(v)$ נחפש את המינימום ב-BST. עבור $cut(v)$, נפצל לפי זמני הכניסה והיציאה ונחלץ את תת-העץ של v, ואז נמזג את עצי השאריות. עבור $link(v,w)$, נפצל את העץ של w ונכניס את העץ של v כבן של w.

ניתן להפוך ל-$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ לשאילתא ו-$\mathcal{O}\left(\frac{\log^2 n}{\log\log n}\right)$ לעדכון ע"י branch factor $\log n$. **שימושי** בשאילתות על תתי-עצים: תחזוקה בעצים המייצגים של השדה הרצוי, ופיצול לקבלת העץ המייצג של תת-העץ של צומת מסוים.

Tango Trees

עלות חיפוש: $W\times\log\log n$, ולכן הוא $\mathcal{O}(\log\log n)$-תחרותי. מתחילים מעץ טורבו, נפרק למסלולים מועדפים (באורך $\leq\log n$) ונשמור כל אחד ב-BST. נשמור מצביעים בין עצים מייצגים סמוכים, המפתח של העץ הוא הערך האמיתי. בנוסף, בכל צומת בעץ מייצג יהיה שמור העומק, ועומק מינימלי/מקסימלי בתת-עץ שלו. כל חיפוש בעץ מייצג עולה $\mathcal{O}(\log\log n)$, מספר העצים בהם נחפש הוא מספר שינויי בנים מועדפים. לאחר החיפוש יש לעדכן את הבנים והמסלולים המועדפים: מפצלים מהעץ המייצג את כל הצמתים עם עומק גדול מספיק (זנב המסלול המועדף) ומאחדים עם שאר המסלול המועדף. הצמתים שיש לנתק מהווים אינטרוול של מפתחות: נמצא את קצוותיו ע"י מציאת הצמתים הגדולים והקטנים ביותר שבתת-העץ שלהם יש עומק $d<$, ונבצע פיצולים ומיזוגים לבידוד האינטרוול משאר העץ.

Dynamic Connectivity

Holm, de Lichtenberg and Thorup

שאילתת קשירות ב-$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$, הוספה ומחיקה של קשתות ב-$\mathcal{O}\left(\log^2 n\right)$; לשיעורין. נתחזק $\mathcal{O}(\log n)$ יערות פורשים, לכל קשת יש רמה, מאותחל ל-$\log n$ ויורדת עד ל-0. G_i הוא תת-הגרף שמכיל את כל הקשתות ברמה $\le i$, ו-F_i יער פורש שלו, מיוצג באמצעות ET tree, $F_{\log n}$ עם branching factor $\log n$ ולכל השאר 2. <u>שמורות:</u> $F_i \subseteq F_{i+1}$, וגודל כל רכיב קשירות ב-G_i $\le 2^i$.

בהכנסת קשת נגדיר את הרמה שלה $\log n$ (ונשלם $\mathcal{O}\left(\log^2 n\right)$ קופונים). לשאילתת קשירות נשתמש ב-$findroot$ ב-$F_{\log n}$. במחיקת קשת (u,v), אם היא לא ב-$F_{\log n}$ פשוט נמחק אותה. אחרת, אם הרמה שלה i נמחק אותה מ-$F_i,\ldots,F_{\log n}$, ונחפש קשת מחליפה בכל אחד (בהכרח ברמה $\ge i$ בשביל השמורות), נחפש החל מ-G_i ונעלה עד שנמצא. הקשת שנמחקה מפרקת את F_i ל-T_v, T_u, בה"כ $|T_v| < |T_u|$ ולכן $|T_v| \le 2^{i-1}$.

נעבור על כל קשת ברמה i שיוצאת מ-T_v (מתבצע ביעילות ע"י שדה "קשת ברמה i בתת-עץ" ב-ET tree). נשלם על כל קשת ע"י הורדה ברמה, וכאשר מצאנו מחליפה נוסיף אותה לעצים הפורשים, החל מזה של רמת הקשת המחליפה. לאחר מכן נוריד את כל הקשתות מרמה i ב-T_v ל-$i-1$, ונוסיף את T_v כעץ ב-F_{i-1} (לשמר הכלה). כדי למצוא את הקטן מבין שני הרכיבים נתחזק שדה תת-עץ ב-ET trees. עלות ההכנסה ל-$F_i,\ldots,F_{\log n}$ היא $\mathcal{O}\left(\log^2 n + \frac{\log^2 n}{\log\log n}\right) = \mathcal{O}\left(\log^2 n\right)$, ועל כל קשת מבוזבזת שילמנו $\mathcal{O}(1)$ מאחר והורדנו בדרגה (לא נעבור את הקופונים ששילמנו בהכנסה מאחר והעץ קטן לפחות $2\times$ בכל הורדת רמה).

