Developing array algorithms. Reading: 8.3..8.5

Prelim Tuesday, 7:30PM. Olin 155 and 255
Review session, Sunday 1–3. Phillips 101
Handout describes what will be covered.

have iclickers ready for quiz

Two-dimensional arrays

A 2-dimensional array

b
P00 P01 P02 P03
P04 P05 P06 P07
P08 P09 P10 P11
P12 P13 P14 P15
P16 P17 P18 P19
P20 P21 P22 P23

Same array in row-major order:

c
P00 P01 P02 P03
P04 P05 P06 P07
P08 P09 P10 P11
P12 P13 P14 P15
P16 P17 P18 P19
P20 P21 P22 P23

You can see that

b[0][j]
is same as

c[0 * (no of columns) + j]
b[1][j]
is same as

c[1 * (no of columns) + j]
and in general:

b[i][j]is same as

c[i * (no of columns) + j]

Pixel (picture element): 4 components, each in 0..255
Contains: alpha component (we never change it)
green component g
blue component b

/** DM provides functions for extracting components of a pixel. */
public static final DirectColorModel DM=
(DirectColorModel)ColorModel.getRGBdefault();

Procedure invert has in it:

DM.getRed(pixel);

alpha red green blue
8 bits 8 bits 8 bits 8 bits
(all 4 components fit into an int
(alpha << 24) (red << 16) (green << 8) | blue)

Partition algorithm: Given an array b[h..k] with some value x in b[h]:
P: b

< x

>= x

Swap elements of b[h,k] and store in j to truthify P:

Q: b

<= x

<= x

change:
b
3 5 4 1 6 2 3 8 1

into:
b
1 2 3 4 6 3 8

or:
b
1 2 3 4 5 6 8

x is called the pivot value.
x is not a program variable; x just denotes the value initially in b[h].
Partition algorithm:
Given an array \( b[h..k] \) with some value \( x \) in \( b[h] \):

\[
\begin{align*}
P: \ b & \leq x \\
Q: \ b & = x \quad \text{and} \quad i = \min(h+1..k)
\end{align*}
\]

You generalize \( P \) and \( Q \) to create a loop invariant

\( x \) is called the pivot value.

\( x \) is not a program variable; it just denotes the value initially in \( b[h] \).

Binary search: Vague spec: Look for \( v \) in \( \text{sorted array} \ b[h..k] \).

- **Invariant as picture:** Generalizing pre- and post-condition
- **Finding the minimum of an array.** Given array \( b \) satisfying

\[
\begin{align*}
P: \ b & = \min(h..k) \\
Q: \ b & = \min(0..n) (\text{values in } 0..n \text{ are unknown})
\end{align*}
\]

Precondition P: \( b[h..k] \) is sorted (in ascending order).

Store in \( i \) to truthify:

\[
\begin{align*}
P: \ b & = \min(h..k) \\
Q: \ b & = \min(0..n) (\text{values in } 0..n \text{ are unknown})
\end{align*}
\]

Vague spec.: Find first occurrence of \( v \) in \( b[h..k-1] \).

Better spec.:

\[
\begin{align*}
P: \ b & = \min(h..k) \\
Q: \ b & = \min(0..n) (\text{values in } 0..n \text{ are unknown})
\end{align*}
\]

Linear search

Vague spec.: Find first occurrence of \( v \) in \( b[h..k-1] \).

Better spec.: Store an integer in \( i \) to truthify postcondition \( Q \):

\[
\begin{align*}
P: \ b & = \min(h..k) \\
Q: \ b & = \min(0..n) (\text{values in } 0..n \text{ are unknown})
\end{align*}
\]

Reversal: Reverse the elements of array segment \( b[h..k] \).

\[
\begin{align*}
\text{precondition P:} & \quad b & \text{not reversed} \\
\text{postcondition Q:} & \quad b & \text{reversed.}
\end{align*}
\]

Change:

\[
\begin{align*}
\text{Change:} & \quad b & 1234567899999 \\
\text{into} & \quad b & 999987654321
\end{align*}
\]