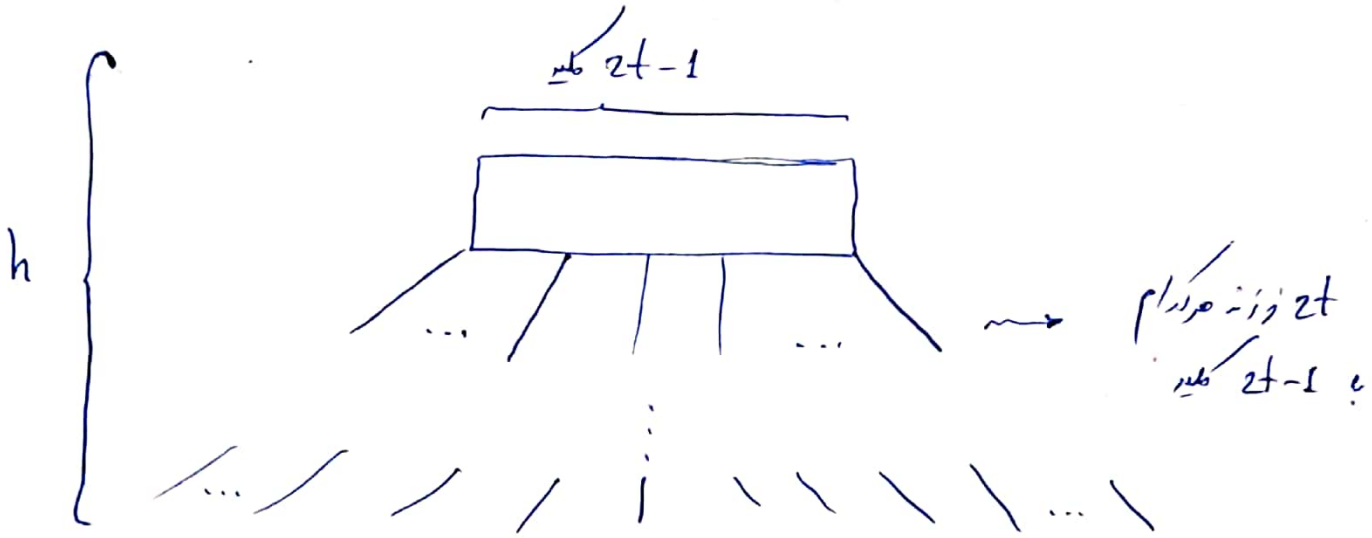
محمد هادی آقایی
شماره دانشجویی: 81019938

«پاسخ تمرین ششم»

(1) می دانیم اگر درجه‌ی B-tree برابر t باشد آن‌گاه هر گره حداکثر $2t-1$ کلید دارد ⇐
هر گره حداکثر $2t$ فرزند دارد و هر فرزند نیز حداکثر $2t-1$ کلید خواهد داشت.
⇐ برای محاسبه حداکثر تعداد کلیدها خواهیم داشت:



⇐ خواهیم داشت:

بیشینه تعداد کلیدها $= 2t-1 + 2t \times (2t-1) + (2t)^2 \times (2t-1) + \cdots + (2t)^h (2t-1)$

$$= (2t-1)\left(1 + 2t + (2t)^2 + \cdots + (2t)^h\right)$$

$$= (2t-1) \times \frac{(2t)^{h+1} - 1}{2t-1} = \boxed{(2t)^{h+1} - 1}$$

محمدجواد شیرازی شماره دانشجویی: 810199386

(2) اگر در عناصر برگی که x داخل آن قرار دارد، برای x برادر سمت چپ وجود داشته باشد، همان برادر سمت چپ x، بزرگ‌ترین عنصر کوچک‌تر از x می‌شود. چون یا x و برادرش از پدرشان بزرگ‌ترند یا کوچک‌تر.

در غیر این صورت اگر برگی که x داخل آن است، فرزند راست پدرش باشد $\Rightarrow$ پدر x، همان بزرگ‌ترین عنصر کمتر از x می‌شود. (چون x برادر سمت چپ ندارد.)

در غیر این صورت (x نه برادر چپ داشته باشد و نه فرزند سمت راست پدرش باشد.) باید دید در اجداد x کدام گره‌ها فرزند چپ پدر خود هستند و در بین آن‌ها دنبال بزرگ‌ترین عنصر کوچک‌تر از x باشیم که در این صورت باید به اندازه ارتفاع درخت (h) به بالای گره پدر داریم.

$\Rightarrow$ در بهترین حالت یک گره را بررسی می‌کنیم و در بدترین حالت h گره

از طرفی چون بدترین حالت برای سمت چپ‌ترین عنصر یک برگ که فرزند راست پدر خودش است، رخ می‌دهد $\Rightarrow$ بدترین حالت خیلی کم رخ می‌دهد چون به ازای یک عنصر خاص از یک برگ خاص است.

