

חזרה בנושא אוסף כללי

תור קדימויות

מטרות

כתיבת מחלקה המגדירה אוסף כללי.

רמת השאלה

בינונית עד קשה. מתאימה גם כשאלת בגרות.

תור-קדימויות (Priority Queue) הוא אוסף כללי המתנהג כתור אך עבור כל איבר באוסף מוגדרת "קדימות". קדימות זו קובעת את הסדר לפיו יצאו האיברים מהתור: האיבר בעל הקדימות הגבוהה ביותר יצא מהתור ראשון גם אם הוא לא הבא על פי סדר ההכנסה לתור. פעולת ההוצאה מוציאה מהתור את האיבר בעל הקדימות הגבוהה ביותר. אם יש כמה איברים בקדימות זו, יצא מביניהם זה שנכנס ראשון לתור.

המחלקה `PQueue<T>`

מגדירה תור קדימויות כללי שטיפוס איבריו גנרי.

<code>PQueue()</code>	הפעולה בונה תור-קדימויות ריק
<code>boolean isEmpty()</code>	הפעולה מחזירה 'אמת' אם תור-הקדימויות ריק, ו-'שקר' אחרת
<code>void insert (T x, int priority)</code>	הפעולה מכניסה את הערך <code>x</code> בעל קדימות <code>priority</code> לתור. הנחה: <code>priority</code> היא מספר טבעי
<code>T remove()</code>	הפעולה מוציאה מהתור את הערך שהוכנס ראשון מבין הערכים בעלי הקדימות הגבוהה ביותר, ומחזירה אותו. הנחה: התור אינו ריק

מה עליכם לעשות?

- א. כתבו את המחלקה `PQueue`.
- ב. מהי היעילות של פעולות הממשק על פי הייצוג שבחרתם. הסבירו.

הנחיות מיוחדות

1. שאלה זו בודקת את היכולת של התלמיד לבנות מחלקה המגדירה אוסף כללי. באומרנו "אוסף כללי" אנו מתייחסים לעובדה שאין כאן עיסוק בבעיה ספציפית מסוימת אלא בהגדרת טיפוס נתונים כללי שיוכל לשמש לאחסון נתונים בבעיות שונות העוסקות באוספים המתנהגים על פי הפרוטוקול של תור קדימויות. ברור שכפועל יוצא מכך התור מוגדר על טיפוס גנרי של עצמים.
2. בררו שהתלמידים מבינים את משמעות הניהול של התור. הדגימו בעזרת טרמפיאדה של חיילים בעלי דרגות שונות, דוא"ל שבו סדרי עדיפויות וכד'.
3. התלמיד צריך לבחור ייצוג מתאים לתור-הקדימויות. על התלמיד להתחשב בשני דברים:
 - א. היכן יישמרו האיברים שיוכנסו לתור-הקדימויות. מאחר ותור-קדימויות הוא אוסף כללי יש לבחור באוסף כללי דינמי כגון רשימה או שרשרת חוליות. מערך לא מתאים לבעיה.
 - ב. איך לשמור איבר שמורכב מזוג: ערך וקדימות. כלומר, התלמיד צריך להבין שיש להשתמש במחלקת עזר פשוטה נוספת המגדירה זוג גנרי. זהו תרגול מצוין לנושא הגנריות.
4. מימוש פעולות הממשק של תור-קדימויות מסתמך על הייצוג שנבחר. פעולות ההכנסה וההוצאה הן הפעולות המעניינות יותר. כאן התלמיד צריך להחליט איך לממש אותן. שתי אפשרויות עומדות בפניו:
 - א. לשמור על איברי תור-הקדימויות ממוינים כל הזמן על פי הקדימות כך שהאיבר בעל הקדימות הגבוהה ביותר יהיה תמיד הראשון באוסף. כלומר, פעולת ההכנסה מבצעת הכנסת ערך חדש לאוסף ממוין. ואילו פעולת ההוצאה פשוטה יותר – מוציאה תמיד את הערך הראשון באוסף.
 - ב. לא לשמור את תור-הקדימויות ממוין. כלומר, פעולת ההכנסה תכניס הערך חדש כראשון באוסף. ואילו פעולת ההוצאה תהיה יותר מורכבת, מכיוון שעליה לאתר את האיבר בעל הקדימות הגבוהה ביותר. (שימו לב שלמרות שבתור ההכנסה היא בסופו, הרי כאן אנו בתוך המחלקה ומתייחסים לייצוג הנבחר ולכן ההכנסה הנוחה היא דווקא לראש הרשימה!).

