COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Erase an ellipse!

Two things:

      if a key is pressed

      make a function to find the first ellipse under the mouse

      delete the ellipse!

Update the draw

COMPLEXITY!

Compare the two techniques…

Second is more flexible (can delete / insert anywhere)

First is more efficient (erasing the array, inserting..)

Core component of computer science!

Extension – automatically grow the array

If the array is full, then double the size.

How can we grow an array?

      can't.

Instead, make a function that..

      takes array as parameter

      makes a new array (double size)

      copies data over

      sets additional bins to empty

      returns new array.

searching

searching…

note: searching, in general, is a fundamental problem of computer science. It is difficult, it is slow, it is everywhere:

a person logs in: does their password match?

you shoot a bad guy: do you hit them?

you load a website…

you ask google something…

google is fast!

linear search versus binary search

two methods for searching an array for a given value: one is **much** faster than the other

note: speed matters!

linear search – search a stack of papers

simple: check every element until you find it. Order doesn't matter

if element not in array?

you must check every element first to be sure!

worst case: have to look at every element!

best case: first element is what we want, 1 check

average case: search half way through and find

Example: lotto tickets

Let's do a lotto search – hand out 1 million random but unique tickets, draw a ticket, and see if that ticket exists.

Ticket generation method:

make an array – 1 bin per ticket

go left to right in the array

add a random increase from the prev ticket

place the ticket in the array

int[] tickets | 1 | 2 | 5 | 8 | 9 |

Make the function:

handOutTickets –

     takes an array and fills with tickets, with random space between them (up to 3)

     returns the largest ticket number so we know what range is possible.

In setup, hand out the tickets.

implement a method to perform linear search

function: take an array and a target, return the index of the target if found

aside:: what to return if not found?

-1 is a good default because it is outside the range of possible valid values. we want to return the index into an array, and -1 is impossible. be careful with choosing a value that could be valid, e.g., 5000, 9999, or 0.

Let's visualize

Draw a number line

Each time a number is checked, draw a tick mark

Ticket: 11567172 found!

Globals and functions

Left of line, line width, tick height

drawLine (at a specific y)

drawTick (at a specific y)

- waayy more bins than pixels, need to scale)

Setup our draw:

Clear background

Generate a random ticket!

Do the linear search (and save the result)

Print out text telling us the result

--- need to update linear search to draw tick marks at bins searched.

structured data

if we can assume that data is structured in some sort of helpful way, we can often leverage this for searching

searching for a page in a book

> does it make sense to start at p1 and check every one along the way? e.g., find page 280

> **we assume structure** and leverage this for more efficient searching

structured data – binary search

if we assume that an array is already sorted smallest to largest, we can perform binary search

the key to binary search is that we can ignore whole sections of the array without checking them, because we can make assumptions about it.
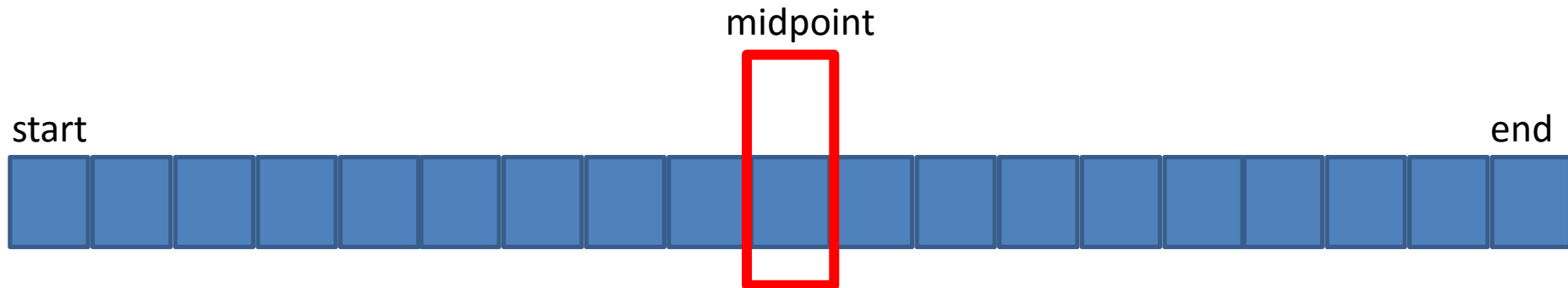
binary search

- assume the data array is already sorted

- look half way in the middle and compare to target (what you're searching for)

- if smaller or larger, we can discount half the data instantly!