محمد مهراد شراتی شماره دانشجویی : 810199386

(3) با توجه به اینکه در هر روز به یک نفر می‌توانیم شکلات بدهیم => در هر روز به فردی که بیشتر از دیگران ناراحت می‌شود، شکلات می‌دهیم (s_i بیشتری داشته باشد) هرگاه به اندازه l_i به چنین فردی شکلات دادیم، او را کنار می‌گذاریم => باید از یک maxHeap استفاده نماییم

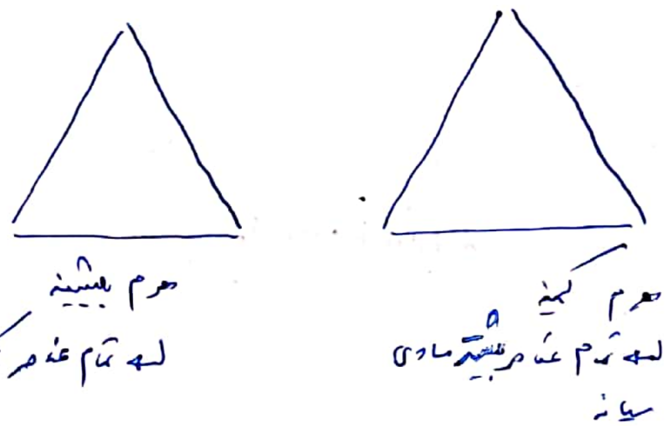
=> در هر روز مسابقه فرد با بیشترین s_i را پیدا می‌کنیم و در top هرم قرار می‌دهیم، و به همین فرد شکلات می‌دهیم. اگر افراد هم با توجه به s_i ای که دارند، در مکان مناسب شان در MaxH اضافه می‌شوند، هرگاه به اندازه l_i عنصر top هرم، به عنصر top شکلات دادیم، او را کنار گذاشته و فرد با s_i بیشتر را از بین افراد باقی مانده جایگزین او می‌کنیم.

حال چون N شرکت کننده داریم => باید N بار در Max Heap مان عملیات درج انجام دهیم و هزینه درج هم $\log N$ است => برای این‌ها باید $N \log N$ هزینه کنیم. از طرفی مسابقه در K روز برگزار شده است و نهایتاً K بار، یک فرد از مسابقه کنار می‌رود (K بار نیاز به تعیین top جدید است) که هزینه هر بار تعیین top جدید هم و حذف top قبلی $\log N$ است => در این جا هم حداکثر $K \log N$ هزینه می‌کنیم و هزینه دیگری هم نداریم => داریم

$$\text{هزینه} = O(K \log N) + O(N \log N) = O((N+K) \log N)$$

محمد جواد شب آویز   شماره دانشجویی: 810199386

④ باید از دو هرم استفاده کنیم. یک هرم کمینه و یک هرم بیشینه. که هر هرم هم حدوداً نصف عناصر را شامل شود (اگر تعداد عناصر زوج باشند هر هرم دقیقاً نصف عناصر را دارد و اگر تعداد عناصر فرد باشند، یک هرم، یک عنصر بیشتر از دیگری خواهد داشت). حال تمام عناصری که از میانه کوچک‌تر باشند یا مساوی میانه باشند در هرم بیشینه، درج می‌کنیم و تمام عناصری که از میانه بزرگ‌تر باشند یا مساوی میانه باشند، در هرم کمینه درج می‌کنیم:



حال اگر هرم کمینه یک عنصر بیشتر از هرم بیشینه داشته باشد، میانه همان top هرم کمینه است و اگر هرم بیشینه یک عنصر بیشتر از هرم کمینه داشته باشد، میانه همان top هرم بیشینه می‌شود. اگر هم تعداد عناصر دو هرم کمینه و بیشینه مساوی باشند، میانه را برابر میانگین top دو هرم درنظر می‌گیریم.

حال برای درج عنصر، میانه را با عنصری که می‌خواهیم اضافه کنیم مقایسه می‌کنیم اگر عنصر جدید، بزرگ‌تر از میانه باشد، آن را در هرم کمینه درج می‌کنیم و اگر عنصر جدید کوچک‌تر از میانه باشد، آن را در هرم بیشینه درج می‌کنیم. حال نکته مهمی که باید توجه کنیم این است که ممکن است در حین درج تفاوت عنصرهای دو هرم بیش از یکی شود برای رفع این مشکل کافی است عنصر top هرمی که تعداد عنصر بیشتری دارد را برداشته و در هرم با تعداد عنصر کمتر درج کنیم.

محمد جبار شبرتی

شماره دانشجویی: 810199386

```
class MPQ :

    def __init__(self, MinHeap, MaxHeap) :
        self.minHeap = MinHeap
        self.maxHeap = MaxHeap

    def getMean (self) :
        if len(self.MaxHeap) < len(self.Min Heap) :
            return self.Min Heap.top
        elif len(self.Max Heap) > len (self.Min Heap) :
            return self.Max Heap.top
        elif len(self.maxHeap) != 0 :
            return (self.Max Heap.top + self.min Heap.top)
                                                         /2
        else :
            return 0
```

```
def insert(self, x):
    mean = self.getMean()
    if len(MaxHeap) < len(MinHeap):
        if x < mean:
            MaxHeap.append(x)
        else:
            MinHeap.pop(len(MinHeap)-1)
            MaxHeap.append(mean)
            MinHeap.append(x)
    elif len(MinHeap) < len(MaxHeap):
        if x > mean:
            MinHeap.append(x)
        else:
            MaxHeap.pop(len(MaxHeap)-1)
            MinHeap.append(mean)
            MaxHeap.append(x)
    else:
        if mean < x:
            MinHeap.append(x)
        else:
            MaxHeap.append(x)
```

