NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 3, Lecture 4

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Sequences of values

- Two basic ways of storing a sequence of values
  - Arrays
  - Lists
- What’s the difference?

Arrays

- Single block of memory, elements of uniform type
  - Typically size of sequence is fixed in advance
- Indexing is fast
  - Access `seq[i]` in constant time for any `i`
  - Compute offset from start of memory block
- Inserting between `seq[i]` and `seq[i+1]` is expensive
- Contraction is expensive

Lists

- Values scattered in memory
  - Each element points to the next—“linked” list
  - Flexible size
- Follow i links to access `seq[i]`
  - Cost proportional to `i`
- Inserting or deleting an element is easy
  - “Plumbing”

Operations

- Exchange `seq[i]` and `seq[j]`
  - Constant time in array, linear time in lists
- Delete `seq[i]` or Insert `v` after `seq[i]`
  - Constant time in lists (if we are already at `seq[i]`)
  - Linear time in array
- Algorithms on one data structure may not transfer to another
  - Example: Binary search

Search problem

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

The unsorted case

```
def search(seq,v):
  for x in seq:
    if x == v:
      return(True)
  return(False)
```

Worst case

- Need to scan the entire sequence `seq`
  - Time proportional to length of sequence
- Does not matter if `seq` is array or list

Search a sorted sequence

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

Binary search ...

```
def bsearch(seq,v,l,r):
// search for v in seq[l:r], seq is sorted

  if r - l == 0:
    return(False)

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

  if v == seq[mid]:
    return (True)

  if v < seq[mid]:
    return (bsearch(seq,v,l,mid))
  else:
    return (bsearch(seq,v,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 look up `seq[i]` in constant time
- By seeing only a small fraction of the sequence, we can conclude that an element is not present!

Python lists

- Are built in lists in Python lists or arrays?
- Documentation suggests they are lists
  - Allow efficient expansion, contraction
- However, positional indexing allows us to treat them as arrays
  - In this course, we will “pretend” they are arrays
  - Will later see explicit implementation of lists