Chapter 1: Background and Review

Devesh C Jinwala , IIT Jammu, India

August 5, 2019

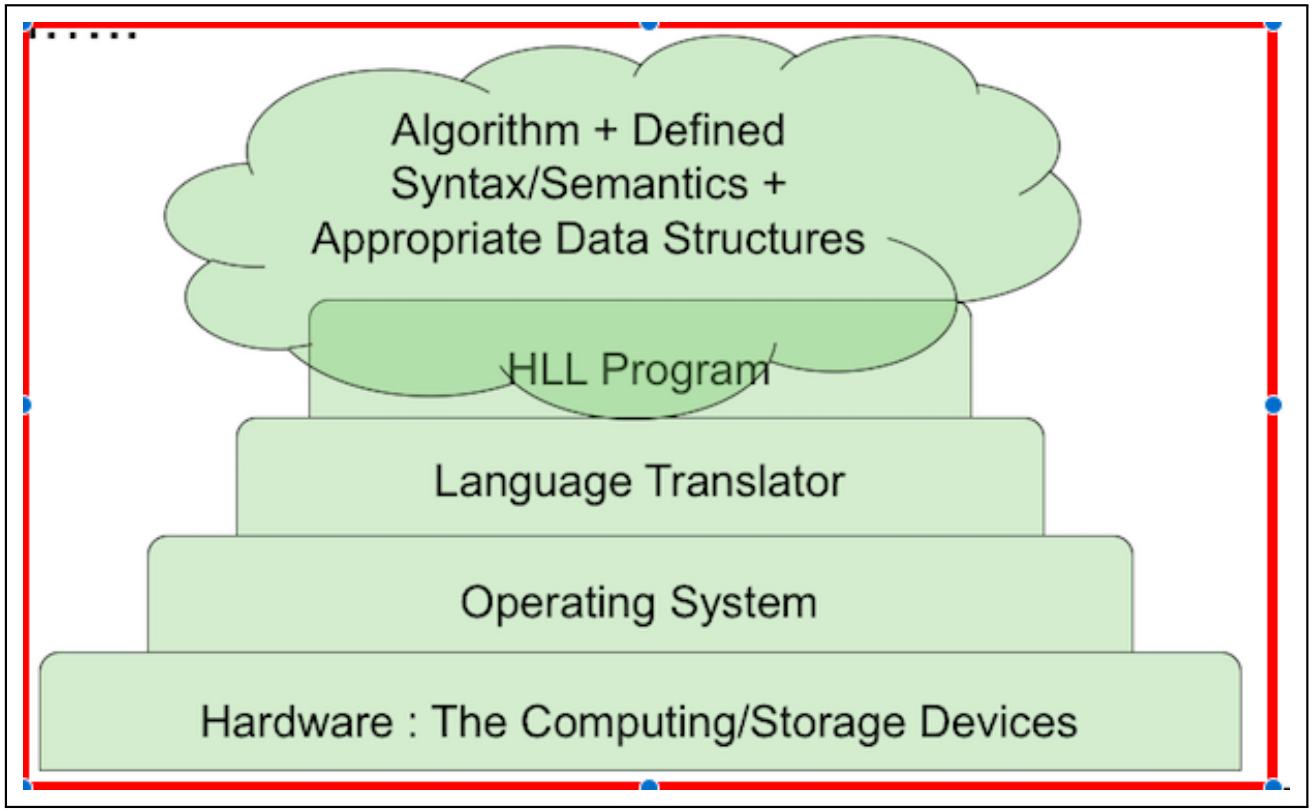
Design and Analysis of Algorithms
IIT Jammu, Jammu

