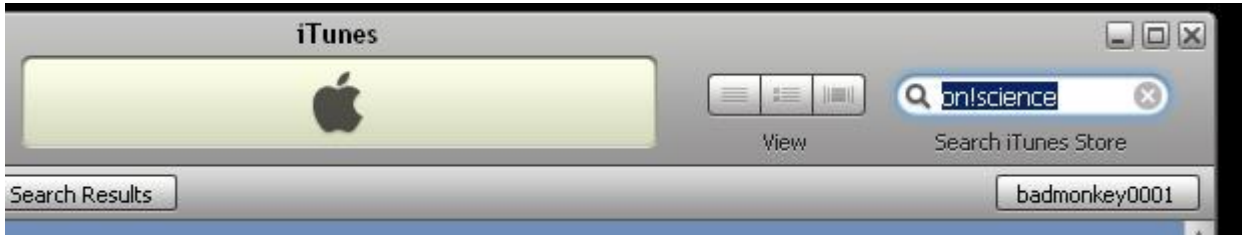
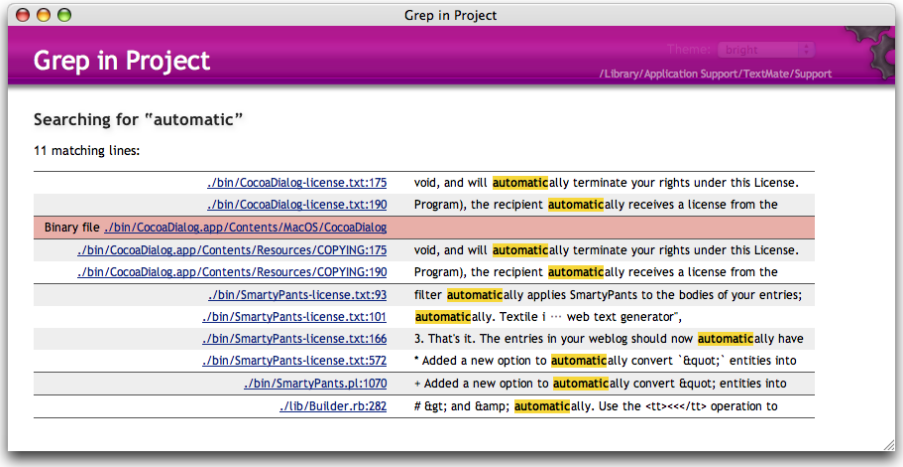
Searching

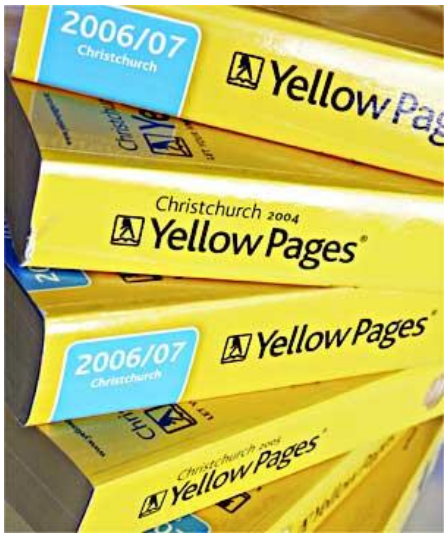

iTunes
bn!science
View
Search iTunes Store
Search Results
badmonkey0001


Google
recursive backtracking
Google Search
I'm Feeling Lucky
Advanced Search
Preferences
Language Tools


Grep in Project
Grep in Project
/Library/Application Support/TextMate/Support
Searching for "automatic"
11 matching lines:
./bin/CocoaDialog-license.txt:175 void, and will automatically terminate your rights under this License.
./bin/CocoaDialog-license.txt:190 Program), the recipient automatically receives a license from the
Binary file ./bin/CocoaDialog.app/Contents/MacOS/CocoaDialog
./bin/CocoaDialog.app/Contents/Resources/COPYING:175 void, and will automatically terminate your rights under this License.
./bin/CocoaDialog.app/Contents/Resources/COPYING:190 Program), the recipient automatically receives a license from the
./bin/SmartyPants-license.txt:93 filter automatically applies SmartyPants to the bodies of your entries;
./bin/SmartyPants-license.txt:101 automatically. Textile i ··· web text generator",
./bin/SmartyPants-license.txt:166 3. That's it. The entries in your weblog should now automatically have
./bin/SmartyPants-license.txt:572 * Added a new option to automatically convert `&quot;` entities into
./bin/SmartyPants.pl:1070 + Added a new option to automatically convert " entities into
./lib/Builder.rb:282 # > and & automatically. Use the <tt><<</tt> operation to

Linear Search

- Given a list of data, find the location of a particular value or report that value is not present
- Linear search
  - intuitive approach:
    - start at first item
    - is it the one I am looking for?
    - If not, go to next item
    - repeat until found or all items checked
- If items not sorted or unsortable this approach is necessary



Linear Search Code

```
/*
      return the index of the first occurrence
      of target in list, or -1 if target not present in
      list
*/
public int linearSearch(int list[], int target)
{
    int i = 0;
    while(i < list.length && list[i] != target)
       i++;

    if(i >= list.length)
       return -1;
    else
      return i;
}
```

Question 1

- What is the average case Big-O of **linear search** in an array with *n* items, if an item is present?

