תרגיל 4 מחשוב מקבילי ומבוזר סמסטר קיץ 2020

יש לכתוב תוכנית שמפענחת טקסט מוצפן. את התוכנית יש לכתוב בעזרת MPI, OpenMP ו- CUDA. (ניתן להשתמש רק בשניים מבין השלושה במחיר אובדן של 15 אחוז מהציון). התוכנית תרוץ על Linux.

ההגשה ביחידים (כלומר לא בזוגות).

הקלט לתוכנית יהיה

(1) אורך מפתח ההצפנה (מספר הסיביות במפתח)

(2) קובץ המכיל את הטקסט המוצפן.

(3) קובץ המכיל מילים צפויות בטקסט (מילה אחת בכל שורה).

אלו יופיעו כ- command line arguments שתהליך 0 של MPI ישתמש בהם.
הארגומנט השלישי יהיה אופציונלי. אם אינו מופיע אז בתור ברירת מחדל יש להשתמש בקובץ המכיל מילים רבות באנגלית (ראו בהמשך).

הפלט לתוכנית יהיה מפתח ההצפנה והטקסט המפוענח. שניהם יכתבו ל- standard output.

שיטת ההצפנה

מפתח ההצפנה הוא סדרת סיביות באורך k. את הטקסט הלא מוצפן (ה- plaintext) מחלקים לתתי מחרוזות באורך k. ואז עושים xor של כל תת מחרוזת כזאת עם מפתח ההצפנה. התוצאה היא הטקסט המוצפן (ה – ciphertext).

למען הפשטות, מותר להניח ש-k שווה ל- 4 או ש-k כפולה של שמונה. כלומר $k = 4, 8, 16, \ldots$.

כדי לפענח טקסט מוצפן פועלים עליו באופן דומה: מחלקים את הטקסט המוצפן לתתי מחרוזות באורך k ועושים xor עם מפתח ההצפנה.

דוגמא: מפתח ההצפנה הוא 1001. הטקסט הוא "no cigar." (הטקסט בדוגמא אינו

כולל את הגרשיים).

נניח שהטקסט מקודד בקוד ascii. בקוד ascii כל תו מיוצג ע"י 7 סיביות משמעותיות בתוספת סיבית שמינית שהיא הסיבית השמאלית ביותר (בדוגמא הסיבית השמאלית ביותר בקידוד של כל תו היא אפס. בסביבות מסוימות הסיבית תהיה סיבית זוגיות).

בדוגמא שלנו זה יראה כך (כדי לקצר נשתמש בספרות הקסדצימליות).

plain text: `no cigar .`

ascii: 6E 6F 20 63 69 67 61 72 2E

כדי לקודד נפעיל xor עם 1001 (9 בכתיב הקסדצימלי)

plaintext:	6E	6F	20	63	69	67	61	72	2E
key:	99	99	99	99	99	99	99	99	99
	==========								
ciphertext:	F7	F6	B9	FA	F0	FE	F8	EB	B7

<u>איך מפצחים את הקוד ?</u>

נשתמש ב- "brute force":

הרעיון הוא לנסות ולפענח את הטקסט המוצפן הנתון בעזרת כל אחד ממפתחות ההצפנה האפשריים באורך k (k נתון כקלט לתוכנית). . אם התוצאה המתקבלת נראית כמו טקסט תקין אז גילינו את המפתח. תצטרכו להגדיר מה זה בדיוק אומר "להראות כמו טקסט תקין". הרעיון הוא שטקסט נראה תקין אם מופיעות בו מילים "צפויות". למשל אם ידוע שהטקסט עוסק בנושא כלכלי אז צפוי שיופיעו בו מילים כמו

"unemployment" "dollars", "budget", "bank". קובץ המכיל מילים

צפויות ינתן כ- command line argument השלישי.

ארגומנט זה הוא אופציונלי. בתור ברירת מחדל יש להשתמש בקובץ המכיל

מילים רבות באנגלית (בהנחה שמדובר בטקסט כללי באנגלית).

ב- linux ניתן למצוא קובץ כזה (ראו https://en.wikipedia.org/wiki/Words_(Unix)).

אם קבוצת המילים הצפויות היא קטנה, ניתן לבדוק בצורה פשוטה אם מילה

נתונה שייכת לקבוצה. אם הקבוצה גדולה אז עדיף להשתמש במבנה נתונים (למשל

hash table) שיאפשר לעשות את הבדיקה בצורה יעילה. ניתן להניח שהשורה הראשונה בקובץ עם המילים הצפויות מכילה את מספר המילים הצפויות.

אם אורך המפתח הוא k אז יש בסך הכל 2 בחזקת k מפתחות אפשריים.

שימו לב שאם k מספר גדול זה לא מעשי לנסות את כל המפתחות כי זה יקח זמן רב מדי.

לכן כשאתם מדבגים את התוכנית השתמשו במספרים סבירים.

בתור אופטימיזציה ניתן לקבוע שאם נתגלה מפתח שמניב טקסט מפוענח שנראה תקין אז אין צורך לנסות מפתחות נוספים.

<u>הגשה</u>

בנוסף לקוד יש להגיש תיעוד של התוכנית. בתיעוד יש לתאר רק את הנקודות המרכזיות בתוכנית למשל: מה חלוקת העבודה בין MPI, OpenMP, CUDA ?

איך נעשית הבדיקה אם מילה נתונה היא צפויה?

(אולי עשיתם שימוש בפונקצית ספריה).

בדרך כלל עמוד אחד או פחות יספיקו.

נכתב 17 אוקטובר 2020