SEARCHING

Today

- Last time we considered algorithms for sorting a list of objects.
- This lecture will look at how one *searches* through a list to see if it contains something we are looking for.
- We will consider a couple of approaches to searching an array, and look at how efficient they are.

Searching

- Sorting a list of things is one very common thing to want to do.
- Another common operation is searching to see if something is in a list.
- The simplest way to do this is to look through the list, item by item.
- We can search a list of numbers using:

```
for(counter = 0; counter < 6; counter++)
  {
     if(numbers[counter] == numberWeWant)
        {
          cout << ''We found it'' << endl;
        }
  }
```

- We'll call this *linear search*.

- This isn't very efficient.
- In the worst case, it means we have to look through every element of the list to find if the number is in there.
- That is fine if the list is 6 elements long, but not so fine if it is a million elements long.
- Much more efficient is *binary search*, though binary search only works if the list is sorted.

Binary search

- Binary search works as follows, assuming the list is sorted smallest first.
- Look at the middle element of the list.
- If it is the one we are looking for, we are done.
- If the middle element is larger than the one we are looking for, then the one we are looking for must be in the first half of the list (if it is in the list).
- If the middle element is smaller than the one we are looking for, then the one we are looking for must be in the second half of the list (if it is in the list).
- Repeat in the relevant half of the list

- The code for doing this is:

```
int mid, start = 0;
int stop = size;

while(start <= stop){
  mid = (start + stop)/2;

  if(want == a[mid]){
    return true;
  }

  if(want > a[mid]){
    start = mid + 1;
  }

  if(want < a[mid]){
    stop = mid - 1;
  }
}

return false;
}
```


Analysis of sorting algorithms

- If you try binary and linear search out on some examples, you will see that binary search usually finds the result (that the thing we want is in or out of the list) quicker than linear search.
- (If you run the example `search.cpp` from the course website you can see exactly how much better binary search is as you search for different numbers.)
- However, as for sorting, we can say more precisely what the efficiency of the algorithms is.

- We consider how many comparisons we will have to do for a list that holds N elements,
- For linear search, the worst thing that can happen is that we have to look at all N elements.
- So the *worst case* complexity is N.
- However, often we will look at less if the thing we are looking for is earlier in the list).
- On average we will end up looking at $\frac{N}{2}$ elements and so we say that the *average case* complexity is $\frac{N}{2}$.
- In binary search we will have to look at $\log_2(n+1)$ elements in the worst case.

- We can look at the worst case number of comparisons for different values of N.

N	Linear	Binary
100	100	7
1,000	1,000	10
1,000,000	1,000,000	20

- So we can see that binary search is a lot more efficient than linear search as the size of the list increases.
- However, to use binary search, we need a sorted list.

- Remembering Big-O notation and the results from the last class, we can say the following.
- Linear searching is $O(N)$, and so will be more efficient than linear sorting since N is always smaller than N^2.
- Binary searching is $O(\log N)$, and so is more efficient than either linear searching or linear sorting since $\log N$ is smaller than N and N^2.
- However, if we sort with linear sort and and then search using binary search, overall that will be less efficient than using linear search.
- Note that there are other algorithms that sort more efficiently than any of the sorting algorithms we have looked at.

Genetic algorithms

- The program `ga.cpp` which you can download from the course website is an example of the use of bubblesort.
- It is also an example of *biologically inspired computing*, where ideas from biology are used to make computer programs more efficient.
- In *genetic algorithms* we breed solutions to a computing problem, and allow them to evolve until we have the best solution.
- Genetic algorithms can be a very efficient way to find solutions to some problems.

Summary

- We looked at two forms of searching an array:
  - Linear search
  - Binary search
- We considered the complexity of both forms of search.
- We also talked a bit about genetic algorithms.