Computing for Math and Stats

Lecture 20

Checking for Sorted

- Sorted arrays are very important
  - Useful on their own right
  - Make searching in them **much** faster
  - Many of the techniques used in sorting arrays are used in other situations too.
- We can check if an array is sorted with a for-loop
  - Check if every element is smaller than the next.
  - Assume (WLoG) ascending order
  - See isunsorted.m

Sorting

- We are given an array of integers
- We have to give back an array in which the integers are sorted
- All the integers in the original array have to still be there
- If two integers are equal
  - We leave them in the order we found them (stable sorting)
  - We leave them in arbitrary order (unstable)
- The above makes sense if there is data associated with each integer (eg sort students by their Lab A mark)

Sorting

- There are many (too many) algorithms for sorting
  - Some good for small arrays
  - Some good for large arrays
  - Some good for huge arrays
  - Some require extra memory, others in-place
  - Some stable, some not.
  - Some fast on average, some fast on worst case
- We look at one of the simplest

Simplest Sorting

- This algorithm is known as bubble-sort
- Scan the array several times
  - If an element is greater than the next swap them
- Eventually the array will be sorted
- This algorithm is good for
  - Small arrays
  - Almost sorted arrays
- See mybubble.m

Improvements to Sorting

- We can iterate enough times to guarantee that it is sorted and stop early if we detect it is sorted.
- After k iterations the last k elements are sorted
  - No need to scan all the way to the end

Run time

- It is important to be able to estimate the running time of an algorithm
- Usually the running time is given as a function of the size of the input
- We usually care for average and worst times.
- Most often we care only for an approximate running time
  - It is called complexity

Complexity

- If an algorithm needs 3.2*N^2+1.1*N steps to complete
  - The complexity is O(N^2)
- It is read “big O” or “order of”
- Bubblesort is O(N^2)
- The best sorting algorithms are O(N*logN)
- It can be shown that this is the optimal

Searching

- We are given an array of integers and an integer
- We have to answer if this integer is in the array
- Can be more complex, but we do not deal with such things here

Searching

- Simple linear search
  - Works in sorted or unsorted arrays
- Faster on sorted arrays because we can stop early if it is not there
- Binary search is much faster
- Works only on sorted arrays

Binary Search

- The basic idea
  - Go to the middle of the array
  - If the query is there report “true”
  - If the query is bigger than the middle element
    - Search in the bottom half
  - If smaller
    - Search in the top half
- Needs O(log N) steps
- See mybinsrch.m

Things to do at home

- Rewrite the bubblesort to stop early if the array is sorted
- Rewrite the binary search algorithm so that it returns the index and the largest value in the array that does not exceed the target value. If all values in the array exceed the target returns 0 for index. For example if the array is [1, 2, 4, 5] and the target is 3 it returns [2, 2].
- Rewrite the bubblesort so that the scanning of the array is from the bottom up, so that if there is a new element added at the end of a mostly sorted array the algorithm will end quickly.