انجمن علمی مهندسی کامپیوتر
دانشگاه مازندران

به نام خدا

پاسخ تمرین اول ساختمان داده‌ها و الگوریتم

الگوریتم‌ها و حل مسئله - پیچیدگی زمانی - الگوریتم های بازگشتی

استاد:

دکتر روستایی

استادیار:

عرشیا عموزاد

۱. ۱.

```
for (int i = 1; i <= p; i++){
      for (int j = 0; j <= 2){
           swap M[i][j] with M[j][i]
       }
   }
```

۲. ابتدا ماتریس اسپارس را تبدیل به ماتریس متراکم می‌کنیم:

$$\begin{bmatrix} 7 & 10 & 10 \\ 0 & 4 & 120 \\ 1 & 6 & 101 \\ 2 & 4 & 83 \\ 3 & 5 & 54 \\ 5 & 1 & 9 \\ 6 & 5 & 7 \\ 7 & 4 & 3 \\ 8 & 3 & 2 \\ 9 & 0 & 1 \\ 9 & 5 & -6 \end{bmatrix}$$

حال با توجه به مرتب بودن ماتریس بر اساس مقادیر می توان عملیات جستجوی دودویی را بر روی آرایه ای از value های ماتریس متراکم انجام داد.
این کار همانند انجام جستجوی دودویی بر روی آرایه ای با ۱۰ عنصر است که تعداد مقایسه ها در حالت متوسط می‌شود:

$\sum_{i=1}^{\log n} \frac{i*2^{i-1}}{n} \Rightarrow \frac{3+2+3+4+1+3+4+2+3+4}{10} = \frac{29}{10}$

۲. یادآوری: در ضرب ماتریس ها اگر ابتدا ماتریس هایی را ضرب کنیم که بعد وسط آن‌ها بزرگتر و بعد کناری آن‌ها کوچکتر است، تعداد ضرب ها کمتر می‌شود.
در این مثال بهترین حالت میشود:

$A * ((B * C) * D)$

با مجموع تعداد ضرب‌های:

$(2 * 25 * 3) + (2 * 3 * 4) + (10 * 2 * 4) = 254$

۳. شرح مرحله به مرحله:

۱. $token = start, stack = [], output = []$
۲. $token = A, stack = [], output = [A]$
۳. $token = *, stack = [*], output = [A]$
۴. $token = (, stack = [(, *], output = [A]$
۵. $token = B, stack = [(, *], output = [A, B]$
۶. $token = -, stack = [-, (, *], output = [A, B]$
۷. $token = D, stack = [-, (, *], output = [A, B, D]$
۸. $token =), stack = [*], output = [A, B, D, -]$
۹. $token = /, stack = [/], output = [A, B, D, -, *]$
۱۰. $token = E, stack = [/], output = [A, B, D, -, *, E]$
۱۱. $token = -, stack = [-], output = [A, B, D, -, *, E, /]$
۱۲. $token = F, stack = [-], output = [A, B, D, -, *, E, /, F]$
۱۳. $token = *, stack = [*, -], output = [A, B, D, -, *, E, /, F]$
۱۴. $token = (, stack = [(, *, -], output = [A, B, D, -, *, E, /, F]$
۱۵. $token = G, stack = [(, *, -], output = [A, B, D, -, *, E, /, F, G]$
۱۶. $token = +, stack = [+, (, *, -], output = [A, B, D, -, *, E, /, F, G]$
۱۷. $token = H, stack = [+, (, *, -], output = [A, B, D, -, *, E, /, F, G, H]$

۱۸. $token = /, stack = [/, +, (, *, -], output = [A, B, D, -, *, E, /, F, G, H]$

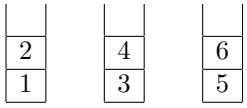
۱۹. $token = K, stack = [/, +, (, *, -], output = [A, B, D, -, *, E, /, F, G, H, K]$

