MC140: lecture #24

today's topic:

searching:
linear search
binary search

searching.

- last class, we talked about searching *unsorted* arrays using a linear search
- today we'll talk about searching *sorted* arrays
- we'll discuss two techniques for searching sorted arrays:
  - modified linear search
  - binary search

linear search.

- first, here's the basic linear search from last time
- it works on unsorted arrays
- it also works on sorted arrays, but is less efficient on sorted arrays than the modified linear search that follows -- when the key being searched for is not in the array being searched

basic linear search.

```
#include <stdio.h>

int linearSearch( int dice[], int size, int key );

int main( void ) {
  int dice[5] = { 3,4,6,5,2,1 };
  int k = linearSearch( dice,6,2 )
  /* in this case, k will be 4, since 2 is stored in the 4th location
     in the dice array */
} /* end of main() */

/* this function returns the index of "key" in the "dice" array.
   it returns -1 if "key" is not found in the "dice" array. */
int linearSearch( int dice[], int size, int key ) {
  int i;
  for ( i=0; i<size; i++ ) {
    if ( dice[i] == key ) {
      return( i );
    } /* end if */
  } /* end for i */
  return( -1 );
} /* end linearSearch() */
```


modified linear search.

```
#include <stdio.h>

int linearSearch2( int dice[], int size, int key );

int main( void ) {
  int dice[5] = { 1,2,3,4,5,6 };
  int k = linearSearch2( dice,6,2 );
} /* end of main() */

/* this function returns the index of "key" in the "dice" array.
   it returns -1 if "key" is not found in the "dice" array.
   it assumes the dice array is sorted in ascending order. */
int linearSearch2( int dice[], int size, int key ) {
  int i = 0;
  while (( i < size ) && ( key > dice[i] )) {
    i++;
  } /* end while */
  if ( key == dice[i] ) {
    return( i );
  } /* end if */
  else {
    return( -1 );
  } /* end else */
} /* end linearSearch() */
```


binary search.

- binary search is much more efficient than linear search, on a sorted array
- it takes the strategy of continually dividing the search space into two halves, hence the name "binary"
- remember, binary search ONLY works on sorted arrays

binary search, 2.

- here's how binary search works
- say you are searching something very large, like the phone book
- if you are looking for one name (e.g., "Gilligan"), it is extremely slow and inefficient to start with the A's and look at each name one at a time, stopping only when you find "Gilligan"
- but this is what linear search does

binary search, 3.

- binary search acts much like you'd act if you were looking up "Gilligan" in the phone book
- you'd open the book somewhere in the middle, then determine if "Gilligan" appears before or after the page you have opened to
- if "Gilligan" appears after the page you've selected, then you'd open the book to a later page
- if "Gilligan" appears before the page you've selected, then you'd open the book to an earlier page
- you'd repeat this process until you found the entry you are looking for

binary search, 4.

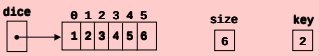
```
/* this function returns the index of "key" in the "dice" array.
   it returns -1 if "key" is not found in the "dice" array.
   it assumes the dice array is sorted in ascending order. */
int binarySearch( int dice[], int size, int key ) {
  int lo = 0, hi = size-1, mid;
  while ( lo <= hi ) {
    mid = ( lo + hi ) / 2;
    if ( key == dice[mid] ) {
      return( mid );
    } /* end if */
    else if ( key < dice[mid] ) {
      hi = mid - 1;
    } /* end else if */
    else /* key > dice[mid] */
      lo = mid + 1;
    } /* end else */
  } /* end while */
  return( -1 );
} /* end of binarySearch() */
```


binary search: sample run.

- suppose main() looks like this:

```
int main( void ) {
  int dice[5] = { 1,2,3,4,5,6 };
  int k = linearSearch2( dice,6,2 );
} /* end of main() */
```

- inside binarySearch(), it goes like this:



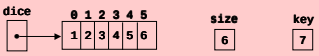
lo	hi	mid	dice[mid]	key	action
0	5	(0 + 5)/2 = 2	3	2	hi=mid-1; loop again
0	1	(0 + 1)/2 = 0	1	2	lo=mid+1; loop again
1	1	(1 + 1)/2 = 1	2	2	return 1

binary search: another sample run.

- what happens if the key is not in the array?
- suppose the call is:

```
k = binarySearch( dice,6,7 );
```

- inside binarySearch(), it goes like this:



lo	hi	mid	dice[mid]	key	action
0	5	(0 + 5)/2 = 2	3	7	lo=mid+1; loop again
3	5	(3 + 5)/2 = 4	5	7	lo=mid+1; loop again
5	5	(5 + 5)/2 = 5	6	7	lo=mid+1; loop again
6	5				exit loop, since 6>5, return -1

reading.

- DD 6.9
- Wed 4th: Prof Yanco, recursion
- Fri 6th: Prof Muller, recursion
- Mon 9th: I'm back! Exam review
- EXAM #3 will be on WED 11 APRIL
- assignment #8 will come after the exam
- for the exam, review the 4 sorting algorithms we've been discussing, and the 3 searching algorithms here!