محمد جواد برتی   شمارهٔ دانشجویی: 810199386

```
def deleteMean(self):
    if len(self.maxHeap) < len(self.MinHeap):
        self.MinHeap.pop(len(self.MinHeap)-1)
    elif len(self.maxHeap) > len(self.MinHeap):
        self.MaxHeap.pop(len(self.MaxHeap)-1)
```

---

(5) فرض می‌کنیم دو پشتهٔ s_1 و s_2 داشته باشیم. برای پیاده‌سازی عملیات enqueue هر عنصر جدید را در s_1، push می‌کنیم و مشکلی هم نداریم. حال برای پیاده‌سازی عملیات dequeue اگر s_1 خالی نباشد، ابتدا عناصر s_1 را وارد s_2 می‌کنیم تا s_1 خالی شود. حال از s_2، pop می‌کنیم. اگر هم s_1 خالی باشد ولی s_2 خالی نباشد، از s_2 pop می‌کنیم.

کاملاً مشخص است که هزینهٔ انجام enqueue از $O(1)$ است. حال برای محاسبهٔ هزینهٔ dequeue از روش حسابداری استفاده می‌کنیم. فرض می‌کنیم n عملیات شامل enqueue و dequeue انجام داده‌ایم. حال به ازای هر enqueue، 1 ریال هزینه می‌کنیم و 3 ریال پس‌انداز می‌کنیم. حال اگر K عمل enqueue انجام شده باشد، 3K ریال پس‌انداز داریم و اگر بخواهیم یک dequeue انجام دهیم، با K ریال پس‌اندازمان عناصر s_1 را pop می‌کنیم و با K ریال دیگر در s_2 push می‌کنیم و یک pop انجام می‌دهیم (در واقع می‌توان گفت ابتدا K عنصر را pop کرده‌ایم از s_1، حال K ریال برای push در s_2 هزینه می‌کنیم و اگر بخواهیم K pop کنیم، K ریال دیگر) $\Longleftarrow$ 3K ریال پس‌اندازمان برای dequeue کافی است $\Longleftarrow$ برای هر enqueue، 4 ریال خرج می‌شود (1 ریال هزینهٔ enqueue + 3 ریال پس‌انداز) و برای dequeue چیزی خرج نمی‌شود.

در n عملیات شامل nq, dq $\Longrightarrow$ $\frac{4 \times n}{n} = 4 \rightarrow \frac{\text{هزینهٔ کل}}{n}$ (هزینه به ازای هر عنصر)

محمدجواد آقایی شماره دانشجویی: 81019938

از طرفی می دانیم هزینه هر enqueue، 1 بود پس داریم:

$$\frac{\text{هزینه کل } n \text{ عملیات}}{n} = \text{هزینه enqueue} + \text{هزینه dequeue}$$

$$\Rightarrow O(4) = O(1) + \text{هزینه dequeue} \Rightarrow \text{هزینه dequeue} = 4-1 = \boxed{O(3)}$$

---

(6) حل به روش aggregate: مجموع هزینه ها را حساب می کنیم که از رابطه زیر به دست می آید:

$$\sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i + \sum_{\substack{i=0 \\ , i \neq 2^i}}^{n} 1 = C_i$$

کافیست واضح است که حاصل $\sum$ دومی کمتر یا مساوی n است پس داریم:

$$C_i \leq \sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i + n = \frac{2^{\lfloor \log_2 n \rfloor + 1} - 1}{2-1} + n$$

$$\Rightarrow C_i \leq 2n - 1 + n \Rightarrow C_i < 3n$$

$\Leftarrow$ هزینه کل نهایتا $3n$ است $\Rightarrow$ هزینه هر مرحله $= \frac{3n}{n} = 3$ است

$$\Rightarrow \text{هزینه هر مرحله} = O(3)$$

حل به روش accounting: اگر i توانی از 2 نباشد، 1 واحد را هزینه می کنیم و 2 واحد پس انداز می کنیم. اگر i توانی از 2 باشد، i واحد از پس اندازمان برمی داریم و 3 واحد پس انداز می کنیم. از طرفی می دانیم هیچ گاه پس اندازمان منفی نخواهد شد

محمد جواد ترابی شماره دانشجویی: 810199386

چون در هر مرحله 3 واحد پرداخت داریم ⇐ در کل $3n$ واحد پرداخت کرده‌ایم. از طرفی طبق قسمت قبلی دیدیم که مجموع هزینه‌ها کمتر از $3n$ است ⇐ هیچ‌گاه پس انداز ما منفی نمی‌شود.

⇐ با این روش هزینه کل $3n$ و هزینه هر مرحله میانگین $\frac{3n}{n} = 3$ خواهد شد.