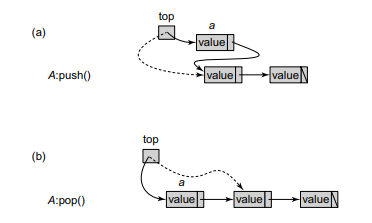
**סיכום פרויקט ניתוח ביצועים / אסף סמואל**

**סוגי תור עדיפויות:**

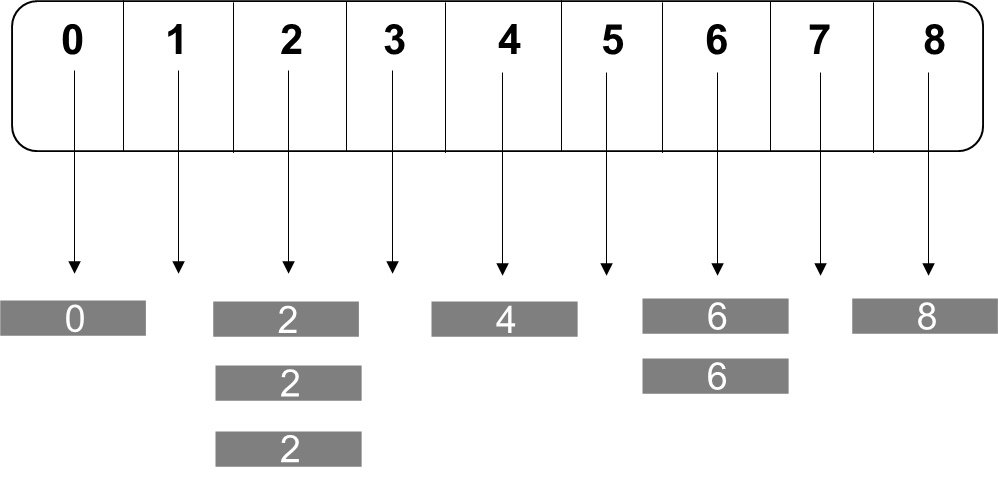
* מבוסס מערך (Array Based):

בשיטה זו אנו בונים מערך של מחסניות ( Non-Blocking Stack ) שההכנסה וההוצאה בו מתבצעת ע"י שימוש ב-CAS.



**הוספה:** כאשר נרצה להוסיף את הערך X לתור העדיפויות אנחנו נוסיף את הערך למחסנית שנמצאת במערך באינדקס X.

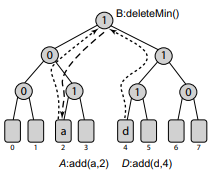
**הסרה:** כאשר נרצה להוציא את המינימום, נעבור על כל המערך ולכל מחסנית ננסה להוציא ערך. אם הערך שונה מ-null (כלומר המחסנית לא ריקה) נחזיר אותה. אם עברנו על כל המערך וכל המחסניות החזירו null, נחזיר null.



*[ בשיטה זו כפי שניתן לראות מהמדידות – הכנסה בזמן סביר וההוצאה איטית ]*

* מבוסס עץ (Tree Based):

בשיטה זו, בדומה לשיטה הקודמת, אנחנו בונים עץ בינרי – כך שכל צומת תכיל סופר (Counter) שיהיה Atomic שמאותחל ל-0, וכל עלה יכיל מחסנית.



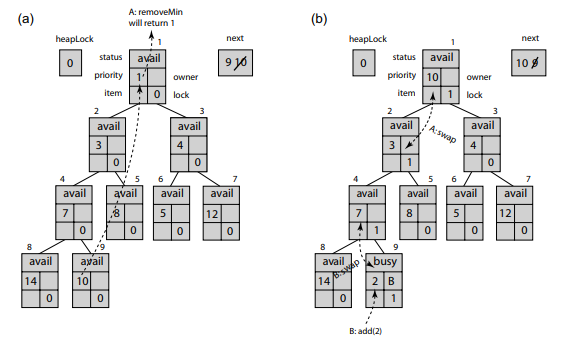
**הוספה:** כאשר נרצה להוסיף את הערך X לתור העדיפויות, אנחנו נוסיף את הערך למחסנית שנמצאת בעלה שהאינדקס שלו X (אנחנו שומרים מערך של עלים במבני נתונים זה), נעלה ברמות העץ - ולכל הורה אם אנחנו הגענו מהבן השמאלי נגדיל את ה-Counter ב-1. נמשיך כך עד שניע לשורש העץ (Root).