Decremental Connectivity

מחיקת קשתות ושאילתת קשירות על עצים. $\mathcal{O}(1)$ w.c לשאילתא ו-$\mathcal{O}(1)$ amort. לעדכון. <u>פתרון ראשון:</u> נשמור מספר לכל צומת, צמתים באותו רכיב קשירות $\iff$ המספר שלהם שווה. למחיקת קשת נעבור בהרצה מבוקרת על שני העצים ונעצור כשהקטן הסתיים. נגדיל את המספר של כל צומת ברכיב הקטן ונמחק את הקשת. מכאן, $\mathcal{O}(1)$ לשאילתא ו-$\mathcal{O}(\log n)$ למחיקה amort. (העץ קטן לפחות פי 2 בכל שינוי).

שיפור: נשתמש ב-Micro-Macro decomp.: נפרק לעצי micro בגודל $\mathcal{O}(x)$, ועץ macro בגודל n/x. נשתמש בפתרון ראשון עבור עץ המאקרו, וכך מחיקת קשת במאקרו עולה $\mathcal{O}(1)$ לשיעורין: בסה"כ לכל הפעולות $\mathcal{O}\left(\frac{n}{\log n}\log\frac{n}{\log n}\right) = \mathcal{O}(n)$.

נשריש את העצי המיקרו ונמספר את הקשתות $1,\ldots,\log n$ לפי העומק. לכל צומת v נשמור מילת מחשב W_v שדלוקים בה הביטים של הקשתות במסלול מהשורש ל-v. בנוסף, מילה גלובלית X עם ביטים דלוקים לכל קשת שלא נמחקה. נבצע micro-connectivity ע"י $(W_u \mathrm{XOR} W_v)\ \mathrm{AND}\neg X$. עבור שאילתא (מלאה), נבצע 4 שאילתות micro/macro-conn., וכך $\mathcal{O}(1)$ זמן. למחיקת קשת, נמחק מעץ-המיקרו, ואם נשברת הקשירות של צמתי הגבול (נבדוק ע"י שאילתא) ננתק גם במאקרו.

Move-To-Front

dynamic OPT יכול להזיז את האיבר שחיפש אחורה ברשימה בחינם (free swaps) ולהחליף בין כל שני שכנים בעלות 1 (paid swaps). MTF הוא 2-תחרותי ביחס ל-static OPT: $cost_{ij}$ מוגדר להיות מספר הפעמים שעברנו על פני i בחיפוש אחר j. העלות הכוללת של סדרת הפעולות היא $\sum cost_{ij}$. נסמן ב-f_i את תדירות i בסדרת החיפוש. עבור static OPT, אם $f_i \ge f_j$ אז $cost_{ij} = f_j$ ו-$cost_{ji} = 0$. עבור MTF מתקיים $cost_{ij}, cost_{ji} \le f_j$. 2-תחרותי ביחס ל-dynamic OPT: פונק' הפונטציאל היא #אי-הסדרים בין שתי הרשימות, בניתוח נבדיל בין אלו שמוסיפים אי-סדרים ובין אלו שלא.

Strings

Suffix Trees

Compressed Trie שמכיל את כל הסיפות של המחרוזת "$T\$$". סיור in-order מחזיר מיון לקסיקוגרפי. מערך בכל צומת גורר $\mathcal{O}(n|\Sigma|)$ מקום ו-$\mathcal{O}(|P|)$ לשאילתא. BST בכל צומת: $\mathcal{O}(n)$ מקום ו-$\mathcal{O}(|P|\log|\Sigma|)$ לשאילתא.

Compressed Trie: חבר אבות עם בן יחיד לבן שלהם, שומרים בכל צומת את האינדקסים של תת-המחרוזת המיוצגת, ולכן $\mathcal{O}(n)$ מקום. עלות בנייה היא $\mathcal{O}(|T| + sort(\Sigma))$, ולכן בפועל לינארית ב-$T$. ST מוכלל עבור מחרוזות $T_1\$_1\cdots T_n\$_n$, מנתקים תתי-עצים של $\$_i$.

Kangaroo Matching: נבנה ST עבור $P\$T$. בכל פעם נקפוץ ל-mismatch הבא ב-$\mathcal{O}(1)$ עם ה-LCA. עלות $\mathcal{O}(k|T|)$.

