CSE 331
Software Design & Implementation

Kevin Zatloukal
Spring 2020
Lecture 4½ – An Interview Question
Sorted Matrix Search

Problem Description

Given a matrix $M$ (of size $m \times n$), where every row and every column is sorted, find out whether a given number $x$ is in the matrix.
Sorted Matrix Search

Given a sorted matrix $M$ (of size $m \times n$), where every row and every column is sorted, find out whether a given number $x$ is in the matrix.

(darker color means larger)
Sorted Matrix Search

Given a sorted matrix $M$ (of size $m \times n$), where every row and every column is sorted, find out whether a given number $x$ is in the matrix.

(One) **Idea**: Trace the contour between the numbers $\leq x$ and $> x$ in each row to see if $x$ appears.
Sorted Matrix Search Code

Partial Invariant: $M[i,0], \ldots, M[i,j-1] < x \leq M[i,j], \ldots, M[i,n-1]$
- for each $i$, holds for exactly one $j$
- holds when we are in the right spot in row $i$
Sorted Matrix Search Code

Initialization: $i \leq j$

Partial Invariant: $M[i,0], ..., M[i,j-1] < x \leq M[i,j], ..., M[i,n-1]$

How do we get the invariant to hold with $i = 0$?

- no easy way to initialize it so the invariant holds
- we need to search...
Sorted Matrix Search Code

Initialization:  

New goal: \( M[0,0], \ldots, M[0,j-1] < x \leq M[0,j], \ldots, M[0,n-1] \)

- will need a loop to find \( j \)
- Loop invariant: \( x \leq M[0,j], \ldots, M[0,n-1] \)
  - weakening of the new goal
  - decrease \( j \) until we get \( M[0,j-1] \) to also hold
Sorted Matrix Search Code

Initialization:

\[
\begin{align*}
\text{int } i &= 0; \\
\text{int } j &= ? \\
\{ \text{Inv: } x &\leq M[i,j], ..., M[i,n-1] \} \\
\text{while } ( ?? ) & \\
\quad ?? & \\
\{ \text{M[i,0], ..., M[i,j-1] } < x &\leq M[i,j], ..., M[i,n-1] \} \\
\end{align*}
\]

What is the easiest way to make this hold initially?
Sorted Matrix Search Code

Initialization:

```c
int i = 0;
int j = n;
```

{{ Inv: $x \leq M[i,j], ..., M[i,n-1] \} }}

while ( ?? )

??

{{ $M[i,0], ..., M[i,j-1] < x \leq M[i,j], ..., M[i,n-1]$ }}
Sorted Matrix Search Code

Initialization:

\[
\begin{align*}
\text{int } i &= 0; \\
\text{int } j &= n; \\
\{\text{ Inv: } x \leq M[i,j], \ldots, M[i,n-1] \} \\
\text{while } ( \ ?\ ? \ ) & \quad \rightarrow \quad \text{When does the postcondition hold?} \\
\quad \{\text{ M[i,0], \ldots, M[i,j-1] < } x \leq M[i,j], \ldots, M[i,n-1] \} \\
\end{align*}
\]
Sorted Matrix Search Code

Initialization:

```c
int i = 0;
int j = n;
{{ Inv: x \leq M[i,j], ..., M[i,n-1] }}
while (j > 0 && x <= M[i,j-1])
    ??
{{ M[i,0], ..., M[i,j-1] < x \leq M[i,j], ..., M[i,n-1] }}
```
Sorted Matrix Search Code

Initialization:

```
int i = 0, j = n;
{{ Inv: x ≤ M[i,j], ..., M[i,n-1] }}
while (j > 0 && x <= M[i,j-1]) {
    ??
    j = j - 1;
}
{{ M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
```

What goes here?
Sorted Matrix Search Code

Initialization: 

```
int i = 0, j = n;
{{ Inv: x ≤ M[i,j], ..., M[i,n-1] }}
while (j > 0 && x <= M[i,j-1]) {
    ??
    j = j - 1;
}
{{ M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
```
Sorted Matrix Search Code

Initialization: int i = 0, j = n;

\{ Inv: x \leq M[i,j], ..., M[i,n-1] \}

while (j > 0 && x \leq M[i,j-1]) {

j = j - 1;

}\{ M[i,0], ..., M[i,j-1] < x \leq M[i,j], ..., M[i,n-1] \}
Sorted Matrix Search Code

Initialization:

```c
int i = 0;
int j = n;

{{ Inv: x ≤ M[i,j], ..., M[i,n-1] }}
while (j > 0 && x <= M[i,j-1])
    j = j - 1;

{{ M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
```
Sorted Matrix Search Code

That finds the right column in row 0

- can now check $M[0,j] = x$ (if $j < n$)
- if not, we can move onto the next row
  - $x$ cannot be anywhere in the row if it’s not at $M[i,j]$
  - set $i = i + 1$

Process continues in each row thereafter...
Sorted Matrix Search Code

• Make progress by setting $i = i + 1$
• When $i$ increases, the invariant may be broken
  – we have $x \leq M[i,j] \leq M[i+1,j]$ since columns are sorted
  – and $M[i+1,j] \leq M[i +1,j+1], \ldots, M[i +1,n-1]$ since rows are sorted
  – so we get $x \leq M[i +1,j], \ldots, M[i +1,n-1]$
Sorted Matrix Search Code

- Make progress by setting $i = i + 1$
- When $i$ increases, the invariant may be broken
  - we have $x \leq M[i+1,j], \ldots, M[i+1,n-1]$
  - may need to restore invariant for $M[i,0], \ldots, M[i,j-1] < x$
  - decrease $j$ until it holds again...
    - when have we seen this before?
    - initialization
Sorted Matrix Search Code

- Make progress by setting $i = i + 1$
- When $i$ increases, the invariant may be broken
  - we have $x \leq M[i+1,j], \ldots, M[i+1,n-1]$
  - may need to restore invariant for $M[i,0], \ldots, M[i,j-1] < x$
  - could copy and paste the same loop
    - or you can do it with one copy

Don’t try this at home!
**Sorted Matrix Search Code**

**instead of**

```c
int i = 0, j = n;
[move j left]
{{ Inv: M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
while (i != n) {
    i = i + 1;
    [move j left]
}
```

**we can write**

```c
int i = 0, j = n;
while (i != n) {
    [move j left]
    {{ M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
    i = i + 1;
}
```
int i = 0;
int j = n;

while (i != n) {
    {{ Inv: \(x \leq M[i,j], \ldots, M[i,n-1]\) }}
    while (j > 0 && x <= M[i,j-1])
        j = j - 1;
    {{ M[i,0], \ldots, M[i,j-1] < x \leq M[i,j], \ldots, M[i,n-1] }}
    if (j < n && x == M[i,j])
        return true;
    i = i + 1;
}
return false;
Sorted Matrix Search Code

```c
int i = 0;
int j = n;
{{ Inv: x not in M[k,l] for k < i and x ≤ M[i,j], ..., M[i,n-1] }}
while (i != n) {
    {{ Inv: x not in M[k,l] for k < i and x ≤ M[i,j], ..., M[i,n-1] }}
    while (j > 0 && x <= M[i,j-1])
        j = j - 1;
    {{ x not in M[k,l] for k < i and M[i,0], ..., M[i,j-1] < x ≤ M[i,j], ..., M[i,n-1] }}
    if (j < n && x == M[i,j])
        return true;
    i = i + 1;
}
return false;
```