1 Background

2 Methods of Analysis

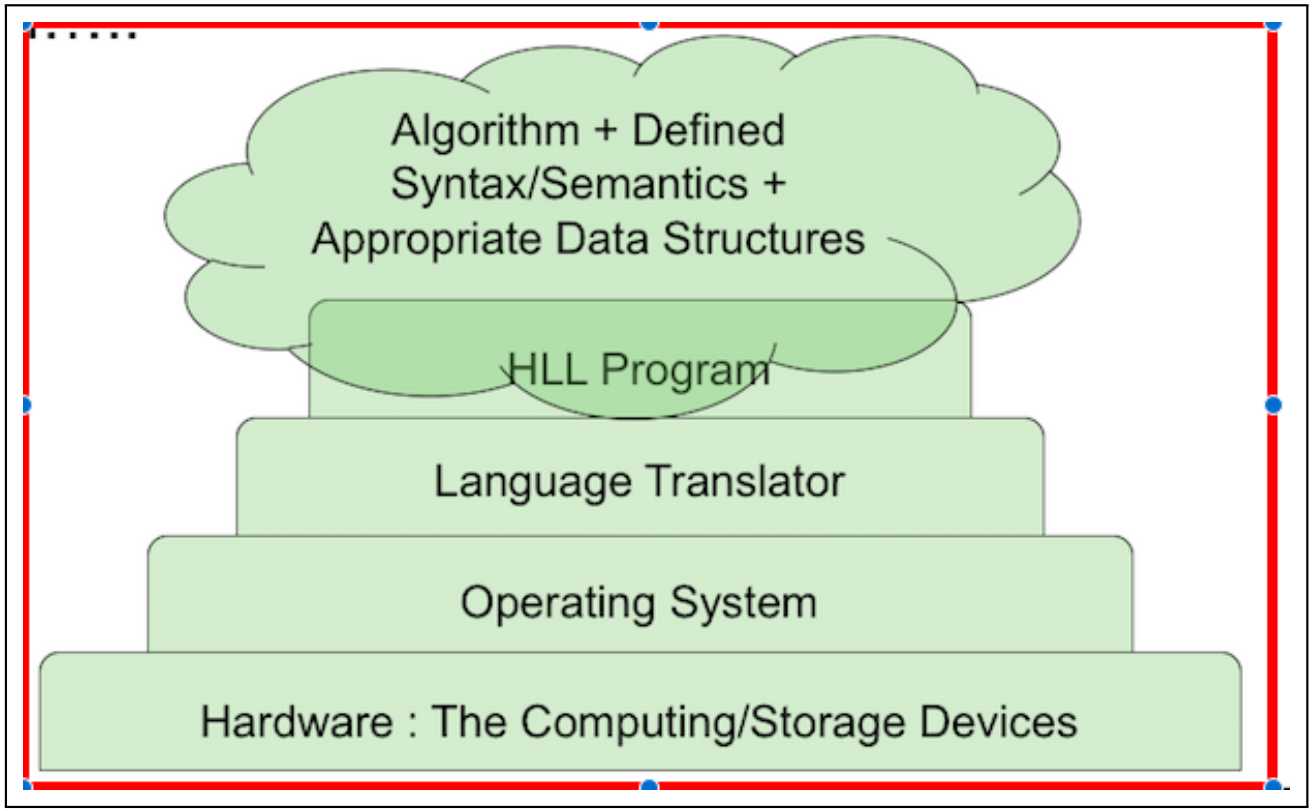
Review and Background

- Let us look a typical layered view of a Computer System



Review and Background

- Let us look a typical layered view of a Computer System



- Where do we see *algorithm* in this schematic ?

A Program and an Algorithm

- Defining an algorithm

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm
- How do we relate a program and an algorithm ?

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm
- How do we relate a program and an algorithm ?
- A Program = a set of conventions for communicating an algorithm to two different entities. Which ones ?

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm
- How do we relate a program and an algorithm ?
- A Program = a set of conventions for communicating an algorithm to two different entities. Which ones ?
- Therefore, an algorithm should not involve any subjective decisions.

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm
- How do we relate a program and an algorithm ?
- A Program = a set of conventions for communicating an algorithm to two different entities. Which ones ?
- Therefore, an algorithm should not involve any subjective decisions.
- What is a subjective decision?

A Program and an Algorithm

- Defining an algorithm
- named after Persian mathematician al-Khowrizm
- How do we relate a program and an algorithm ?
- A Program = a set of conventions for communicating an algorithm to two different entities. Which ones ?
- Therefore, an algorithm should not involve any subjective decisions.
- What is a subjective decision?
- Tutorial-1: Give at least two different real-life examples of a subjective decision and a non-subjective decision.

Exceptions to Non-Subjective Decisions in Algorithms

- There are three exceptions that allow non-subjectivity in algorithms.

Exceptions to Non-Subjective Decisions in Algorithms

- There are three exceptions that allow non-subjectivity in algorithms.
- What are those algorithm categories ?

Exceptions to Non-Subjective Decisions in Algorithms

- There are three exceptions that allow non-subjectivity in algorithms.
- What are those algorithm categories ?
- Why do we tolerate and permit such exceptions ?

Exceptions to Non-Subjective Decisions in Algorithms

- There are three exceptions that allow non-subjectivity in algorithms.
- What are those algorithm categories ?
- Why do we tolerate and permit such exceptions ?
- What is the difference between a heuristic and an approximation algorithm?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?
  - What does solving a problem algorithmically mean?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?
  - What does solving a problem algorithmically mean?
  - What can be and what cant be computed?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?
  - What does solving a problem algorithmically mean?
  - What can be and what cant be computed?
  - Is every algorithm conceived a feasible algorithm ?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?
  - What does solving a problem algorithmically mean?
  - What can be and what cant be computed?
  - Is every algorithm conceived a feasible algorithm ?
  - When can an algorithm be considered feasible ?

Why are we reviewing these basics again ? Takeways

- Our goal is to reemphasize that
  - To understand that the practical success of computer science is also built on elegant & solid foundations.
- We study algorithms to answer the questions like
  - What is an algorithm ?
  - What does solving a problem algorithmically mean?
  - What can be and what cant be computed?
  - Is every algorithm conceived a feasible algorithm ?
  - When can an algorithm be considered feasible ?
  - How can we compare two algorithms ?

Related areas of study

- Computability theory

Related areas of study

- Computability theory
- Computational Complexity study

Related areas of study

- Computability theory
- Computational Complexity study
- Undecidability

Related areas of study

- Computability theory
- Computational Complexity study
- Undecidability
- An example of an undecidable problem.

Comparing two algorithms

- How can we compare two algorithms ?

