מבני נתונים – מטלת הגשה 4

א.בעבודה השתמשנו במחלקות הבאות:

LinkedList- מייצגת רשימה דו כיוונית עם מצביעים head, tail וכן שדה length. נעזרנו ברשימה שכזאת להחזקת הPairs המתקבלים לאחר קידוד.

Link-חוליה בודדת ברשימה המקושרת. בעלת Pair כשדה.

Trie-מממשת את מבנה הנתונים Trie. המחלקה מחזיקה מצביע לשורש העץ.

Node-מייצגת צומת בודד בעץ. לכל צומת ישנו מצביע לאבא, ערך value המקשר בינו לבין אביו, מצביע לאביו parent, מערך של ילדים children (כל ילד הוא Node), index המציין את סדר ההכנסה לעץ וכן השדות appear וdepth המייצגות את כמות ההופעות של הvalue בעץ ועומק הצומת, בהתאמה.

בקצרה, לקידוד טקסט input כלשהו השתמשנו במבנה נתונים מסוג Trie למימוש הפסאדו-קוד הנתון (הכנסת צמתים לעץ כך שהענפים בינם לבין שורש העץ מייצגים רישא כלשהי בטקסט) וכן ברשימה מקושרת לשמירת הPairs. כדי לשחזר את הטקסט המקודד נעזרנו ברשימה השמורה ובמערך (בדומה להסבר בגוף העבודה).

ב.עמידה ביעילות הנדרשת:

encode (int[] input)- המרנו את המערך לטקסט (מס' הפעולות כאורך הinput) ואז שלחנו לencode(String input). סך הכל $O(|input|) + O(|input|) = O(|input|)$.

getMaxPrefix(Node v, String text)- פונקציית עזר המחזירה את הצומת שהvalue שלו הוא התו האחרון ברישא של הקלט text. במקרה קיצון הפונקציה תבצע מספר פעולות כגובה העץ, כלומר $O(h)$. נציין שגודל זה קטן ממש מגודל הקלט input (בפונקציית encode), זאת מכיוון שככל שגובה העץ גדל קטנה הרישא המוכנסת כtext ונעשות פחות פעולות למציאת הצומת המתאימה.

encode (String input)- בדומה לפסאדו-קוד הנתון, הפונקציה בודקת אם קיים בעץ רצף השווה לרישא בקלט input ובהתאם לכן מוסיפה זוגות לרשימת הpairs. הפונקציה במקרה קיצון תבצע מספר פעולות כגודל הקלט, כאשר בכל פעולה תקרא getMaxPrefix. הראנו כי $h < |input|$ לכן נקבל $O(|input|) \cdot O(h) = O(|input|)$.

getPairs()- שמרנו את הPairs ברשימה מקושרת ולכן מעבר על הרשימה ייקח בדיוק כמספר הPairs בה. כלומר $O(n) = O(number\ of\ pairs)$.

reconstruct()- השתמשנו בפונקציה getPairs[] לקבלת מערך ואז שלחנו אותו לreconstruct(Pair[]). גודל המערך שווה למספר החוליות ברשימה לכן סך הכל $O(|pairs|) + O(|pairs|) = O(|pairs|)$.

reconstruct(Pair[])- בדומה לקוד בגוף העבודה, עברנו על המערך ולכל pair שמרנו רישא מתאימה במערך ושרשור מתאים במחרוזת. מספר הפעולות כאורך מערך הקלט (כלומר מספר הזוגות שקודדו) לכן $O(|pairs|)$.

add(String input)- שמרנו את הצומת האחרון אליו רצינו לחבר צומת חדש במהלך הencode כשדה במחלקה LempleZiv. בשל כך, במידה והתו הראשון של הקלט שקיבלנו יוצר רישא מתאימה עם סוף הinput הקודם, נוכל לחבר צומת מתאים ב$O(1)$ פעולות. לאחר מכן נמשיך בקידוד באופן דומה לפונקציית encode, כלומר $O(1) + O(|input|) = O(|input|)$.

maxCodeLength()- במהלך הקידוד תחזקנו שדה depth לכל Node וכן עדכנו מצביע maxDepth במחלקה LempleZiv המצביע על הצומת עם העומק המקסימלי בעץ. עומק הצומת שווה בדיוק לאורך הרישא אותה מייצגים הענפים לבינה לבין השורש, לכן יכולנו פשוט להחזיר את עומק הmaxDepth ב$O(1)$.

maxCode()- לכל צומת ישנו מצביע parent לאבא שלו לכן יכולנו לעלות במעלה העץ מהmaxDepth עד לשורש (כל עוד parent != null) ולהחזיר את ערכי הvalue של הצמתים עליהם עברנו בסדר הפוך. במקרה קיצון נעלה כגובה העץ, כלומר $O(h)$.

randomBinarySeries(int length, double ratio)- לכל תא במערך הפלט ביצענו הטלה כלשהי, לפיה הוספנו ערך מתאים בתא. מספר הפעולות כגודל המערך כלומר $O(length)$.

codes(String prefix)- במהלך השימוש בפונקציית העזר getMaxPrefix תחזקנו שדה appear לכל Node המציין את כמות הרישות בהן מופיע הvalue שלו. בשל כך יכולנו להתחיל בחיפוש משורש העץ ולרדת עד שאנו מגיעים לצומת שהvalue שלה שווה לתו האחרון בprefix הקלט. מכיוון שהילדים של כל צומת שמורים במערך, יש לנו בכל שלב גישה ישירה לצומת בעלת הvalue המתאים. אנו יורדים במורד העץ כמספר התווים בprefix, כלומר $O(|prefix|)$.

ג. להלן התוצאות:

Length	Ratio	Average number of pairs	Minimum number of pairs	Maximum number of pairs
200	0.5	48	46	50
200	0.7	45	42	49
200	0.9	32	26	38
200000	0.5	16292	16281	16305
200000	0.7	14673	14626	14728
200000	0.9	8763	8683	8838

ד.a+b.

הטקסט בעברית: נלקח מהספר "צבעים ברוח" מאת נאוה דיקסטרה.

מס' תווים בטקסט : 10158

מס' הזוגות (אורך הקידוד): 2903

אורך הרישא הארוכה ביותר בטקסט הוא: 8

הרישא הארוכה ביותר בטקסט היא: שרי הוש (כולל רווח לפני ש' ובין י' ל ה')

אחוז הכיווץ: 28.5%

הטקסט באנגלית: " AN ALUMINUM CAN A DAY"

by Carl H. Mitchell.

מס' תווים בטקסט : 10156

מס' הזוגות (אורך הקידוד): 2870

אורך הרישא הארוכה ביותר בטקסט הוא: 9

הרישא הארוכה ביותר בטקסט היא: `Jack's v` (כולל רווח לפני J ובין s ל v)

אחוז הכיווץ: 28.25%

c.תחילה נשים לב שאורך הקידוד בעברית גדול מאורך הקידוד באנגלית, כלומר מצאנו רישות ארוכות יותר באנגלית, כלומר יותר מילים באנגלית בנויות על בסיס רצפים מסוימים מאשר בעברית.

הבדלים אלו עשויים לנבוע ממספר גורמים המהווים הבדלים בין השפות: מס' האותיות בכל שפה, אותיות נפוצות בשימוש, ביטויים נפוצים בשימוש.

דניאל מרגלית 204093983

נוי סוג'ז 307933143