## Lec. 2 Introduction

- Presentation Choice 1
  - MapReduce paper
    - http://static.usenix.org/event/osdi04/tech/full_papers/dean/dean.pdf
    - http://code.google.com/edu/parallel/mapreduce-tutorial.html
    - You can suggest an interesting algorithm to present
- Reading Assignment and Homework
  - Week 1: Read Ch.1, Ch.2, Appendix A, B, C and D
  - Week 2: Read Ch3

1

## Course Focus

- The theoretical study of design and analysis of computer algorithms
  - Not about programming, not about math
  - Design: how to design correct algorithms which minimize cost
    - Efficiency is the design criterion
  - Analysis: predict the cost of an algorithm in terms of resource and performance

2

## What are the Basic Goals of Designing Algorithms?

- Basic goals for an algorithm
  - always correct
  - always terminates
- More, we also care about *performance*
  - Tradeoffs between what is possible and what is impossible
  - We usually have a deadline
    - E.g., Computing 24-hour weather forecast within 20 hours

3

## What is an Algorithm?

- An Algorithm → a well-defined computational procedure
  - take a set of values, as input
  - produce a set of values as output
- to solve a well-specified computational problem
- Here is how to formulate a sorting problem

  **Input:** A sequence of numbers $\{a_1, a_2, \ldots, a_n\}$

  **Output:** a permutation of input sequence such that $a_1 < a_2 < \ldots < a_n$

  Instance Input of a problem {2, 5, 9, 6, 4}

  Instance Output of a problem {2, 4, 5, 6, 9}

4

## Hard Problems

- We focus on efficient ALs in this class
- But some problems which we do NOT know any efficient solutions → NP-complete problems
  - **NP: non-deterministic polynomial**
- E.g., Traveling-salesman problem, **knapsack**, ...
  - **Input:** Distance-weighted graph *G*
  - **Problem:** Find the shortest route to visit all of the vertices exactly once

5

## NP-complete Problems

- Three interesting facts about NP-complete
  1. No existing efficient algorithms to solve them
  2. If we find one, then exist one for all of them
  3. Some of them similar to solved problems
- Under certain assumptions, efficient algorithms might give a near-optimum solution
  - these are called approximation algorithms

6

## Parallelism improve performance, not solving NP complete problems

- Computer performance increase: two key methods
  - Hierarchy: Cache, memory, I/O system, etc.
  - Parallel
    - Co-processors, Multicore
    - Data Center, Cloud computing
    - → hardware improvement does not help with NP-complete
- Map/Reduce: Simplified Data Processing on Large Clusters
  - http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/en/us/papers/mapreduce-osdi04.pdf
  - Large-Scale Data Processing
  - use 1000s of CPUs but don't want to ***manage*** things
  - 1st extra credit assignment → who like to make a 30-minute presentation on this?

7

## Basics of Algorithms

- An algorithm is said to be correct, if it halts with a correct output for every instance
  - Convergence → stop gradually
- An algorithm can be specified
  - In English → pseudo-code
  - as a computer program → word count program (wc)
  - a hardware design → TPM
  - The only requirement is that the specification must provide a precise description of the computational procedure to be followed

8

## Why do we study Algorithms

- Suppose computers were infinitely fast and computer memory was free
  - never true in reality
- Do we still have reasons to study algorithms?
- YES!!!
  - We still need to demonstrate our solution terminates with a correct answer
- Algorithms as technology
  - Resources are always limited → Efficiency is the center of algorithms

9

## Why study algorithms? Tech. Com.

- Cadence Design Systems, 1988
  - electronic design automation
  - Cadence claimed Avanti stole Cadence code !
  - Business Week: "*The Avanti case is probably the most dramatic tale of white-collar crime in the history of Silicon Valley*"
- Akamai, 1998 (old story: $300 per share in 1998)
  - Proxy and cache web contents, MIT Algorithm group
- Google, 1998
  - PageRank, Larry Page, Sergey Brin
  - MapReduce
- Baidu, 2000
  - RankDex site-scoring algorithm for search engines
  - NYU Buffalo, InfoSeek, Yanhong Li
- Match.com, 1993; eharmony.com, chemistry.com
  - 22 dimension matching! Positive and Negative

10

## The List goes on: Why study algorithms?

Their impact is broad and far-reaching

- Stocking trading: **May 6, 2010 Flash Crash**
- Internet. Web search, packet routing, distri. file sharing
- Biology. Human genome project, protein folding.
- Computers. Circuit layout, file system, compilers.
- Computer graphics. Hollywood movies, video games, 3-D
- Security. Cell phones, e-commerce, voting machines.
- Multimedia. CD player, DVD, MP3/4, JPG, DivX, HDTV.
- Transportation. Airline crew scheduling, map routing.
- Physics. N-body simulation, particle collision simulation.