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.
- Why do we need to analyze the performance of algorithms.

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.
- Why do we need to analyze the performance of algorithms.
- Performance in terms of what ?

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.
- Why do we need to analyze the performance of algorithms.
- Performance in terms of what ?
- Algorithm analysis - concerned with demand for resources and performance

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.
- Why do we need to analyze the performance of algorithms.
- Performance in terms of what ?
- Algorithm analysis - concerned with demand for resources and performance
  - A-priori estimates - Performance analysis

Comparing two algorithms

- How can we compare two algorithms ?
- We will need to analyze the performance of two algorithms.
- Why do we need to analyze the performance of algorithms.
- Performance in terms of what ?
- Algorithm analysis - concerned with demand for resources and performance
  - A-priori estimates - Performance analysis
  - A-posteriori analysis - Measurement & testing

Methods of Analysis

- Mathematical Analysis .... are the efforts put-in worth it ?

Methods of Analysis

- Mathematical Analysis .... are the efforts put-in worth it ?
- Empirical Analysis .....is it feasible ?

Methods of Analysis

- Mathematical Analysis .... are the efforts put-in worth it ?
- Empirical Analysis .....is it feasible ?
- Asymptotic Analysis ...is the way to go.

Methods of Analysis

- Mathematical Analysis .... are the efforts put-in worth it ?
- Empirical Analysis .....is it feasible ?
- Asymptotic Analysis ...is the way to go.
- How to convince ourselves?

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.
  - Time ???? !!!!

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.
  - Time ???? !!!!
- For the time being, let us not bother about the exact time taken on a machine.. the computational/storage power on different machines vary

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.
  - Time ???? !!!!
- For the time being, let us not bother about the exact time taken on a machine.. the computational/storage power on different machines vary
- But, let us assume that

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.
  - Time ???? !!!!
- For the time being, let us not bother about the exact time taken on a machine.. the computational/storage power on different machines vary
- But, let us assume that
  - the cost of execution of a statement - in terms of time units - is some absract cost c_i.

Mathematical Analysis

- Let us first focus on a simple example to understand the notations and conventions
- Assumptions
  - The algorithms we wish to compare are correct
  - The algorithm that provably takes lesser time is a better one.
  - Time ???? !!!!
- For the time being, let us not bother about the exact time taken on a machine.. the computational/storage power on different machines vary
- But, let us assume that
  - the cost of execution of a statement - in terms of time units - is some absract cost c_i.
  - An algorithm to find the maximum element of an n-element array.

Mathematical Analysis

- An algorithm to find the maximum element of an n-element array.

Mathematical Analysis

- An algorithm to find the maximum element of an n-element array.
  - Method1: Algorithm $Largest1(x_i, n)$ takes total steps = total time = $C_1 + nC_2 + (n-1)C_3 + (n-1)C_4 + C_5$

Mathematical Analysis

- An algorithm to find the maximum element of an n-element array.
  - Method1: Algorithm $Largest1(x_i, n)$ takes total steps = total time = $C_1 + nC_2 + (n-1)C_3 + (n-1)C_4 + C_5$
  - Method2: Algorithm $Largest2(x_i, n)$ takes total steps = total time = $nC_1 + (n-1)C_2 + (n-1)C_3 + C_4$

Mathematical Analysis

- An algorithm to find the maximum element of an n-element array.
  - Method1: Algorithm $Largest1(x_i, n)$ takes total steps = total time = $C_1 + nC_2 + (n-1)C_3 + (n-1)C_4 + C_5$
  - Method2: Algorithm $Largest2(x_i, n)$ takes total steps = total time = $nC_1 + (n-1)C_2 + (n-1)C_3 + C_4$
- Which algorithm is better for the same size of input n ?

Mathematical Analysis...

- Major Assumptions
  - the abstract operations are machine independent.
  - a constant amount of time is required to execute each line of pseudocode
- How to count the program steps?
  - comments, declarations
  - assignment statement
  - iterative statement
- How to instantiate the values of $c_i's$?

Tutorial Problems 2 to 6

- Devise the algorithm and perform the analysis as illustrated in the previous example
  - To find the sum of n elements in an integer array without using recursion.
  - To perform the bubble sort.
  - To find the smallest element from an integer array.
  - To find the factorial of a given number without using recursion.
  - To find the n_{th} Fibonacci number without using recursion.

Tutorial Problem No 7: Analyzing Insertion Sort

- The Insertion sort code as written here on board.
- Do the dry run on the input array 10,11,12 and prepare a table as shown below for all iterations ?

j = 2	i = 1	key=12	13	12	10
j = 2	i = 1	key=12	13	13	10
j = 2	i = 0	key=12	12	13	10

- What is the rough estimate of the complexity of the insertion sort ?
- But, how to carry out its mathematical analysis ?

Tutorial Problem No 7: Analyzing Insertion Sort...

- We need to analyze how many times the while loop is executed ?

Tutorial Problem No 7: Analyzing Insertion Sort...

- We need to analyze how many times the while loop is executed ?
- Assume while loop test is executed t_j times for every value of j.I

Tutorial Problem No 7: Analyzing Insertion Sort...

