NPTEL MOOC,JAN-FEB 2015
Week 2, Module 2

DESIGN AND ANALYSIS OF ALGORITHMS

Searching in an array

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Search problem

- Is a value K present in a collection A?
- Does the structure of A matter?
  - Array vs list
- Does the organization of the information matter?
  - Values sorted/unsorted

The unsorted case

```
function search(A,K)

  i = 0;

   while i < n and A[i] != K do
     i = i+1;

   if i < n
     return i;
   else
     return -1;
```

Worst case

- Need to scan the entire sequence A
  - O(n) time for input sequence of size A
- Does not matter if A is array or list

Search a sorted sequence

- What if A is sorted?
  - Compare K with midpoint of A
  - If midpoint is K, the value is found
  - If K < midpoint, search left half of A
  - If K > midpoint, search right half of A
- Binary search

Binary search ...

```
bsearch(K,A,l,r)  // A sorted, search for K in A[l..r-1]

if (r - l == 0) return(false)

mid = (l + r) div 2  // integer division

if (K == A[mid]) return (true)

if (K < A[mid])

  return (bsearch(K,A,l,mid))

else

  return (bsearch(K,A,mid+1,r))
```

Binary Search …

- How long does this take?
  - Each step halves the interval to search
  - For an interval of size 0, the answer is immediate
- T(n): time to search in an array of size n
  - $T(0) = 1$
  - $T(n) = 1 + T(n/2)$

Binary Search …

- T(n): time to search in a list of size n
  - T(0) = 1
  - T(n) = 1 + T(n/2)
- Unwind the recurrence
  - $T(n) = 1 + T(n/2) = 1 + 1 + T(n/2^2) = \ldots$
    $= 1 + 1 + \ldots + 1 + T(n/2^k)$
    $= 1 + 1 + \ldots + 1 + T(n/2^{\log n}) = O(\log n)$

Binary Search …

- Works only for arrays
    - Need to be look up A[i] in constant time
- By seeing only a small fraction of the sequence, we can conclude that an element is not present!