- in each step, we can cut the remaining data in half – much more efficient than linear search

smallest

25

largest

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

target: 15

binary search algorithm

repeat:

      find the midpoint between the start and end

      location is (start index + end index) / 2

      test data at midpoint against target

midpoint

start

end

binary search algorithm

repeat:
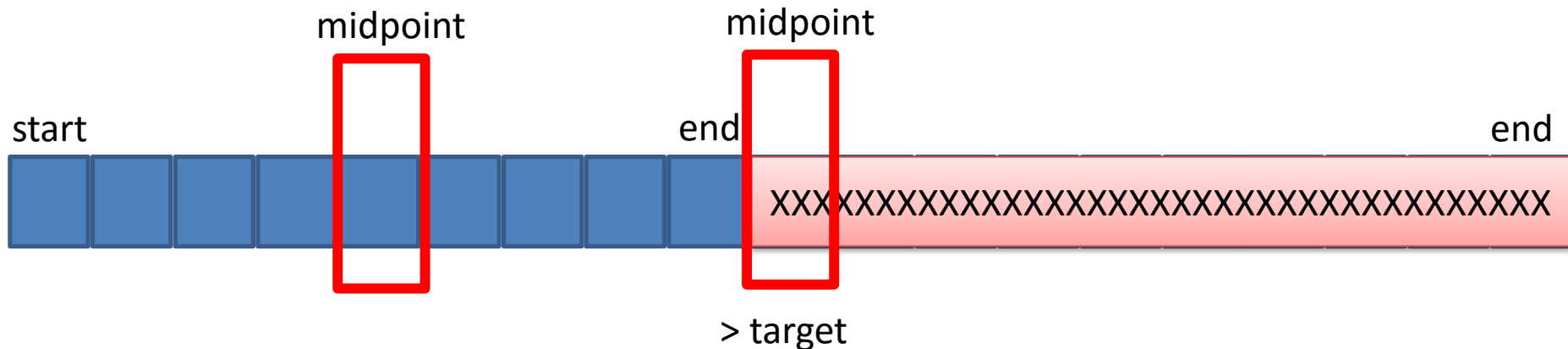
find the midpoint between the start and end

location is (start index + end index) / 2

test data at midpoint against target

if data > target, cut right half
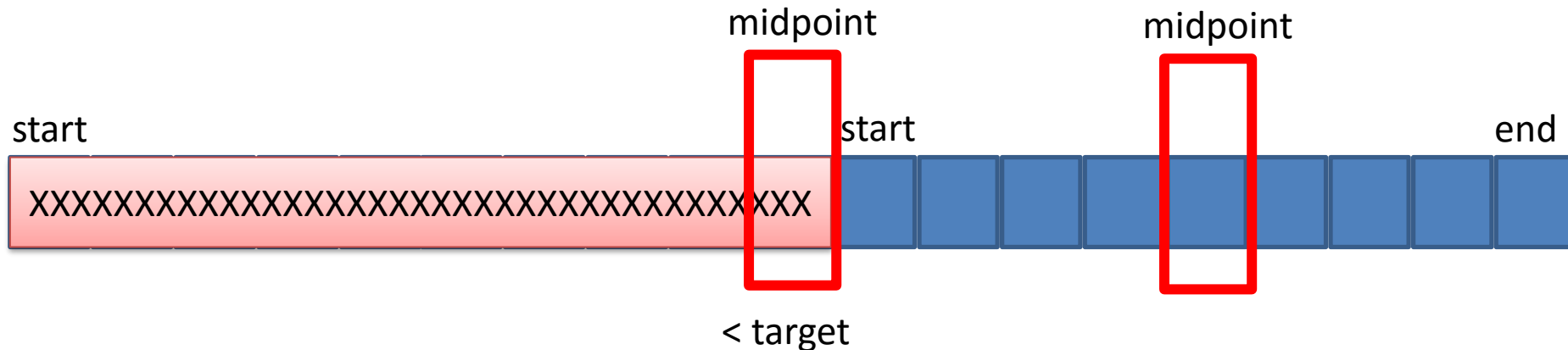
end = midpoint - 1