פתרון

א. המחלקה PQueue:

```
public class PQueue<T>
{
    private List<Pair<T>> data;

    public PQueue ()
    {
        this.data = new List<Pair<T>> ();
    }

    public boolean isEmpty ()
    {
        return this.data.isEmpty ();
    }

    public void insert (T x, int priority)
    {
        Node<Pair<T>> prev = null;
        Node<Pair<T>> pos = this.data.getFirst ();

        while (pos != null && priority >= pos.getInfo ().getPriority ())
```

```

    {
        prev = pos;
        pos = pos.getNext();
    }

    this.data.insert(prev, new Pair<T>(x,priority));
}

public T remove()
{
    T x = this.data.getFirst().getInfo().getItem();
    this.data.remove(this.data.getFirst());
    return x;
}
}

```

כפי שניתן לראות, ממימוש הפעולה insert, תור-הקדימויות נשמר ממזין כל הזמן על פי הקדימות. כלומר, פעולת ההכנסה מבצעת הכנסה לאוסף הממוין, כך שהערך החדש מוכנס למקומו המתאים על פי קדימותו. הערך בעל הקדימות הגבוהה ביותר יהיה תמיד ראשון. פעולת ההוצאה פשוטה יותר. הפעולה מוציאה תמיד את הערך הראשון בתור.

ניתן לממש את פעולות המחלקה כך שתור-הקדימויות לא יישמר ממזין. במקרה זה פעולת ההכנסה תהיה פשוטה ותכניס את הערך החדש לתחילת הרשימה (זו ההכנסה בעלת היעילות הטובה ביותר). במקרה זה תהיה פעולת ההוצאה מורכבת יותר. בכל הוצאה תצטרך הפעולה לסרוק את הרשימה כדי לאתר את הערך בעל העדיפות הגבוהה ביותר ולהוציאו.

תור הקדימויות משתמש במחלקת העזר הפשוטה Pair המגדירה זוג: ערך וקדימות. לפניכם קוד המחלקה:

```

public class Pair<T>
{
    private T value;
    private int priority;

    public Pair(T value, int priority)
    {
        this.value = value;
        this.priority = priority;
    }

    public int getPriority()
    {
        return priority;
    }

    public T getValue()
    {
        return value;
    }

    public String toString()
    {
        return this.value + ":" + this.priority;
    }
}

```

ב. יעילות פעולות על פי המימוש שהוצג לעיל:

הפעולות `remove()`, `isEmpty()`, `PQueue()` כולן מתבצעות ב- $O(1)$.

הפעולה `insert(...)` היא מסדר גודל $O(n)$.

הערה: מאחר ותור-הקדימויות מיוצג על ידי רשימה חישוב היעילות של הפעולות תלויה ביעילות הפעולות של רשימה. בכל זאת כאשר ידוע שהשימוש בפעולות הממשק של הרשימה הוא שימוש ספציפי – נוכל להתייחס לכך בחישוב היעילות. לדוגמה, בעוד שהוצאה מרשימה היא מסדר גודל $O(n)$ הרי שהוצאה מתור קדימויות הנשמר ממוין כל הזמן, נעשית תמיד בפנייה ישירה לראש התור ולכן למרות שמימוש הפעולה מזמן את הוצאה מרשימה, הרי שאנו מתייחסים כאן בוודאות למקרה הטוב ביותר של הוצאה מרשימה שיעילותו קבועה.

דיון זה במלואו מופיע בפרק רשימה בדיון על יעילות הפעולה `remove(...)` (הוצאה מתחילת הרשימה).

מקור השאלה

מדריך למורה "עיצוב תוכנה", תשנ"ז, מתוך שאלון בגרות לדוגמה - שאלה 3 עמוד 206.