A. O(n)

B. $O(n^2)$

C. O(1)

D. O(log(n))

E. O(n log(n))

Question 1

- What is the average case Big-O of **linear search** in an array with *n* items, if an item is present?

A. O(n)

B. $O(n^2)$

C. O(1)

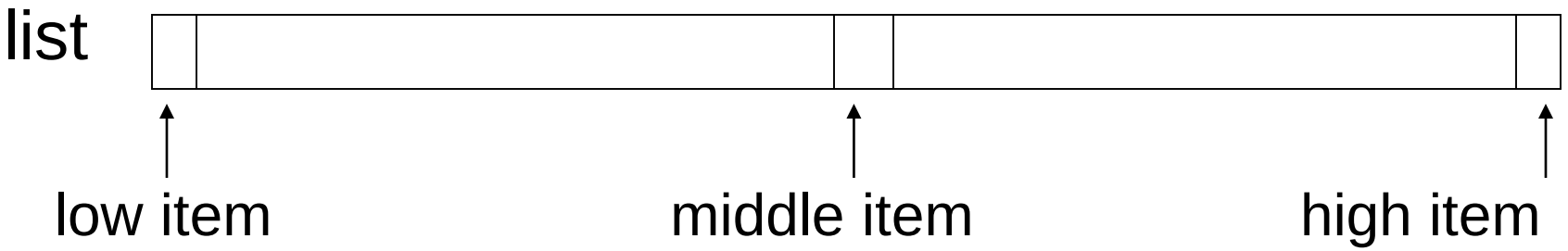
D. O(log(n))

E. O(n log(n))

Searching in a Sorted List

- If items are sorted, we can *divide and conquer*
- Dividing your work in half with each step
  - Generally a good thing
- The Binary Search on list in ascending order
  - start at middle of list
    - is that the item?
    - if not, is it less than or greater than the item?
      - less than, move to second half of list
      - greater than, move to first half of list
  - repeat until found or sub-list size = 0

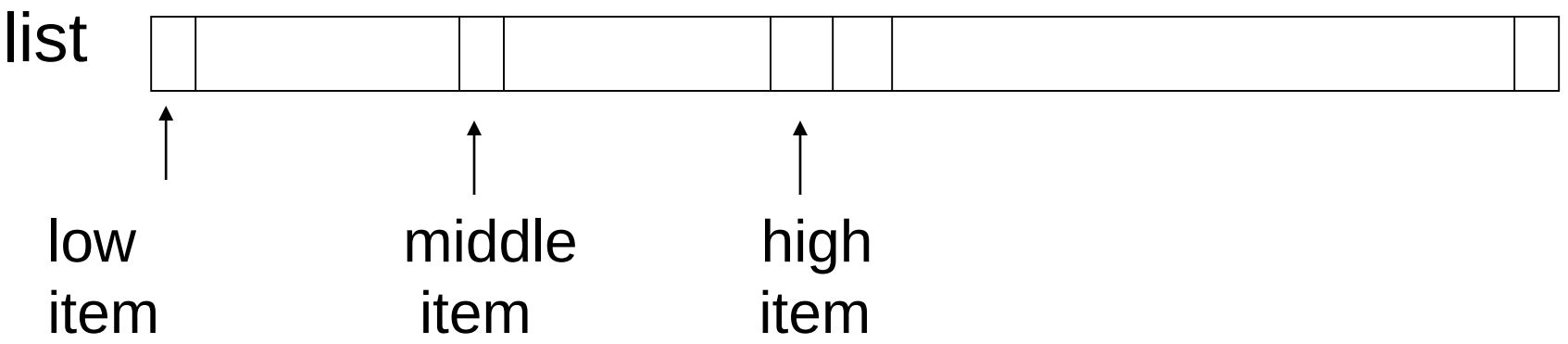
Binary Search



Is middle item what we are looking for?

If not, is it more or less than the target?

If lower…



and so forth…

Recursive Binary Search

```
public static int search(int list[], int target)
{
    return b-search(list, target, 0, list.length – 1);
}

public static int b-search(int list[], int target,
                                    int low, int high)
{
    if( low <= high )
    {
        int mid = low + ((high - low) / 2);
         if( list[mid] == target )
            return mid;
        else if( list[mid] > target )
            return b-search(list, target, low, mid – 1);
        else
            return b-search(list, target, mid + 1, high);
    }
    return -1;
}
```

Question 2

- What is the worst case Big O of **binary search** in an array with *n* items, if an item is present?

  A. O(n)

  B. $O(n^2)$

  C. O(1)

  D. O(log(n))

  E. O(n log(n))

Question 2

- What is the worst case Big O of **binary search** in an array with *n* items, if an item is present?

A. O(n)

B. $O(n^2)$

C. O(1)

D. O(log(n))

E. O(n log(n))

Other Searching Algorithms

- Interpolation Search
  - more like what people really do
- Binary Search Trees
- Hash Table Searching
- Best-First
- A*