שאלה 2:

הערות:  
כלל הזמנים נמדדו בעזרת חישוב ממוצע על time.perf\_counter()  
בהתאם לנאמר בפורום- מימוש מערך התבצע באמצעות הרשימה המובנת של פייתון  
במימוש רשימה מקושרת (להכנסות להתחלה) השתמשנו ב add\_at\_start ולא ב add\_at(0,val)

לכלל המימושים הוכנסו אותם ערכי str

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה ממוצע ערך i | עץ AVL  הכנסות להתחלה | רשימה מקושרת הכנסות להתחלה | מערך (רשימה) הכנסות להתחלה |
| 1 | 2.0396396983414887e-05 | 1.747764428310827e-06 | 1.3706668590505918e-06 |
| 2 | 2.3455800954252482e-05 | 1.747764428310827e-06 | 1.3706668590505918e-06 |
| 3 | 2.2795222803122466e-05 | 1.5842642616062436e-06 | 1.3706668590505918e-06 |
| 4 | 2.13433132265453e-05 | 1.5842642616062436e-06 | 1.3706668590505918e-06 |
| 5 | 2.4086576731254658e-05 | 1.5567264236844624e-06 | 1.3706668590505918e-06 |
| 6 | 2.3092576095627415e-05 | 1.5567264236844624e-06 | 1.3706668590505918e-06 |
| 7 | 2.4143361589974826e-05 | 1.63373760989035e-06 | 1.3706668590505918e-06 |
| 8 | 2.554414787640174e-05 | 1.63373760989035e-06 | 1.3706668590505918e-06 |
| 9 | 2.515583684934037e-05 | 1.645433010757124e-06 | 1.3706668590505918e-06 |
| 10 | 2.9198008279005686e-05 | 1.7043483151786447e-06 | 1.3706668590505918e-06 |

התחלנו מרשימות ריקות, וכדי לבחור בכל פעם היכן להכניס איבר חדש,השתמשנו בrandom.randint כדי להגריל את מיקום ההכנסה  
לכלל המימושים הוכנסו אותם ערכי str

ההכנסה לעץ AVL נעשתה באמצעות מצביע לאינדקס הראשון (איבר ה"מינימום" בעץ)

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה ממוצע ערך i | עץ AVL  הכנסות אקראיות | רשימה מקושרת הכנסות אקראיות | מערך (רשימה) הכנסות אקראיות |
| 1 | 2.663506598522266e-05 | 1.1432731834550698e-05 | 3.9566582781189717e-07 |
| 2 | 2.4573234220345816e-05 | 1.892553192252914e-05 | 4.571542347538825e-07 |
| 3 | 2.570124508606063e-05 | 3.032899337510268e-05 | 4.928450800427121e-07 |
| 4 | 2.7095746248960494e-05 | 4.080872020373742e-05 | 5.439885381288756e-07 |
| 5 | 3.0837772910793625e-05 | 5.370405353605747e-05 | 6.364198790617557e-07 |
| 6 | 2.7832612809207703e-05 | 6.643742493664224e-05 | 7.114526401529122e-07 |
| 7 | 3.0119162274613266e-05 | 8.30496885325937e-05 | 8.450832322665434e-07 |
| 8 | 2.870787342544645e-05 | 9.768053248990327e-05 | 9.302965142077657e-07 |
| 9 | 3.077174119513344e-05 | 0.00011948675589842929 | 9.302965142077657e-07 |
| 10 | 3.077174119513344e-05 | 0.0001384650979191065 | 9.302965142077657e-07 |

התחלנו מרשימות ריקות , וכל פעם הכנסו לסוף הרשימה.  
בחרנו לממש את הרשימה המקושרת ללא מצביע לסוף הרשימה, מאחר ועם מצביע לסוף הרשימה הדבר דומה קונספטואלית להכנסה לתחילת הרשימה, ורצינו להראות את התוצאות כשצריך לעבור ממש על כל הרשימה (שיערנו כי ברשימה במימוש פייתון לא נוכל לראות תוצאות אלה טוב).  
ההכנסה לעץ AVL נעשתה באמצעות מצביע לאינדקס האחרון (איבר ה"מקסימום" בעץ)

