תיעוד חיצוני HW2

מטרת התכנית : מקבלת כקלט רשימה של מילים (מילון) ורשימה של תחיליות, התכנית מחזירה את התחילית אשר מתאימה להכי הרבה מילים ואת מספר המילים אליה מתאימה.

התאמה נקבעת באופן הבא:

-אם המילה המילונית היא בעצם העתק של התחילית, בתוספת אותיות לאחר האות האחרונה במילית.

-אם המילה המילונית היא שינוי אות אחת במילית, ולאחר מכן צירוף אותיות נוספות

-אם המילה המילונית היא השמטת אות אחת במילית, ולאחר מכן צירוף אותיות נוספות

-אם המילה המילונית היא התחילית בתוספת אות אחת, ולאחר מכן צירוף אותיות נוספות.

הרעיון מאחורי התכנית :

נבין תחילה את הדברים הבאים :

1. קל מאד לבדוק אם התחילית היא התחילית המדויקת של המילה. לשם כך נדרש לעבור על האותיות המתאימות בתחילית ובמילה ולהשוות ביניהם, אם כולן שוות, מצאנו התאמה.
2. לשם בדיקת התאמה על ידי שינוי אות בתחילת, ניתן פשוט לדלג על אות אחת שאינה מהווה התאמה בתחילית ובמילה ולבדוק האם אחריה (ולפניה) יש

התאמה. אם יש התאמה בכל האותיות חוץ מאחת, זאת האות שנחליף והתחילית אכן תואמת. אם יש יותר מאות אחת שאינה תואמת, נחליף את הראשונה ונמשיך לבדוק, וכאשר נגיע לאות הנוספת, נפסול את התאמת התחילית. עובדה זו מאפשרת לנו פשוט לדלג על האותיות הראשונות בתחילית ובמילון שאינן תואמת.

3. אם התחילית מתאימה למילה, אך יש בתחילית אות נוספת, נוכל לגלות זאת על ידי כך שבמהלך בדיקת האותיות, נדלג על אות אחת בתחילית שאינה מתאימה לאות במילון, הדילוג על האות מייצג את השמטתה.

4. באופן דומה, אם התחילית מתאימה למילה, אך יש בתחילת המילה אות נוספת אשר מונע מההתאמה להיות זהה לחלוטין, נדלג על האות הזאת במילון, והדילוג הזה מייצג את השמטתה.

אם אחת מהבדיקות הללו מגלה לנו שהתחילית מתאימה למילה, אין צורך לבדוק את השאר. באופן דומה, אם אחת מהבדיקות הללו מגלה לנו שהתחילית אינה מתאימה, עדיין יש לבדוק בשאר הדרכים.

בתכנית, נבדוק את התחיליות אחת אחת, כלומר, נקח את התחילית הראשונה, נשווה בינה לבין כל המילים בתכנית, ולאחר מכן "נזרוק" אותה ונעבור לתחילית הבאה.

'מילים' שונות שיש להכיר :

Strings_len- ניתן כקלט, מחזיק את מספר התחיליות.

Dict_len- ניתן כקלט, מחזיק את מספר המילים במילון.

Dict- המילון עצמו. מחזיק את הכתובת לאות הראשונה במילה הראשונה במילון.

Strings- מערך תחיליות, מחזיק את הכתובת לאות הראשונה בתחילית הראשונה.

Max_score- בית אשר יחזיק את מספר ההתאמות המקסימלית.

Max_string- בית אשר יכיל את הכתובת של האות הראשונה של התחילית אשר מתאימה להכי הרבה במילים. במקרה של שיוויון בין תחיליות, ניקח את זאת שהופיעה ראשונה במערך התחיליות.

סטרינגס ודיקט מתקבלים כמערך. בסיומה של כל מילה, תחילית ומילה כאחד, יופיע התו '@' .

רגיסטרים :

R0- מייצג את מספר ההתאמות שיש לתחילית הנבדקת. יעלה באחד בכל פעם שנמצאה התאמה

R1- יחזיק את הכתובת לאות הראשונה במילה (המילונית). כך שבעת המעבר על האותיות, האינדקס הזה יקודם ויחוזר אחורנית בעת הצורך.

R2- בדומה לr1, רק הפעם הרגיסטר מחזיק את הכתובת לאות הראשונה בתחילית.

R3- מספר התחיליות שבדקנו עד כה.

R4- המקום במילה שכעת אנו משווים. כלומר, אם אנו משווים את האות השלישית במילה לאות השלישית במילון, רגיסטר זה יחזיק את הערך 3.

R5- מספר המילים שבדקנו עד כה.

R6,R7- לא נעשה שימוש.

מימוש הרעיון בצירוף ה'מילים' –

בתחילה ניתן לרגיסטרים את הערכים המתאימים בהתאם להגדרות שבחרנו עבורם.