Suffix Arrays

מערך ממוין לקסיקוגרפית של הסיפות, יחד עם מערך LCP של עבור סיפות עוקבות במערך ו-RMQ עליו. שקול לעלים של ה-ST משמאל לימין. חיפוש נאיבי $\mathcal{O}(|P|\log|T|)$ עם חיפוש בינארי.

ניתן להגיע ל-$\mathcal{O}(|P| + \log|T|)$: אם כבר הגענו ל-LCP באורך k בין $left$ ו-P, נוכל להשוות את $\mathrm{LCP}(left, mid)$ מול k: אם $< k$ נמשיך שמאלה, אם $> k$ נמשיך ימינה, ואם $= k$ נקרא אות נוספת של P.

SA ↔ ST

בהינתן ST, נעבור ב-in-order על העץ ונכניס את העלים כסיפות. ה-LCP הוא ה-LCA, בסה"כ זמן $\mathcal{O}(|T|)$. **בהינתן SA**, נכניס את הסיפות אחת אחר השנייה ל-ST. נגיע ל-LCA באמצעות ה-LCP, אולי נפצל קשת ונכניס את העלה והקשת החדשים. עלות $\mathcal{O}(|T|)$: ניגע בכל צומת מספר קבוע של פעמים.

Linear Time Construction

תחילה, נמיין את Σ ב-$\mathcal{O}(sort(\Sigma))$. רקורסיבית: **1.** מיין ב-radix-sort ב-$\mathcal{O}(|T|)$ זמן **2.** החלף כל אות במיקום שלה במיון, לשמירה על הסדר **3.** T_i יכיל את הסיפות שמתחילות באינדקס ששקול ל-$i \pmod 3$, קרא רקורסיבית על "T_0T_1" עם $\Sigma' = \Sigma^3$, כלומר כל 3 אותיות מהוות אות אחת, הטקסט באורך $\frac{2n}{3}$ **4.** מיין את סיפות T_2 באמצעות המיון של סיפות מ-T_1 ואות ראשונה **5.** מזג את T_2 באמצעות השוואת 2-1 אותיות ראשונות, ושאר השוואת המחרוזות נתונה מיחס הסדר בין סיפות ב-T_0, T_1 (וה-LCP נתון מהרקורסיה) **6.** באופן דומה נחשב את ה-LCP

סיבוכיות הזמן היא $T(n) = T\left(\frac{2n}{3}\right) + \mathcal{O}(n) \Rightarrow T(n) = \mathcal{O}(n)$

Suffix Trays

שילוב של ST ו-SA. צומת-Σ היא כזו שבתת-העץ שלה יש $\ge |\Sigma|$ עלים ולכל בן שלה יש $\le |\Sigma|$ עלים. צומת Σ-מסועף הוא כזה עם לפחות 2 בנים עם $\ge |\Sigma|$ עלים בתת-עץ. נחתוך תתי-עצים עם פחות מ-$|\Sigma|$ עלים, ונשמור את האינטרוול שתת-העץ מייצג ב-SA. לכל צומת אחר יש בן אחד שהוא צומת-Σ, נוכל לבדוק ב-$\mathcal{O}(1)$ האם האות מתאימה לבן המשמעותי, ואחרת נעבור לאינטרוולים. בחיפוש נגיע עד לצומת-Σ או קטן מכך, ולאחר מכן נקפוץ לחפש באינטרוול של ה-SA (עלות חיפוש באינטרוול I שם היא $|P| + \log|I|$). בסה"כ $\mathcal{O}(|P| + \log\Sigma)$ לשאילתא.

המקום לינארי - ה-SA צורך מקום לינארי, יחד עם $\frac{n}{|\Sigma|} \le$ צמתים Σ-מסועפים: $\mathcal{O}\left(n + \frac{n}{|\Sigma|}\cdot|\Sigma|\right) = \mathcal{O}(n)$.

<u>תזכורת:</u> אתה נמצא **במבחן סוף** בקורס **מבוא לחמרה** עם המרצה **ארז גרליץ**.

Fusion Trees

BST עם שימוש במודל ה-Word-RAM, branching factor של $w^{1/5}$ (גודל מילה w). גובה העץ הוא $\mathcal{O}(\log_w n) = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$. הקושי: מציאת הבן המתאים ב-$\mathcal{O}(1)$.

נתייחס למפתחות בצמתים הפנימיים כמחרוזות בינאריות x_i, ונחשוב עליהם כמסלולים בעץ בינארי בגובה w. יש לכל היותר $w^{1/5}$ ביטים מבדילים, כמספר המפתחות. נסמן אותם ב-b_i. $Sketch(x_i)$ מכילה רק את המשמעותיים, ולכן $Sketch(x_i) \le Sketch(x_{i+1})$.