11

## Course Goals

- **Learn critical thinking for problem solving**
  - **One of the most important computer science classes**
  - **How to think?**
    - **Learn to design, using well known methods**
  - **What can we try (e.g., to find a min in a set)?**
    - **Basic technique: brute-force, divide-and-conquer, dynamic programming, greedy**
- **Implementing algorithms correctly & efficiently**
  - **Correctness → Arguing correctness**
  - **Efficiency → Analyzing time complexity**

12

## Consider an algorithm, Main questions we have to answer

- **What is the Problem?**
  - **Find position of key x in a sorted array**
- **What is our Strategy?**
  - **Binary search**
- **Efficiency: how to achieve?**
  - **Log(n)**
- **Analysis of correctness**
  - **Prove it is correct**

13

## Importance of Algorithm Efficiency

- Time → CPU time (new: multicores)
- Storage → main memory (new: data overlay)
- I/O → new criterion in our current systems
- Examples
  - Sequential search vs. Binary search
    Basic operation: comparison
    Number of comparisons is grown in different rate
  - n-th Fibonacci sequence
    Recursive versus Iterative

14

## Example: search strategy

- Sequential search vs. Binary search

Problem: determine whether key x is in the sorted array S of n keys

- Inputs:
  - n: a positive integer
  - S: a sorted (*non-decreasing order*) array of n keys
    - **indexed from 1 to n**
  - x: a search key
- Output: the location of x in S (0 if x is not in S)

15

## Example: Sequential search

Basic operation: comparison

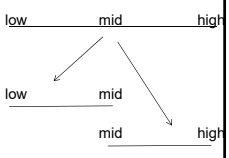
```
int Seq_search(int n, int S[], int x)
{
  int loc=0;
  while ( loc<=n && S[loc] != x)    // comparsion
          loc++;  // due to the non-decreasing order
  if (loc > n)
      return 0;
  else
      return loc;
}
```

Question: How to prove this AL is correct?

16

## Example: Binary search

```
int Binary_search(int n, const int S[], int x)
{          // Same inputs, different local variables
  int low, high, mid, location=0;
  low = 1; high =n;
  while ( low <= high && location ==0) {
    mid = floor( (low+high)/2 );
    if (x == S[mid])      // comparsion
        location = mid;
    else  if (x < S[mid] )
              high = mid -1;
          else   low = mid +1;
  }
  return location;
}
```




Question: How to prove this AL is correct?

17

## Example: number of comparisons

- Sequential search: worst case → search all n keys

  n = 32 128 1024 1,048,576

- Binary search: worst case → at most lg(n)+1

  lg(n) + 1 6 8 11 21

E.g., when n = 32 → lg(n) + 1 = 6

S[1],…, S[16],…, S[24], S[28], S[30], S[31], S[32]

(1st) (2nd) (3rd) (4th) (5th) (6th)

18

## Analysis of Algorithm Complexity

- Two main characters
  - Input size: n
  - Basic operation: e.g., comparison
- Time complexity notions
  - T(n) : Single-step time complexity of size n
  - W(n): Worst-case time complexity
  - A(n): Average-case time complexity
  - B(n): Best-case time complexity
- T(n) example
  - Add n array members together: T(n) = n-1
  - Matrix multiplication: (n by n) n*n*n
  - Exchange sort: n(n-1)/2

19

## Math preparation

- Please read Appendix A, B, C, and D
  - You are supposed to know them
- Induction
- Logarithm
- Sets
- Permutation and combination
- Limits
- Series
- Probability theory

20

## Programming preparation

- Data structure
- Install (1) Java 8 (2) Eclipse Neon
  - Eclipse Neon Help online
    - http://help.eclipse.org/neon/index.jsp
- Eclipse Develop Environment, http://www.eclipse.org/
  - For Java, C, and many other languages
  - **Install Eclipse on Linux**
  - **Install Eclipse on Windows**
    - **http://www.cs.dartmouth.edu/~cs5/install/eclipse-win/index.html**
  - Eclipse And Java: Free Video Tutorials (16 flash videos)
    - **http://eclipsetutorial.sourceforge.net/totalbeginner.html**
    - **Companion Tutorial Document http://eclipsetutorial.sourceforge.net/Total_Beginner_Companion_Document.pdf**
- Java Code of this book is in the "Resources" Section
  - You can create a Java project in Eclipse and import

21