

- Theory of Computation
 - Big-Oh (and its “relatives”)
 - Tractable vs intractable
 - Deterministic vs non-deterministic
 - P vs NP vs NP-complete

These slides were based on those of C Johnson, E McCreath and W Liang.

Searching

- Fact:
 - The problem of determining whether a sorted list of n items contains a particular item is $O(\log n)$
 - Why? - because we know about *Binary Search*
- Fact:
 - Searching a sorted list is $\Omega(\log n)$
 - Why? - because it has been proven to be!
- Question:
 - Is searching a sorted list $\Theta(\log n)$?
 - Answer - Yes

Sorting

- Fact: The problem of sorting a list of n items into sorted order is $O(n^2)$
 - Why? - because we know about the algorithm *Insertion Sort*
- Fact: Comparison-based sorting is $\Omega(n \log n)$
 - Why? - because it has been proven to be!
- Question: Is comparison-based sorting $\Theta(n \log n)$?
- Answer - Yes (providing we can find an algorithm that sorts a list of n items with a worst-case time complexity of $O(n \log n)$)

Merge Sort

- Merge small, sorted portions of the list into larger, sorted portions
- Merge these larger sorted portions of the list into even larger, sorted portions
- Eventually end up with one, large, sorted portion (the entire list is sorted)
- A list that has less than 2 items is sorted
- A list that has more than 2 items can be sub-divided, sorted recursively and recombined

Merging two sorted lists

```
procedure Merge ( InListA, InListB, OutList )
(
  if ( InListA and InListB = empty ) then ( OutList = empty; Stop; )

  if ( InListA = empty )           then ( InListA = "exhausted"; )
                                  else ( CurrentA = first item in InListA; )

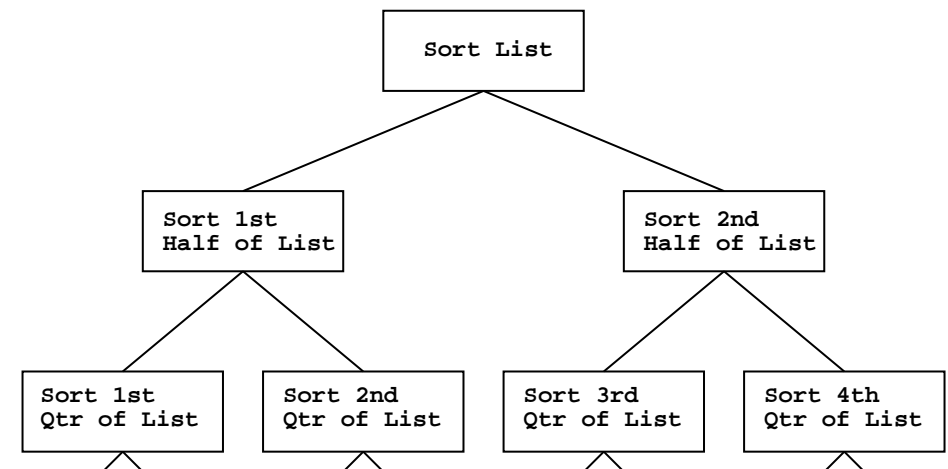
  if ( InListB = empty )           then ( InListB = "exhausted"; )
                                  else ( CurrentB = first item in InListB; )

  while ( InListA and InListB not "exhausted" )
  do (
    Compare CurrentA to CurrentB;
    Put "smaller" into OutList;
    if ( "smaller" is the last item in it's List )
    then ( That list = "exhausted"; )
    else ( Set next entry in that List to be Current; )
  )
  if ( There is a List that is not "exhausted" )
  then ( Copy remaining items into OutList; )
)
```

Merge sort

```
procedure MergeSort ( List )
[
  if ( List has more than one item )
  then ( half1 ← MergeSort ( first half of List; )
        half2 ← MergeSort ( second half of List; )
        Merge ( half1, half2 )
  )
]
```

Merge sort



Merge sort

- The number of comparisons made across each *level* of the *tree* is $O(n)$
- The number of levels in the tree is $O(\log n)$
- The worst-case time complexity of Merge sort is $O(n \log n)$
- Comparison-based sorting is $O(n \log n)$, $\Omega(n \log n)$ and therefore $\Theta(n \log n)$

Polynomial Problems

- Saying $g(n)$ is bounded by $f(n)$ means that as $n \rightarrow \infty$, $|f(n)| > |g(n)|$, e.g.
 - $\log n$ is bounded by n
 - $n \log n$ is bounded by n^2
- A problem is a *polynomial problem* if it is in $O(f(n))$ where $f(n)$ is a polynomial or is bounded by a polynomial
- The collection of all polynomial problems is denoted by \mathcal{P}

Non-Polynomial Problems

- Identifying whether a problem belongs to \mathcal{P} is important
- A problem that doesn't belong to \mathcal{P} does not have a *practical* solution
- A problem that doesn't belong to \mathcal{P} has an extremely long execution time even for moderate input sizes
- A problem that doesn't belong to \mathcal{P} has an exponential running time
- How many subsets are there of a set of n items ?

Intractability

- Problems exist whose complexities are large even if the “answer” is just YES/NO
- Is it true that there exists a number that when added to itself, produces the value 6 ?
- Is it true that there exists a non-zero real number that when added to itself produces zero ?
- Problems that are theoretically solveable but not in \mathcal{P} are said to be *intractable*

Non-determinism

- A deterministic algorithm does not rely on the creative capabilities of the mechanism executing the algorithm
- A nondeterministic algorithm may do so
- If a deterministic algorithm is executed repeatedly with the same input data, the same output will always be produced
- A non-deterministic algorithm might produce different output when executed under identical conditions

The Travelling Salesperson Problem

- A salesperson needs to visit a number of cities
- The distance between each pair of cities is known
- The salesperson has a travel budget that cannot be exceeded
- Why not consider each path in a sequential manner ?
- The number of paths grows exponentially as the number of cities increases !

The Travelling Salesperson Problem

- Pick a path and compute its distance
 - if this distance is within the budget, declare success
 - if not, declare failure
- This algorithm is non-deterministic
- Note that given an instance of this problem and a possible solution, we can check in polynomial time whether the solution is valid

Non-deterministic Solutions

- A non-deterministic solution is unsatisfactory
- A problem that has a non-deterministic solution may or may not have a deterministic solution
- In some cases we just don't know !
- There exist many problems in this category

The Class \mathcal{NP}

- $\mathcal{NP} = \text{non-deterministic polynomial}$
- i.e. a problem that can be solved in polynomial time by a non-deterministic algorithm
- All problems in \mathcal{P} are in \mathcal{NP}
- Any deterministic algorithm can have a non-deterministic instruction added to it without changing the performance
- **BIG QUESTION:** Are all problems in \mathcal{NP} in \mathcal{P} ?

\mathcal{NP} -Complete Problems

- An \mathcal{NP} -Complete problem is a problem that is in \mathcal{NP} that, if we could find a deterministic polynomial time solution, we could solve any other problem in \mathcal{NP} in polynomial time
- “The most difficult problems in \mathcal{NP} ”
- However, there are problems that are in \mathcal{NP} for which no polynomial time solution is known but they’re not known to be \mathcal{NP} -Complete either ! ... ?

Reading and Self Assessment

- Required reading:
 - Chapter 11
- Questions:
 - 11.1: 1, 2, 3
 - 11.2: 1, 2, 3
 - 11.4: 1, 2
 - 11.5: 1, 2, 3, 4