binary search algorithm

repeat:

find the midpoint between the start and end

location is (start index + end index) / 2

test data at midpoint against target

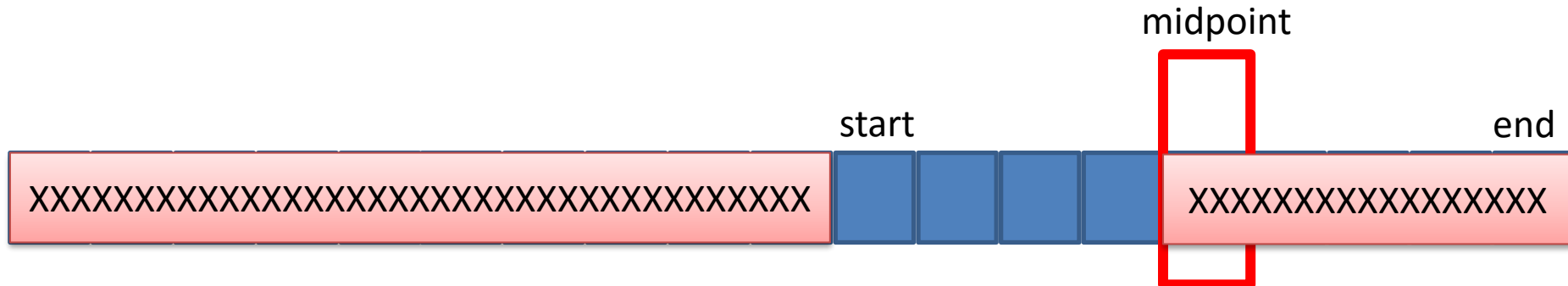if data < target, cut left half

start = midpoint + 1

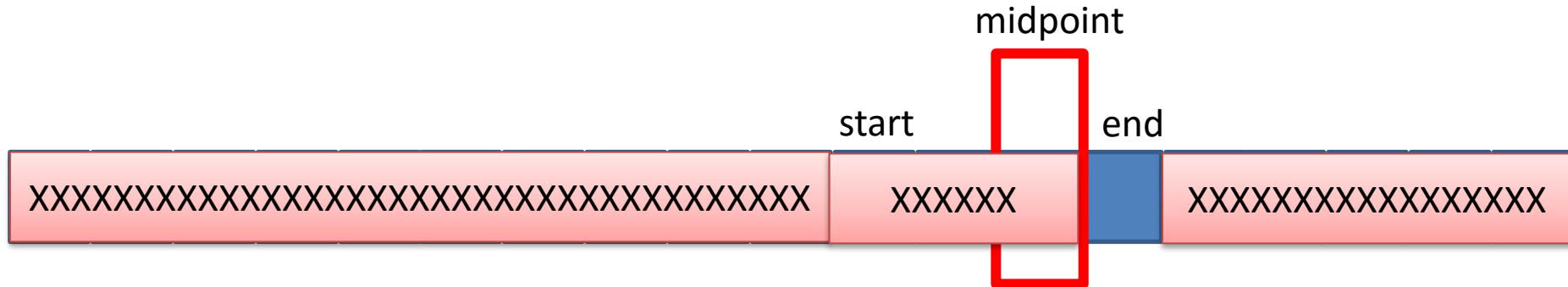midpoint   midpoint

start   start   end

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

< target

binary search algorithm – end case

a) we find the target

b) there is no target:

midpoint

start

end

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXX

binary search algorithm – end case

a) we find the target

b) there is no target:

midpoint

start

end

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXX XXXXXXXXXXXXXXXXXXXX

binary search algorithm – end case

a) we find the target

b) there is no target:

midpoint

start
end

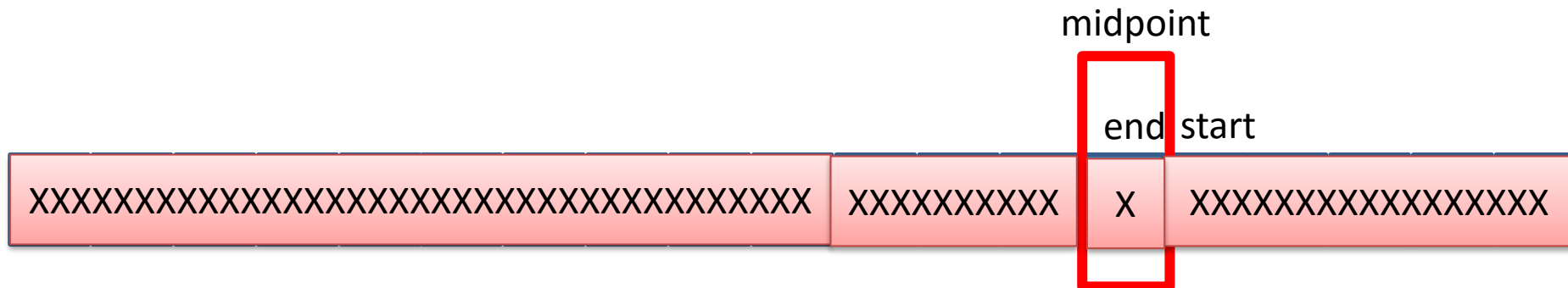XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXX   X   XXXXXXXXXXXXXXXXXXX

binary search algorithm – end case

a) we find the target

b) there is no target:

eventually start > end, (or end < start, same thing)

midpoint

end start

| XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | XXXXXXXXX | X | XXXXXXXXXXXXXXXXXX |