- We need to analyze how many times the while loop is executed ?
- Assume while loop test is executed t_j times for every value of j.I
- Time=

$$c_1 n + c_2(n-1) + c_3(n-1) + c4 \sum_{j=2}^{n} t_j + c5 \sum_{j=2}^{n} (t_j - 1) + c6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1) \quad (1)$$

Tutorial Problem No 7: Analyzing Insertion Sort...

- We need to analyze how many times the while loop is executed ?
- Assume while loop test is executed t_j times for every value of j.I
- Time=

$$c_1 n + c_2(n-1) + c_3(n-1) + c4 \sum_{j=2}^{n} t_j + c5 \sum_{j=2}^{n} (t_j - 1) + c6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1) \quad (1)$$

- Worst case Time = ?

Tutorial Problem No 7: Analyzing Insertion Sort...

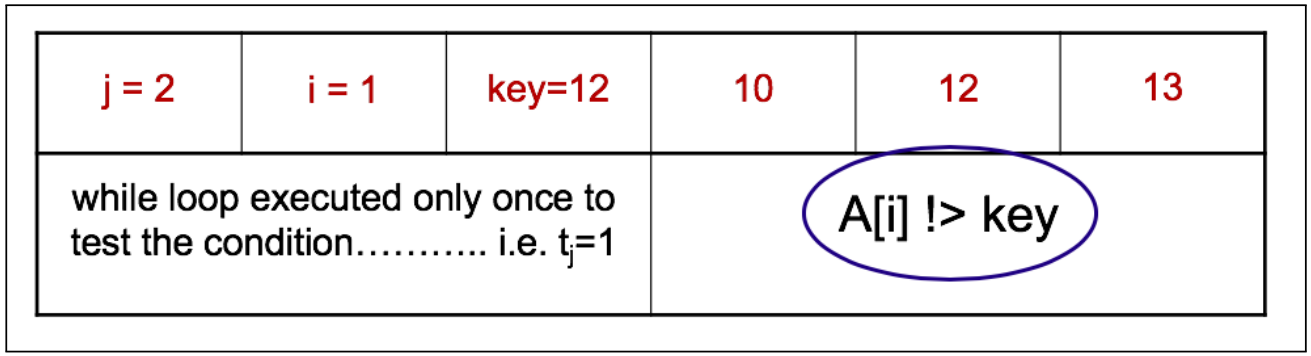
- We need to analyze how many times the while loop is executed ?
- Assume while loop test is executed t_j times for every value of j.I
- Time=

$$c_1 n + c_2(n-1) + c_3(n-1) + c4 \sum_{j=2}^{n} t_j + c5 \sum_{j=2}^{n} (t_j - 1) + c6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1) \quad (1)$$

- Worst case Time = ?
- Best case Time = ?

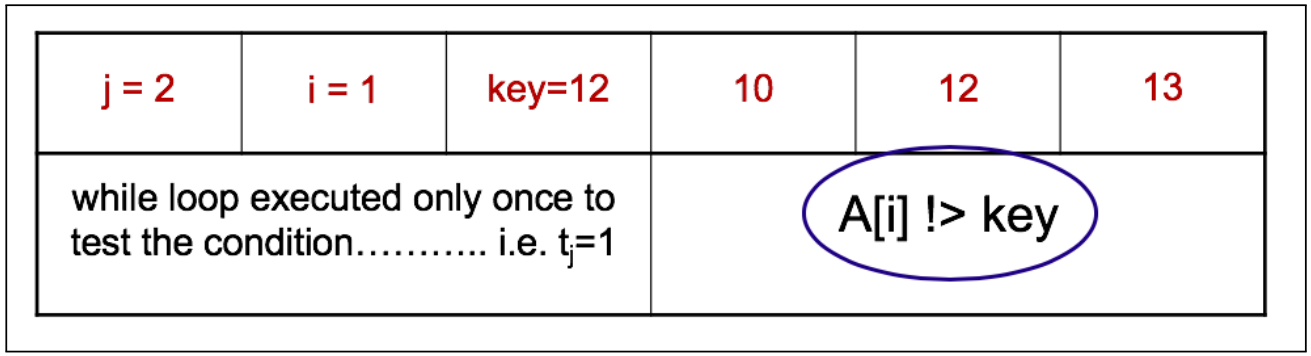
Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?