۲۰. $token =), stack = [*, -], output = [A, B, D, -, *, E, /, F, G, H, K, /, +]$

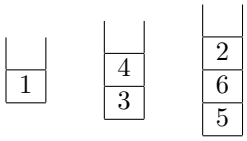
۲۱. $token = end, stack = [], output = [A, B, D, -, *, E, /, F, G, H, K, /, +, *, -]$

خروجی نهایی: $ABD-*E/FGHK/+*-$

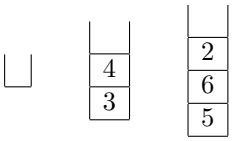
۴. شرح مرحله به مرحله:



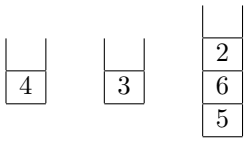
$\Downarrow poppush(1,3)$



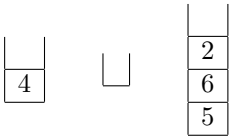
$\Downarrow pop(1)$



$\Downarrow (poppush(2,1))$

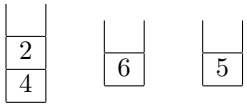


$\Downarrow pop(2)$



$\Downarrow poppush(3,1)$

$\Downarrow poppush(3,2)$



$\Downarrow pop(3), pop(1), pop(1), pop(2)$

$Output = 1, 3, 5, 2, 4, 6$

number of poppush calls: 4

۵. الگوریتم ذکر شده آینه‌ای بودن(palindrome) یک لیست پیوندی یک طرفه را بررسی می‌کند.
در این الگوریتم از یک پشته جهت ذخیره سازی و معکوس کردن نیمه اول لیست استفاده شده؛ به این صورت که با روش نشانگر تند و کند وسط لیست پیدا می‌شود.
سپس عناصر نیمه اول لیست به به پشته می‌روند.
if بررسی میکند که آیا لیست طولش فرد است یا خیر که اگر فرد باشد باید برای بررسی آینه‌ای بودن لیست عنصر میانی را در نظر نگیریم.
در بخش انتهایی تابع هم عناصر نیمه دوم لیست پیوندی با معکوس نیمه اول لیست مقایسه میشوند تا آینه‌ای بودن یا نبودن لیست مشخص شود.

۶. دو اشاره‌گر به ابتدای لیست نگه دارید. در هر مرحله، با استفاده از تابع next اولی را یک واحد افزایش دهید و دومی را ۲ واحد. سپس این دو اشاره‌گر را مقایسه کنید. اگر یکسان بودند، دور وجود دارد. برای اثبات درستی، دقت کنید که اگر دور نباشد، هرگز این دو اشاره‌گر یکی نمی‌شوند و در نهایت، در زمان خطی، یکی از آن‌ها به ته لیست می‌رسد (اولی اشاره‌گر دوم می‌رسد). اگر دور وجود داشته باشد، پس از زمان خطی، هر دو اشاره‌گر وارد دور می‌شوند. هنگامی که درون دور قرار گرفتند، چون در هر مرحله، یکی با سرعت ۲ و دیگری با سرعت ۱ حرکت می‌کند، مثل این است که یکی ثابت است و دیگری با سرعت ۱- حرکت می‌کند. پس در زمان خطی به هم می‌رسند.

۷. ۱. تغییرات صف:

$$[] \to [10] \to [10, 20] \to [10, 20, 30] \to [20, 30] \to [20, 30, 40]$$
$$\to [20, 30, 40, 50] \to [30, 40, 50] \to [30, 40, 50, 60] \to [40, 50, 60]$$

۲. بررسی مرحله به مرحله:

$$start : front = -1, rear = -1$$
$$enqueue(10) : front = 0, rear = 0$$
$$enqueue(20) : front = 0, rear = 1$$
$$enqueue(30) : front = 0, rear = 2$$
$$dequeue() : front = 1, rear = 2$$
$$enqueue(40) : front = 1, rear = 3$$
$$enqueue(50) : front = 1, rear = 4$$
$$dequeue() : front = 2, rear = 4$$
$$enqueue(60) : front = 2, rear = 0$$
$$dequeue() : front = 3, rear = 0$$

پاسخ نهایی:
$$rear = 0, front = 3$$