Implement and play with it!
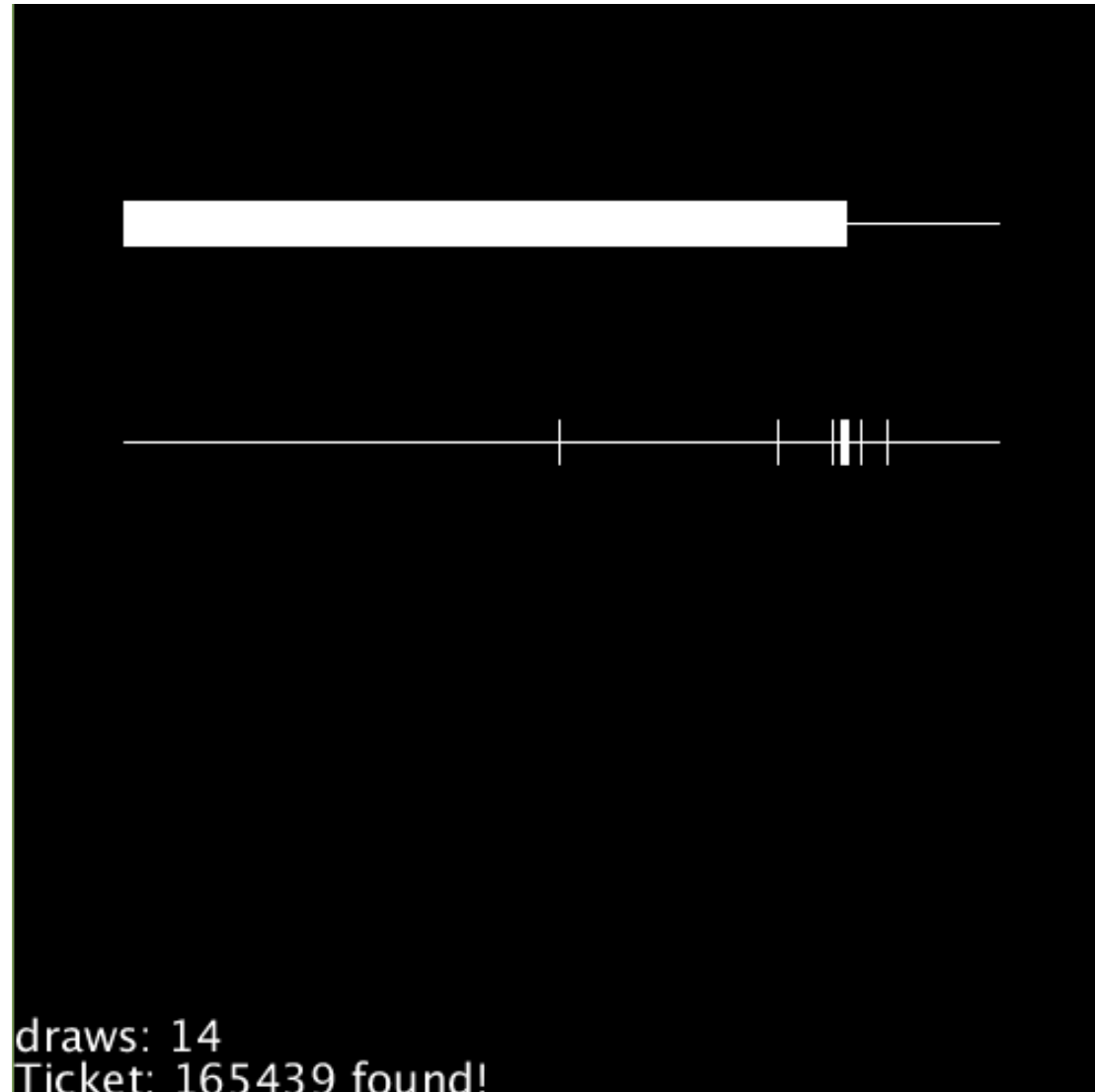
search – how much work?

Algorithm	Data Size	Best Case	Worst Case	10 elements	1,000 elements	1,000,000,000
Linear Search	n	1 check	n checks	10	1000	1,000,000,000

1,000,000,000,000,000,000,000,000 ?
Linear search: 1 check per cycle, 1GHz pc: 10 million years
Binary search: 80 checks

If you stay in computer science, a key theme becomes *algorithms* like these, and specific *data structures* like a sorted array, all of which help manage large amounts of information and information queries.

WELCOME

to the end of the course!

in this course we learned Processing

but programming is programming, and the concepts you learn transfer to other languages

note: Pseudo-code: computer code that describes an algorithm in programming terms, but is not necessarily a specific langauge.

I bet you can read other languages pretty easily

pseudocode

for j ←1 to length(A)-1
   key ← A[j]
   i ← j - 1
   while i >= 0 and A [i] > key
      A[i +1] ← A[i]
      i ← i -1
   A [i +1] ← key

```pascal
var
  endPoint, startPoint, midPoint, target, indexFound : Integer;
  data : Array[0..11] of Integer = (1,1,1,1,2,4,5,5,10,90,100,1000);

Begin
 startPoint := 0;
 endPoint := Length(data)-1;
 target := 101;
 indexFound := -1;

 while (startPoint <= endPoint) AND (indexFound = -1) do
 begin
   midPoint := Floor((startPoint + endPoint) /2);

   if data[midPoint] = target then
     indexFound := midpoint
   else if data[midPoint] < target then
     startPoint := midPoint+1
   else
     endPoint := midPoint -1;
 end;
 writeln(indexFound);
End.
```