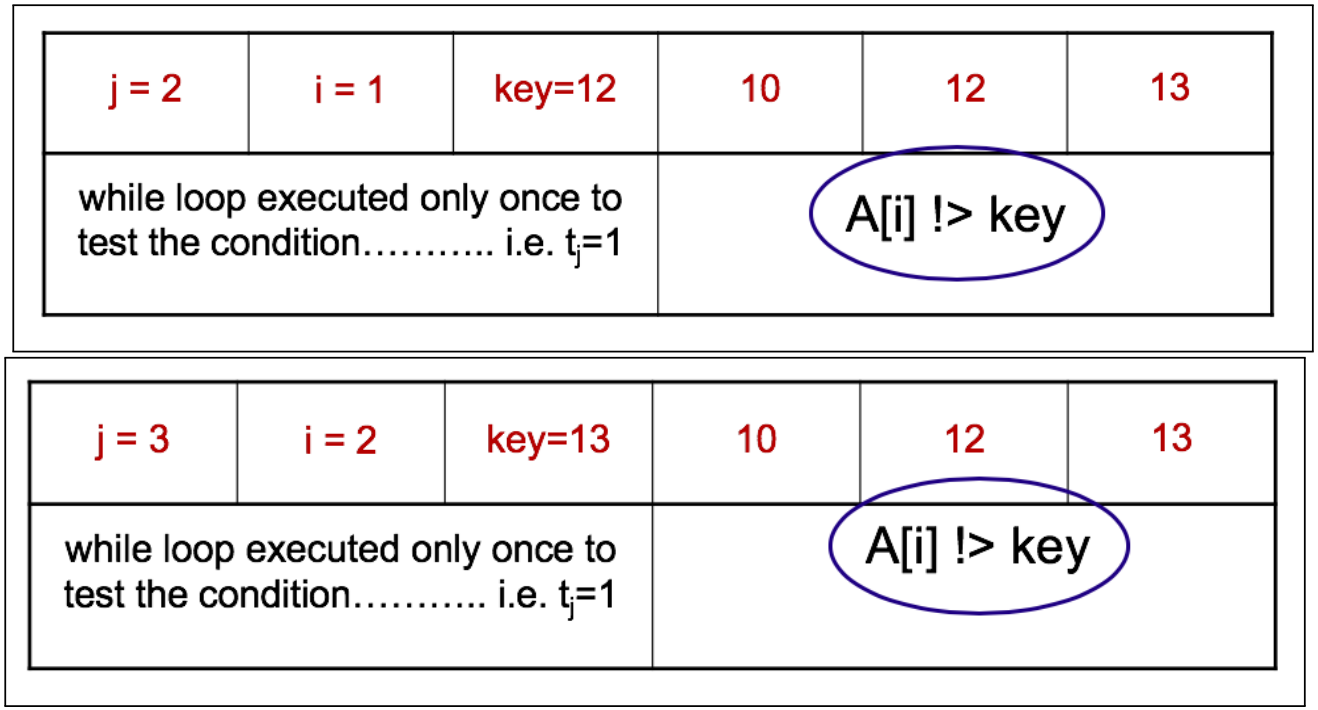
Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?



Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?



$$Time = c_1 n + c_2(n-1) + c_3(n-1) + c4 \sum_{j=2}^{n} t_j + c5 \sum_{j=2}^{n} (t_j - 1) + c6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1) \quad (2)$$

- Best case Time = ?

Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?

j = 2	i = 1	key=12	13	12	10
j = 2	i = 1	key=12	13	13	10
j = 2	i = 0	key=12	12	13	10

- j=2 and so two iterations.....

Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?

j = 2	i = 1	key=12	13	12	10
j = 2	i = 1	key=12	13	13	10
j = 2	i = 0	key=12	12	13	10

- j=2 and so two iterations.....

Tutorial Problem No 7: Analyzing Insertion Sort...

- Best case - Apply the fact that the while loop test is executed t_j times for every value of j. What is t_j in this case ?

j = 2	i = 1	key=12	13	12	10
j = 2	i = 1	key=12	13	13	10
j = 2	i = 0	key=12	12	13	10

- j=2 and so two iterations.....

j = 3	i = 2	key=10	12	13	10
j = 3	i = 2	key=10	12	13	13
j = 3	i = 1	key=10	12	12	13
j = 3	i = 0	key=10	10	12	13

$$Time = c_1 n + c_2(n-1) + c_3(n-1) + c4 \sum_{j=2}^{n} t_j + c5 \sum_{j=2}^{n} t_j - 1 + c6 \sum_{j=2}^{n} t_j - 1 + c_7(n-1)j \quad (3)$$

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?
- Our assumptions ?

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?
- Our assumptions ?
  - We assumed that the time taken by a statement to execute is some abstract units... c_i's

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?
- Our assumptions ?
  - We assumed that the time taken by a statement to execute is some abstract units... c_i's
  - We analyzed only the best-case or the worst-case performance.

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?
- Our assumptions ?
  - We assumed that the time taken by a statement to execute is some abstract units... c_i's
  - We analyzed only the best-case or the worst-case performance.
  - Should we compute the exact time taken by a statement to execute ? Would it make sense ?

Reviewing our analysis approach

- Which analysis technique we have been studying so far ?
- What do the c_i's represent ?
- Our assumptions ?
  - We assumed that the time taken by a statement to execute is some abstract units... c_i's
  - We analyzed only the best-case or the worst-case performance.
  - Should we compute the exact time taken by a statement to execute ? Would it make sense ?
- How to find the exact values of c_i's?