כעת נתחיל את הבדיקה. בשלב הראשון, התכנית תבצע הסתעפות אל התווית check_exactly, אשר משווה בין האותיות התואמות של התחילית ושל המילה. באזור זה נקדם את הרגיסטרים R1,R2 עד שנגיע לאחד משלושה מצבים:

1. התחילית נגמרה, R2 מכיל כתובת אל תו '@'. במצב זה נסתעף לתווית success_check_exactly. שתוסבר בהמשך.
2. המילה הגיעה לסיומה, R1 מכיל כתובת אל בית שמכיל את '@'. נבדוק האם התחילית מסתיימת באות הבאה, שהרי אם כן, ניתן להשמיט את האות הנוספת ולהגיע להתאמה, ואם לא, נסתעף אל failed_check. שיוסבר בהמשך.

3. הגענו אל מקום אשר בו המילה והתחילית אינן זהות. נתרכז במצב זה שכן טיפלנו כבר בשאר המצבים.

נסתעף חזרה לראש התכנית, <u>כאשר אנו יודעים שהגענו לאינדקס בו האותיות שונות בין המילה המילונית לבין התחילית.</u> ולכן נרצה לבדוק את המצבים בהם: באינדקס זה יש להשמיט אות מהתחילית, באינדקס זה יש להשמיט אות מהמילה, ושבאינדקס זה יש להחליף אות במילון, כפי שתואר מקודם.

נבדוק האם מדובר באחד המצבים באופן הבא, נתייחס לכל מצב בקטע קוד שונה, אשר יופיע תחת התוויות הבאות: second_check, third_check ,fourth check

קטע הקוד עצמו יהיה דומה בכל אחת מהתוויות, מכיוון שיש כאן הליך של השוואת אותיות עד למקרה של אי התאמה או סיום התחילית, או כמובן סיום המילה מה שאומר שהתחילית אינה מתאימה (לא ניתן להשמיט אות נוספת במצבים האלה).

ההבדל יהיה, שבבדיקה השנייה (second_check, שנייה לבדיקת ההתאמה הזהה check_exactly שהתבצעה קודם לכן), נקדם את R1 לכתובת האות הבאה במילה (המילונית), מה שמייצג את המצב בו אות אחת מושמטת מהמילה. בבדיקה השנייה נקדם את R2 לכתובת האות הבאה בתחילית, מה שמייצג את המצב בו אות אחת מושמטת מהתחילית. ובמצב השלישי, נקדם את שניהם, על מנת לייצג את המצב בו החלפנו אות אחת בתחילית באחרת.

יש לשים לב: חזרנו מן check_exactly כאשר באינדקס הנבדק במילה והתחילית האותיות אינן תואמות, עד לאינדקס זה לא נותר מה לבדוק יותר כי מבחינתו המילה והתחילית זהות עד כאן בין אם הזהות מכילה אפס תווים או מאה תווים. לכן רק משלב זה נתחיל לספור את כמות ההזזות קדימה במילה, ונשמור את הספירה ברגיסטר R4, וזאת נעשה על מנת שאם נגלה שאחד המצבים אשר הגדרנו אינו מתאים למילה ולתחילית הספציפיים שאנו בודקים, נחזור חזרה לאינדקס אשר עד אליו אנו יודעים שהכל בסדר, ונעבור הלאה לבדיקות הבאות.

תהליך החזרת האינדקס אחורה מתבצע בלולאה לאחר כל בדיקה שכזאת, אשר נושאת שם תווית הדומה לשם הבדיקה. לאחר הבדיקה הנוספת הורדנו את הלולאה, משום שאין

בדיקות נוספות לבדוק ואם עד כאן לא נמצאה התאמה אזי התחילית אינה מתאימה למילה.

בסוף הבדיקה. הגענו לאחד משתי ההסתעפויות הבאות, failed_check ,success_check

אשר שניהם זהות לחלוטין וניתן לראות שההבדל היחידי הוא שfailed_check מוצבת מיד לאחר success_check וזאת משם שבמקרה של מציאת התאמה אנו רוצים להוסיף אחד לרגיסטר אשר שומר את מספר ההתאמות עד כה עבור התחילית הספציפית. בקטע זה של הקוד אנו מבצעים את הדברים הבאים :

מאפסים את R4 על מנת שנבצע בו שימוש חוזר בהשוואה הבאה. בודקים אם יש עוד מילים לבדוק, אם יש, עוברים למילה הבאה (מתרחש בתווית next_dict_word) ומכניסים לr2 את הכתובת של האות הראשונה בתחילית (מתרחש בתווית lp_rst_preffix), ואם אין אז בודקים אם עבור התחילית שכעת סיימנו להשוות לכל המילים במילון התקבל מספר התאמות מקסימלי מבין ההתאמות שהתבצעו עד כה. אם כן, מחליפים את התוויות הרלוונטיות שניתנו כקלט בערכים החדשים (מתבצע בתווית replace_max). לאחר מכן בודקים אם יש עוד תחיליות לבדוק. אם יש, עוברים לתחילת הבאה (תווית next_preffix) ואם אין, מסיימים את התכנית.