לכלל המימושים הוכנסו אותם ערכי str

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה ממוצע ערך i | עץ AVL  הכנסות בסוף | רשימה מקושרת הכנסות בסוף | מערך (רשימה) הכנסות בסוף |
| 1 | 2.5476395152509212e-05 | 1.9032958263731383e-05 | 4.737991839647293e-07 |
| 2 | 2.231754067664345e-05 | 2.9510106343593386e-05 | 2.2789944584170976e-07 |
| 3 | 2.3441462467114132e-05 | 4.7164705005863765e-05 | 2.371583961778217e-07 |
| 4 | 2.5591266031066577e-05 | 6.042495320232377e-05 | 2.1831683504084747e-07 |
| 5 | 2.7160413935780527e-05 | 7.901267144952731e-05 | 2.500830839077632e-07 |
| 6 | 2.5978793027914232e-05 | 9.180866769683773e-05 | 2.191868196758959e-07 |
| 7 | 2.729517106144201e-05 | 0.00010965462306203734 | 2.5371927767992017e-07 |
| 8 | 2.664901661531379e-05 | 0.0001421856733663215 | 2.8616255925347406e-07 |
| 9 | 2.7889802913974832e-05 | 0.00015759169364089887 | 2.158158631236465e-07 |
| 10 | 2.752777300775051e-05 | 0.0001637927483844514 | 2.2011495505770047e-07 |

ניתוח תוצאות:

נשים לב כי בהכנסה בתחילת הרשימה, הרשימה במימוש מערך (של פייתון) מבצעת את ההכנסה במהירות הטובה ביותר, בפער קטן אחריה מימוש הרשימה המקושרת ולאחר מכן רק המימוש שלנו.

בהכנסות אקראיות – כמו מקודם, מימוש פייתון היה הכי מהיר באופן עקבי בכל הגדלים, בגדלים הקטנים מאוד מימוש הרשימה המקושרת ועץ הAVL היו יחסית דומים עם יתרון קטן לAVL  
אך ככל שהגדלים גדלו, עץ הAVL צבר יתרון על פני הרשימה המקושרת.

הכנסות בסוף- כמו מקודם, מימוש פייתון היה הכי מהיר באופן עקבי בכל הגדלים, כאן בשלב מוקדם מאוד כבר היה פער כך שמימוש הAVL היה מהיר יותר מהרשימות המקושרות.

באופן כללי לאורך כל סוגי ההכנסות, רשימת פייתון הייתה הכי מהירה, ובפרט בהכנסות בסוף, בהכנסות בהתחלה המהירות בין הרשימה הפייתונית לרשימה במימוש מקושר הייתה קטנה מאוד.

לאורך כל ההכנסות, מימוש הAVL היה בעל אחידות יחסית מבחינת זמן ריצה.

הרשימה המקושרת הייתה מאוד מהירה בהכנסות לראש, אך בשאר ההכנסות הייתה יחסית איטית.

התוצאות יצאו כפי שצפינו,

אומנם רשימה של פייתון אמורה לבצע הכנסות בתהחלה בO(n) וגם הכנסות רנדומליות ידרשו העתקה של איברים פיזית בזיכרון, אך פייתון ממש רשימה זו בעזרת קוד בC שמהיר מאוד יחסית לקוד שנכתב בפייתון "טהור".

הרשימה המקושרת הייתה מאוד מהירה בהכנסות לראש, מאחר והיה לה מצביע לשם (ולמעשה כמעט השתוותה למהירות הרשימה הפייתונית), אך שאר ההכנסות היו איטיות יחסית, בפרט לסוף הרשימה מכיוון שיש ממש צורך לעבור על כל איברי הרשימה המקושרת כדי להגיע לסופה.

מהירות העץ AVL הייתה יחסית אחידה, מכיוון שבכל מקרה שנכניס, גם באמצעות המצביעים לmin ו max שיצגו את האינדקס הראשון והאחרון, אנו מבצעים עליה לאורך העץ לבדוק את הפרמטרים (ולמעשה זה O(log n) בשביל התיקונים).