בהינתן מפתח q, נרצה למצוא i כך ש-$x_i \le q \le x_{i+1}$. **1.** מצא i כך ש-$Sketch(x_i) \le Sketch(q) \le Sketch(x_{i+1})$: **a.** חסר את $(0\circ Sketch(q))^{w^{1/5}}$ מ-$1\circ Sketch(x_0)\circ\cdots\circ 1\circ Sketch\left(x_{1/5}\right)$; **b.** חשב את ה-AND של התוצאה עם $1000\cdots1001$, להשארת ה-carry bits בלבד, ו-**c.** הביט ה-i בתוצאה הוא 1 $\iff$ $Sketch(x_i) \le Sketch(q)$, ולכן נמצא לזה MSB. **2.** $C = \mathrm{LCP}(x_i, q) = \mathrm{MSB}(x_i \mathrm{XOR} q)$ הוא המסלול המשותף מהשורש **3.** q הוא $C1D$ ונרצה $x_j = C0E$, ולכן באותה הצורה כמו קודם נמצא x_j כך ש-$Sketch(x_j) \le Sketch(C011\cdots1) \le Sketch(x_{j+1})$, והוא מקיים $x_j \le q \le x_{j+1}$.

לא ניתן לחשב את $Sketch$ במלואה, ונחשב במקום את $ApproxSketch$. נשמור את b_i באותו הסדר, אך יהיו אפסים ביניהם. $|ApproxSketch| \le w^{4/5}$, כך שבסך הכל יהיה קטן מ-w. נאפס ביטים שאינם b_i עם AND, ונכפיל ב-M לקבלת ה-$ApproxSketch$. ב-$M * x_i$ יש ביטים מהצורה $b_i + m_j$. **1.** $b_i + m_j$ ייחודיים (אין התנגשויות); **2.** $b_i + m_i \le b_{i+1} + m_{i+1}$ (שמירה על הסדר של b_i), ו-**3.** אורך כולל של $\mathcal{O}\left(w^{4/5}\right)$

נמצא את M כך: **1.** אינדוקטיבית, נניח שמצאנו $m_0,\ldots,m_t < r^3$ כך ש-$b_i + m_j$ ייחודיים מודולו r^3; **2.** נרצה $m_{t+1} + b_l \neq m_i + b_j \Rightarrow m_{t+1} \neq m_i - b_l + b_j$ ($\bmod\ r^3$); **3.** בסך הכל $tr^2 < r^3$ אפשרויות, ולכן קיימת בחירה של m_{t+1}; **4.** לכל i: $m_i \leftarrow m_i + i\cdot r^3$; **5.** כך $m_i + b_i$ נמצאים באינטרוול באורך r^3, ושומרים על הסדר, ו-**6.** הבטחנו ייחודיות בבחירה, והאורך הוא $\mathcal{O}(r * r^3) = \mathcal{O}\left(w^{4/5}\right)$.

שילוב של vEB ו-Fusion גורר מיון ב-$\mathcal{O}\left(n\sqrt{\log n}\right)$.

Marked Ancestor

שאילתת LMA ב-$\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$, $mark/unmark$ ב-$\mathcal{O}(\log\log n)$. נשתמש ב-ART decomp. עבור $x = \log n$. עומק הרקורסיה הוא $\frac{\log n}{\log\log n}$. עבור עדכון נעדכן בעץ התחתון, ועבור שאילתא נבצע על כל אחד מהעצים התחתונים שמעל הצומת. עבור עצים תחתונים (בגודל $\log n$), נפרק למסלולים זרים בהם אב לבן יחיד מצטרף למסלול של הבן שלו. $\mathcal{O}(\log n)$ עלים ולכן $\mathcal{O}(\log\log n)$ מסלולים.

לכל מסלול נשמור vEB לשאילתות LMA תוך-מסלוליות, ומילה W_p שדולקים בה הביטים של המסלולים שהם אבות של המסלול הנוכחי. בנוסף, מילה גלובלית X לכל העץ התחתון, בה דלוקים הביטים של מסלולים שיש בהם צמתים marked. עבור שאילתא בעץ תחתון, מסלול ה-LMA יימצא ע"י $\mathrm{LSB}(X \mathrm{AND} W_p)$. לאחר מכן נמצא באמצעות ה-vEB קודם ב-$\mathcal{O}(\log\log)$. סה"כ $\mathcal{O}\left(\frac{\log n}{\log\log n} + \log\log n\right) = \mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ עבור עדכון, נכניס ב-vEB המתאים ונעדכן את X אם ה-vEB היה/הופך לריק.