Design and Analysis of Algorithms

(Week 1, Lecture 2)

Marks Distribution

	Frequency	Marks	Total
Quizzes	3	5	15
Assignments	3	5	15
Mid-Term	1	25	25
Mid-Term Viva	1	5	5
Final Exam	1	30	30
Final Viva	1	10	10

Algorithm Properties

- An algorithm *must* have five properties:
  1. Input specified
  2. Output specified
  3. Definiteness
  4. Effectiveness
  5. Finiteness

1. Input Specified

- The **input** is the data to be transformed during the computation to produce the output.
- What data do you need to begin to get the result you want?
- Input precision requires that you know what kind of data, how much and what form the data should be

2. Output Specified

- The output is the data resulting from the computation (your intended result)
- Frequently the name of the algorithm contains the output:
  - "Algorithm to compute batting average"
- Output precision also requires that you know what kind of data, how much and what form the output should be (or even if there will *be* any output at all!)

3. Definiteness

- Algorithms must specify every step and the order the steps must be taken in the process
- Definiteness means specifying the sequence of operations for turning input into output
- Details of each step must be also be spelled out (including how to handle errors)

4. Effectiveness

- For an algorithm to be effective, it means that all those steps that are required to get to output MUST BE DO ABLE!

5. Finiteness

- The algorithm must stop, eventually!
- Stopping may mean that you get the expected output OR you get a response that no solution is possible
- Finiteness is not usually an issue for noncomputer algorithms
- Computer algorithms often repeat instructions with different data and finiteness may be a problem

Other Algorithmic Characteristics

- “Algorithms Should Be General” and be applicable to several cases
- “Algorithms Should Use Resources Efficiently”:
  - Fast Speed
  - Minimal RAM or Disk Space
- Algorithms Should Be Understandable” so that the operations are clear to readers

Other Algorithmic Characteristics

- “Algorithms Should Be Clear and Precise” despite the language used
  - Natural languages are *ambiguous* in that words or phrases may have multiple meanings
  - Programming languages are formal languages with clear, unambiguous rules

The Stair Steps

- **S** TATE the problem in ENGLISH
- **T** OOLS for the job
- **A** LGORITHM development
- **I** MPLEMENTATION of the algorithm
- **R** EFINEMENT

S – State the problem in English

- This step seems obvious enough, but it is often the one people skip.
- We frequently start solving problems before we really think carefully about what problem we are trying to solve.
- The result is confusion and wasted time. Take the time to describe carefully to yourself what you are trying to accomplish.

S – State the problem in English

- At this stage of the process, all your thoughts should be in ENGLISH (or, of course, whatever other human language you think most comfortably in).
- Use whatever you need to make sure you have a firm understanding of what you want to accomplish.
- It's best to write this down on paper so the problem statement can be checked later if you get confused.
- If you don't know where you are trying to go, how will you know when you get there?

T OOLS for the job

- Once we know what kind of a job we want to accomplish, we can begin to explore which tools we might use to solve the job.
- A tool might be a command, a button on a toolbar, a selection on a drop-down menu, a strategy, a program, or something else, depending on the kind of job we are trying to do and the context in which we are working.

T OOLS for the job

- Knowing the capabilities of a certain program, and the kinds of things different types of computer applications can do, are the main ways you add new tools to your toolbox of computing skills.
- As you gain experience, you will constantly be adding new tools of all kinds to your range.
- Even the STAIR strategy itself can be considered a tool.

T OOLS for the job

- Most of the time, there is more than one tool available to do a job.
- At this point, just list the available tools
- Later on, you will decide which tool or tools to use.
- You might need to come back to this list if you chose an inappropriate tool the first time through the process.

A – Algorithm development

- An algorithm is a computer science term for a strategy or plan of action.
- Part of developing an algorithm is choosing an appropriate tool or set of tools from the previous step.
- The other component is determining how those tools will be used to solve the problem.
- The algorithm can vary widely in the type and complexity of the strategy you will use.

A – Algorithm development

- In simple problems, your algorithm may be as simple as saying "I'm going to try pressing this particular button on the menu bar."
- A more complex problem will probably require a more complex algorithm.
- It is a good practice to write down your algorithm on paper.

I – Implementation of the algorithm

- It is interesting to note that none of the above steps require the use of a computer.
- The implementation step is the actual process of translating our human thought into something the computer can understand.
- In simple problems, our algorithm may be implementing a command or two.
- Implementation can mean "just do it" when we are dealing with these simple problems.

I – Implementation of the algorithm

- Programmers think of the implementation step as translating the algorithm into some type of computer programming language.
- The process is the same regardless of the complexity of the problem

R – Refinement

- We like to think if we learn a skill and prepare ourselves properly, we can solve a problem on the first attempt.
- Experience shows us this is not usually the case.
- It is normal for a computer user to attempt a solution several times before the problem is solved.
- A skilled problem solver will analyze what happened, review the other steps, and try again.

R – Refinement

- Each unsuccessful attempt should bring you closer to an understanding of the problem and its solution.
- Refinement usually means going back and looking at the previous steps critically.
- Ask yourself if you really defined the problem properly.
- If so, have you used all the possible tools at your disposal?

R – Refinement

- Are you sure there is not a tool available that you have overlooked?
- Did you choose the best algorithm for the job?
- Did you implement the solution properly? (You would be amazed at the number of computer errors that are the result of simple typing or spelling errors!)
- Again, you will find that practice will make you much more confident at this critical stage of the process.

Algorithm Facts

1. Algorithms can be specified at different levels of detail
   - Algorithms use functions to simplify the algorithmic description
   - These functions (such as *scan*) may have their own algorithms associated with them

Algorithm Facts

2. Algorithms always build on functionality previously defined and known to the user
   - Assume the use familiar functions and algorithms
   - For example, how might *scan* be defined? Would it be consistent for everyone? Could it mean alphabetize? Look for similar formats? Are these the same?

Algorithm Facts

3. Different algorithms can solve the same problem differently, and the different solutions can take different amounts of time (or space)

How Do We Know it Works?

- Algorithm solution is clear and simple and efficient
- Then, how do we know it works?
- If there is no loop, the program runs, gets to an end, and we can check the result
- What if there is a loop?
  - Programs with loops cannot be absolutely verified that it works…there are too many possible cases

Then, what?

- *The way to know that an algorithm works is to know why it works…*

- Strategy for knowing why it works:
  - Find one or more properties that ensure the algorithm works
  - Explain, using the program, why they make it work.