Complexity of Events

0) Insert

Complexity: O(n)

دو حلقه وابسته به اندازه در این تابع وجود دارد:

```
78                while (cur.next_in_row != null && cur.next_in_row.col < col)
79                    cur = cur.next_in_row;
```

شرط اتمام این حلقه، رسیدن به ستون مورد نظر است. بدترین حالت آن آخرین ستون است. پس اگر عرض ماتریس (اندازه هر سطر) را n در نظر بگیریم، دستور اصلی در بدترین حالت n بار تکرار می شود.

```
94                while (cur.next_in_col != null && cur.next_in_col.row < row)
95                    cur = cur.next_in_col;
```

حلقه دوم نیز مشابه قبلی و برای پیمایش در یک ستون است. اگر اندازه سطر و ستون ماتریس را یکسان درنظر بگیریم (بهتر است n ماکزیمم اندازه سطر و ستون باشد)، این حلقه نیز در بدترین حالت n بار تکرار می شود. بنابرین پیچیدگی تابع از مرتبه $O(n)$ است.

1) Delete

Complexity: O(n)

در این تابع نیز دو حلقه عینا مشابه عکس های قبل وجود دارد و توضیحات یکسان است.

2) Search

Complexity: $O(n^2)$

در اینجا نیز فرض میکنیم n ماکزیمم طول و عرض ماتریس باشد.

در این تابع، یک حلقه for ، سطر های ماتریس را پیمایش میکند، پس n تکرار دارد.

داخل آن حلقه while ، ستون های مربوط به هر سطر را پیمایش می کند و این حلقه نیز حداکثر n بار تکرار می شود.

پس تعداد تکرار دستورات اصلی $n * n$ است و پیچیدگی این تابع $O(n^2)$ بدست می آید.

```
137
138       for (Node head : R) {
139           Node cur = head;
140
141           while (cur != null) {
142               if (cur.value == value) {
143                   System.out.printf("(%d, %d)\n", cur.row, cur.col);
144                   found = true;
145               }
146               cur = cur.next_in_row;
147           }
148       }
```

3) Update

Complexity: O(n)

```
160               while (cur != null && cur.col < col)
161                   cur = cur.next_in_row;
```

حلقه وابسته به اندازه در این تابع، ستون های سطر مورد نظر را پیمایش می کند و حداکثر n مرتبه تکرار می شود.

4) Print

Complexity: $O(n^2)$

```
173             for (Node head : R) {
174                 int col = 0;
175                 Node cur = head;
176 
177                 while (cur != null) {
178                     for (int i = col; i < cur.col; i++)
179                         System.out.print("0\t");
180 
181                     System.out.print(cur.value + "\t");
182                     col = cur.col + 1;
183                     cur = cur.next_in_row;
184                 }
185 
186                 for (int i = 0; i < width - col; i++)
187                     System.out.print("0\t");
188 
189                 System.out.println();
190             }
191         }
```

در تابع پرینت با کد صفر، حلقه for بیرونی به تعداد سطر ها (n) تکرار می شود. حلقه while درونی نود های هر سطر را پیمایش میکند و for درون آن صفر های بین هر دو نود متوالی را چاپ می کند. آخرین for هم صفر های باقیمانده بعد از آخرین نود را چاپ می کند. پس در مجموع این سه حلقه پیچیدگی n دارند. پس دستورات اصلی چهار حلقه در کل n*n مرتبه تکرار می شود.

```
197             for (Node head : R) {
198                 Node cur = head;
199                 while (cur != null) {
200                     System.out.printf("%-5d %-5d %-6d %-6s %-6s\n",
201                             cur.row, cur.col, cur.value,
202                             cur.next_in_row, cur.next_in_col);
203 
204                     cur = cur.next_in_row;
205                 }
206             }
```

در تابع پرینت با کد 1 هم حلقه for به تعداد سطر ها و while درون آن حداکثر به تعداد ستون ها تکرار می شود. پس دستورات اصلی n*n مرتبه تکرار می شوند.

بنابرین در هر دوحالت پیچیدگی تابع پرینت $O(n^2)$ است.

5) Save File

Complexity: $O(n^2)$

این رویداد هیچ تفاوتی با پرینت ندارد و فقط بجای چاپ در کنسول، درون فایل رایت شده است.