Empirical Analysis

- Do not make any assumptions.

```
typedef struct timeval {long tv_sec;
                        long tv_usec;
} timeval;
```

Empirical Analysis

- Do not make any assumptions.
- How to time a program ? Any Linux commands ?

```
typedef struct timeval {long tv_sec;
                        long tv_usec;
} timeval;
```

Empirical Analysis

- Do not make any assumptions.
- How to time a program ? Any Linux commands ?
- What we need is a multi-resolution timer... How to design one ?

```
typedef struct timeval {long tv_sec;
                        long tv_usec;
} timeval;
```

Empirical Analysis

- Do not make any assumptions.
- How to time a program ? Any Linux commands ?
- What we need is a multi-resolution timer... How to design one ?
- Time the program using *timeval()* structure and *gettimeofday()*, *settimeofday()* family of calls in linux multiresolution timer

```
typedef struct timeval {long tv_sec;
                        long tv_usec;
} timeval;
```

Empirical Analysis

- Do not make any assumptions.
- How to time a program ? Any Linux commands ?
- What we need is a multi-resolution timer... How to design one ?
- Time the program using *timeval()* structure and *gettimeofday()*, *settimeofday()* family of calls in linux multiresolution timer
- Logic

```
typedef struct timeval {long tv_sec;
                        long tv_usec;
} timeval;
```

Tutorial Problem No 8, 9

1. Create a data set or find a dataset from the internet - consisting of at least a million interger values in a vector. Write the Insertion sort routine in C and time the function to sort using the approach just discussed. Now repeat the same on the sorted output. Note the difference in time - in sorting an unsorted interger vector and a sorted one.
2. Repeat the above exercise for the Bubble sort, the Merge sort and the Quick sort covered. Note the time diffences.

Empirical Analysis...: Performance is relative

The outputs of such timing program obviously depend

- on many local factors
  - Machine
  - Compiler, Operating System
  - Algorithm
  - Input Data
  - ... ...
- and on many NOT so obvious factors
  - Caching
  - Garbage collection routines
  - Just-in-time compilation
  - CPU sharing/not.

Our observations and inferences

- It is often difficult to get precise measurements of c_is

Our observations and inferences

- It is often difficult to get precise measurements of c_is
- Therefore, our assumptions are justified.. let the abstract operations remain machine independent.

Our observations and inferences

- It is often difficult to get precise measurements of c_is
- Therefore, our assumptions are justified.. let the abstract operations remain machine independent.
- let *a constant amount of time* be required to execute each line of pseudocode

Our observations and inferences

- It is often difficult to get precise measurements of c_is
- Therefore, our assumptions are justified.. let the abstract operations remain machine independent.
- let *a constant amount of time* be required to execute each line of pseudocode
- Hence, Empirical analysis may not be useful
  at least in comparing the algorithms..

Our observations and inferences

- It is often difficult to get precise measurements of c_is
- Therefore, our assumptions are justified.. let the abstract operations remain machine independent.
- let *a constant amount of time* be required to execute each line of pseudocode
- Hence, Empirical analysis may not be useful
  at least in comparing the algorithms..
- The last question that still remains is "How to estimate the values of c_i's?

Estimating the value of c_i's

Integer add	a + b	2.1 ns
Integer multiply	a * b	2.4 ns
Integer divide	a / b	5.4 ns
Floating point add	a + b	4.6 ns
Floating point multiply	a * b	4.2 ns
Floating point divide	a / b	13.5 ns
sine	Math.sin(theta)	91.3 ns
arctangent	Math.atan(theta)	129. Ns

*Source : Sedgewick *Running OS X on Macbook Pro.....

Estimating the value of c_i's ...

- Therefore, now what could be our estimation of a typical c_i value ?

Cost of c_i's

Therefore, now, we shall assume that the abstract costs c_1, c_2, c_3, are all equal and unity

Estimating the value of c_i's ...

- Therefore, now what could be our estimation of a typical c_i value ?
- Say when the input size is very large - typically mlllion or ten million or so, does this value of c_i have any impact on the time taken ?

Cost of c_i's

Therefore, now, we shall assume that the abstract costs c_1, c_2, c_3, are all equal and unity

How to estimate the time of recursive routines ?

- The same method i.e. assume that a statement whose basic cost is c_i, if executed n times, costs nc_i - does not work.

How to estimate the time of recursive routines ?

- The same method i.e. assume that a statement whose basic cost is c_i, if executed n times, costs nc_i - does not work.
- How to solve then...

How to estimate the time of recursive routines ?

- The same method i.e. assume that a statement whose basic cost is c_i, if executed n times, costs nc_i - does not work.
- How to solve then...
- Tutorial Problem No 8.....

Tutorial Problem No 10: Computing sum of vector recursively

- The recursive algorithm to sum

Tutorial Problem No 10: Computing sum of vector recursively

- The recursive algorithm to sum
- Writing a recursion relation

Tutorial Problem No 10: Computing sum of vector recursively

- The recursive algorithm to sum
- Writing a recursion relation
- Solving the recursion relation.

Tutorial Problem No 10: Computing sum of vector recursively

- The recursive algorithm to sum
- Writing a recursion relation
- Solving the recursion relation.
- Comparing the complexity with the iterative version.