**הסרה:** כאשר נרצה להוציא את המינימום, נתחיל מהשורש ונבדוק אם ה-Counter שלו גדול מ-0. אם כן, תקטין אותו ותעבור לבן השמאלי. אחרת עבור לבן הימני. ככה נמשיך לכל הצמתים עד שנגיע לעלה. ברגע שהגענו לעלה נוציא את הערך מהמחסנית (אם קיים) ונחזיר אותו.

*[ בשיטה זו כפי שניתן לראות במדידות – ההכנסה וההוצאה מתבצעת בזמן סביר ]*

* מבוסס ערימה (Heap):

בשיטה זו אנו בונים מערך שמייצג ערימת מינימום, וערך שלם (Integer) שמייצג את האינדקס הבא הפנוי במערך (חשוב לציין שבשיטה זו אנחנו משתמשים במנעולים, כך שבכל פעם שנרצה להשתמש באיבר מהמערך ננעל אותו לפני).



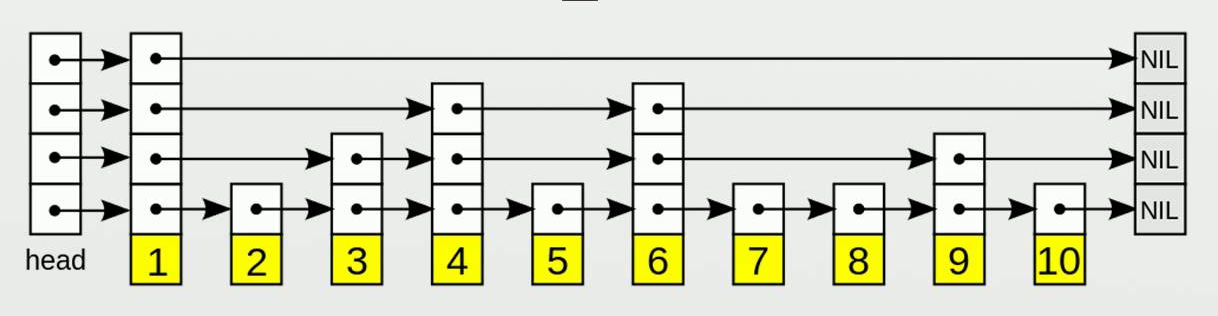
**הוספה:** כאשר נרצה להוסיף ערך X לתור העדיפויות, נוסיף אותו לערך הפנוי הבא במערך, ונסדר את המערך כך שהאיבר הראשון יכיל את הערך המינימלי ביותר.

**הסרה:** כאשר נרצה להוציא את המינימום, נוציא את האיבר הראשון במערך, ונחליף אותו עם הערך האחרון במערך. נבצע שוב סידור של המערך כך, שהאיבר הראשון יהיה המינימלי בערימה.

*[ בשיטה זו כפי שניתן לראות מהמדידות ההכנסה וההוצאה בזמן איטי בגלל השימוש במנעולים ]*

* מבוסס Blocking Skip List:

בשיטה זו אנחנו משתמשים במבני נתונים Skip-List עם מנעולים, כדי לייצג את התור העדיפויות שלו.



**הוספה:** כאשר נרצה להוסיף את הערך X לתור העדיפויות, נבדוק קודם אם הוא כבר קיים – אם כן תחזיר false. אחרת נוסיף אותו ל-Skip List (אנחנו נועלים את כל האיברים הקודמים לאותו ערך שנוסיף, ומשנים את המצביעים).

**הסרה:** כאשר נרצה להוציא את הערך המינימלי, נצטרך להוציא את הערך הראשון ב-Skip List שאינו מסומן (איבר מסומן הוא איבר ש-Thread אחר רוצה למחוק אותו..). אנחנו נשנה את ה-Skip List כך שה-head יצביע לאיברים הבאים לאותו איבר שאנחנו מוחקים.

*[ בשיטה זו כפי שניתן לראות במדידות ההכנסה איטית וההוצאה מהירה – בגלל השימוש במנעולים ]*

* מבוסס Free Lock Skip List:

בשיטה זו, בדומה לשיטה הקודמת, אנחנו גם משתמשים ב-Skip List רק שכעת במקום מנעולים, אנחנו משתמשים בפעולות שהם Free Lock (כדוגמת CAS). בשיטה זו, השתמשתי במבני הנתונים E , Boolean>>ConcurrentSkipListMap.

**הוספה:** כאשר נרצה להוסיף את הערך X לתור העדיפויות, נוסיף את הערך X עם הערך true לממפה.

**הסרה:** כאשר נרצה להוציא את הערך המינימלי מהתור, נסיר את הערך ה-X מהמפה, ונבדוק אם הוא לא null או ערך שלו אינו false.

*[ בשיטה זו כפי שניתן לראות במדידות ההכנסה וההוצאה מתבצעים בזמן מהיר ]*