Revisiting our results

A relook at the time complexity expressions we have obtained so far

- Finding the maximum of n elements. Time complexity $= c_1 + c_2$ n $+ c_3$ (n-1) $+ c_4$ (n-1) $+ c_5 = 3n$

Revisiting our results

A relook at the time complexity expressions we have obtained so far

- Finding the maximum of n elements. Time complexity $= c_1 + c_2$ n $+ c_3$ (n-1) $+ c_4$ (n-1) $+ c_5 = 3n$
- Finding the sum of n elements of an input vector. Time complexity $= c_1 + c_2(n+1) + c_3 n + c_4 = 2n + 3$

Revisiting our results

A relook at the time complexity expressions we have obtained so far

- Finding the maximum of n elements. Time complexity $= c_1 + c_2$ n $+ c_3$ (n-1) $+ c_4$ (n-1) $+ c_5 = 3n$
- Finding the sum of n elements of an input vector. Time complexity $= c_1 + c_2(n+1) + c_3 n + c_4 = 2n + 3$
- Insertion Sort - Best Case: $5n - 4$. Worst Case:$3n^2 + 7n - 8$

Revisiting our results

A relook at the time complexity expressions we have obtained so far

- Finding the maximum of n elements. Time complexity $= c_1 + c_2$ n $+ c_3$ (n-1) $+ c_4$ (n-1) $+ c_5 = 3n$
- Finding the sum of n elements of an input vector. Time complexity $= c_1 + c_2(n+1) + c_3 n + c_4 = 2n + 3$
- Insertion Sort - Best Case: $5n - 4$. Worst Case:$3n^2 + 7n - 8$
- Bubble Sort - Best Case: $2n^2 - 2n + 1$. Worst case: $3n^2 - 4n + 2$
  Which term dominates the overall result in the above expression, especially at large values of n ?

The Growth of Functions

n	2n	4.5n	$n^3/2$	$5n^2$
5	10	22		
10	20	45		
100	200	450		
1000	2000	4500		
10000	20000	45000		
100000	$2.0 * 10^5$	$4.5 * 10^5$		
1000000	$2.0 * 10^6$	$4.5 * 10^6$		

The Growth of Functions ...

n	2n	4.5n	$n^3/2$	$5n^2$
5	10	22	45	125
10	20	45	500	500
100	200	450	$5*10^5$	$5*10^4$
1000	2000	4500	$5*10^8$	$5*10^6$
10000	20000	45000	$5*10^{11}$	$5*10^8$
100000	200000	450000	$5*10^{14}$	$5*10^{10}$
1000000	2000000	4500000	$5*10^{17}$	$5*10^{12}$

The Growth of Functions...

n	$T(n) = 3n^2 + 7n - 8$	$T(n) = 3n^2$
10	362	300
100	30692	30000
1000	$3.006992 * 10^6$	$3.00 * 10^6$
100000	$3.0000699992 * 10^{10}$	$3.00 * 10^{10}$

The Growth of Functions...

The Time Complexity

Hence, we shall now also drop the all the terms except the highest degree of the polynomial, when analyzing the running time of the algorithm.

Reviewing our Assumptions

So, now we have assumed/abstracted at three different levels viz.

- level 1 ignored the actual cost of execution of each statement.

Reviewing our Assumptions

So, now we have assumed/abstracted at three different levels viz.

- level 1 ignored the actual cost of execution of each statement.
- level 2 ignored even the abstract cost (Ci) of each statement

Reviewing our Assumptions

So, now we have assumed/abstracted at three different levels viz.

- level 1 ignored the actual cost of execution of each statement.
- level 2 ignored even the abstract cost (Ci) of each statement
- level 3 ignore all the terms except for the one with the highest degree in the expression of time complexity

Reviewing our Assumptions

So, now we have assumed/abstracted at three different levels viz.

- level 1 ignored the actual cost of execution of each statement.
- level 2 ignored even the abstract cost (Ci) of each statement
- level 3 ignore all the terms except for the one with the highest degree in the expression of time complexity

Reviewing our Assumptions

So, now we have assumed/abstracted at three different levels viz.

- level 1 ignored the actual cost of execution of each statement.
- level 2 ignored even the abstract cost (Ci) of each statement
- level 3 ignore all the terms except for the one with the highest degree in the expression of time complexity

Asymptotic Analysis

Such analysis is based on the asymptotic growth rate, asymptotic order or order of functions and called asymptotic analysis

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?
  - Abstract costs are insignificant, anyway

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?
  - Abstract costs are insignificant, anyway
  - At higher magnitude, only the first term in the polynomial matters.

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?
  - Abstract costs are insignificant, anyway
  - At higher magnitude, only the first term in the polynomial matters.
  - At lower magnitude, do we need to care about even comparing the algorithms ?!!!!

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?
  - Abstract costs are insignificant, anyway
  - At higher magnitude, only the first term in the polynomial matters.
  - At lower magnitude, do we need to care about even comparing the algorithms ?!!!!

Reviewing : Methods of Analysis

There can be at least three different ways of analysing the algorithms

- Empirical analysis. is it feasible ?
  - The resources on different machines globally vary..
  - Not possible to make a universally true judgment about the algorithm
- Mathematical analysis..is it worth ?
  - Abstract costs are insignificant, anyway
  - At higher magnitude, only the first term in the polynomial matters.
  - At lower magnitude, do we need to care about even comparing the algorithms ?!!!!

Way to go

Therefore, the asymptotic analysis is the best way to go.

Various Asymptotic Orders

$\lg n$	$n^{1/2}$	n	$n \lg n$	$n (\lg n)^2$	n^2
3	3	10	33	110	100
7	10	100	664	4414	10000
10	32	1000	9966	99317	10^6
13	100	10000	132877	1765633	10^8
17	316	100000	16660964	27588016	10^{10}
20	1000	1000000	19931569	397267426	10^{12}

An interesting seconds conversion

10^2	1.7 min
10^4	2.8 hours
10^5	1.1 days
10^6	1.6 weeks
10^7	3.8 months
10^8	3.1 years
10^9	3.1 decades
10^{10}	3.1 centuries

An interesting observation

	n	$n \lg n$	N^2	N^3	1.5^n	2^n	$n!$
n=10	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 4 sec
n=30	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} yrs
n=50	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 yrs	very long
n=100	< 1 sec	< 1 sec	< 1 sec	1 sec	12.89 yrs	10^{17} yrs	< 4 sec
n=1000	< 1 sec	< 1 sec	1 sec	18 min sec	very long	very long	very long
n=10K	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
n=100K	< 1 sec	2 sec	3 hrs	32 yrs	very long	very long	very long
n=1M	1 sec	20 sec	12 days	31.71 yrs	very long	very long	very long

Basic Asymptotic Efficiency classes

1	constant
log n	logarithmic
n	linear
$n \log n$	$n \log n$
n^2	quadratic
n^3	cubic
2^n	exponential
$n!$	factorial

Common Expressions & Complexity

Growth Rate	Typical Code Framework	Description	Example
1	a = b + c;	statement	add two statements
log n	while (n>1) { n = n/2; ……}	divide in half	binary search
n	for i= 1 to n { …….}	loop	find the max
n log n	……	divide & conquer	mergesort
n^2	for i=1 to n { for j = 1 to n { … … …}}	double loop	check all pairs
n^3	for i=1 to n { for j = 1 to n { for k = 1 to n } … … …}}}	triple loop	check all triples
2^n	… … … … …	exhaustive search	check all possibilities

Tutorial Problem No 11

What is/could be the input size, in the following ?

- Find x in an array of names

Tutorial Problem No 11

What is/could be the input size, in the following ?

- Find x in an array of names
- Multiply two matrices with real entities

Tutorial Problem No 11

What is/could be the input size, in the following ?

- Find x in an array of names
- Multiply two matrices with real entities
- Sort an array of numbers

Tutorial Problem No 11

What is/could be the input size, in the following ?

- Find x in an array of names
- Multiply two matrices with real entities
- Sort an array of numbers
- Traverse a binary tree

Tutorial Problem No 11

What is/could be the input size, in the following ?

- Find x in an array of names
- Multiply two matrices with real entities
- Sort an array of numbers
- Traverse a binary tree
- Solve a problem concerning graphs

Tutorial Problem No 12, 13

- Design the recursive version of the Fibonacci algorithm and only obtain the recurrence relation.

Tutorial Problem No 12, 13

- Design the recursive version of the Fibonacci algorithm and only obtain the recurrence relation.
- Design an algorithm for matrix addition and analyze its time complexity.

Tutorial Problem No 14

Find the cost of execution of the following code snippet

```
for i = 1 to n
        for j = 1 to i
                x = x + 1
```

Tutorial Problem No 15

Find the cost of execution of the following code snippet

```
j=n
while (j >= 1){

        for i = 1 to j
                x = x + 1

        j = n/2
}
```

Tutorial Problems 16

Find the cost of execution of the following code snippet

```
for i = 1 to n
        for j = 1 to i
                        for k = 1 to i

                                x = x + 1
```

Tutorial Problems 17

Find the cost of execution of the following code snippet

```
i=n
while (i >= 1){
        for j = 1 to n

                x = x + 1

        i = i/2
}
```

Tutorial Problem No 18

- Write an algorithm EXPONENT(a, n) to find an using an appropriate method. Analyze the asymptotic complexity of the algorithm and compare it with the conventional method to do so.

Tutorial Problem No 18 ...

```
1.      let ans = 1
2.      divide 2 into m giving quotient q &
remainder r
3.      if r = 1
4.              then ans = ans * x

5.      if q = 0
6.              goto exit

7.      let m = q
8.      let x = x * x
9.      goto step 2
10.     exit
```

Analysis of BINEXPONENT

Analysis

- Which are the vital steps ?

Analysis of BINEXPONENT

Analysis

- Which are the vital steps ?
- Can we say in general, how many times the multiplication in step 4 is performed?

Analysis of BINEXPONENT

Analysis

- Which are the vital steps ?
- Can we say in general, how many times the multiplication in step 4 is performed?
- The complexity turns out to be $3 + 3lgm$

Analysis of BINEXPONENT

Analysis

- Which are the vital steps ?
- Can we say in general, how many times the multiplication in step 4 is performed?
- The complexity turns out to be $3 + 3lgm$
- Compare the above with traditional algorithm

Analysis of BINEXPONENT

Analysis

- Which are the vital steps ?
- Can we say in general, how many times the multiplication in step 4 is performed?
- The complexity turns out to be $3 + 3lgm$
- Compare the above with traditional algorithm
- Effect of doubling the input size

Blank