***Lecture Two***

***Section* 2.1 − Sequences and Summations**

***Sequences***

***Definition***

A sequence is a function from a subset of the set of integers (usually either the set {0, 1, 2, …} or the set {1, 2, 3, …} to a set *S*. We use the notation  to denote the image of the integer *n*. We call  a term of the sequence.

The sequence , where 

The list of the terms of this sequence: 

***Definition***

A ***geometric progression*** is a sequence of the form  where the *initial* term *a* and the ***common ratio*** *r* are real numbers.

The common ratio for:  is 

***Definition***

An ***arithmetic progression*** is a sequence of the form  where the *initial* term *a* and the ***common difference*** *d* are real numbers.

**Recurrence Relations**

***Definition***

A ***recurrence relation*** for the sequence  is an equation that expresses  in terms of one or more of the previous terms of the sequence, namely, , for all integers *n* with , where  is a nonnegative integer. A sequence is called a solution of a recurrence relation if its terms satisfy the recurrence relation. (A recurrence relation is said to *recursively define* a sequence.)

***Example***

Let  be a sequence that satisfies the recurrence relation  for  and suppose that . What are , , and ?

***Solution***







***Example***

Let  be a sequence that satisfies the recurrence relation  for  and suppose that  and . What are , and ?

***Solution***





***Definition***

The Fibonacci sequence,  is defined by the initial conditions  and the recurrence relation



***Example***

Find the Fibonacci number 

***Solution***











***Example***

Determine whether the sequence , where  for every nonnegative integer *n*, is a solution of the recurrence relation  for *n* = 2, 3, 4, . . . . Answer the same question whenere  and where 

***Solution***

Suppose that . Then, for ,





 Is a solution of the recurrence relation

Suppose that . Then, for ,

 or 







 Is ***not*** a solution of the recurrence relation

Suppose that . Then, for ,



Is a solution of the recurrence relation

***Example***

Find the formalc for the sequences with the following first five terms:

1. 
2. 1, 3, 5, 7, 9
3. 1, −1, 1, −1, 1

***Solution***

1. The sequence with , *n* = 0, 1, 2, …, This proposed sequence is a geometric progression with  and .
2. Each term is obtained by adding 2 to the previous term, The sequence with , *n* = 0, 1, 2, …, This proposed sequence is an arithmetic progression with  and .
3. The terms alternate between 1 and −1, The sequence with , *n* = 0, 1, 2, …, This proposed sequence is an geometric progression with  and .

***Example***

How can we produce the terms of a sequence if the first 10 terms are 1, 2, 2, 3, 3, 3, 4, 4, 4, 4?

***Solution***

In this sequence, the integer 1 appears once, the integer 2 appears twice, the integer 3 appears three times, the integer 4 appears four times. A reasonable rule for generating this sequence is that the integer *n* appears exactly *n* times.

The sequence generated this is possible match.

***Example***

How can we produce the terms of a sequence if the first 10 terms are 5, 11, 17, 23, 29, 35, 41, 47, 53, 59?

***Solution***



The sequence can be obtained by adding 6 to previous term. This produce to .

This sequence is an arithmetic progression with  and .

***Example***

How can we produce the terms of a sequence if the first 10 terms are 1, 3, 4, 7, 11, 18, 29, 47, 76, 123?

***Solution***

4 = 1 + 3

7 = 4 + 3

11 = 4 + 7

And so on. We can see that the third term is the sum of the two previous term.

The sequence is determined by the recurrence relation  with initial conditions  and .

|  |  |
| --- | --- |
| Some Useful Sequences | |
|  | 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, … |
|  | 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, … |
|  | 1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000, … |
|  | 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, … |
|  | 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, … |
|  | 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, … |
|  | 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, … |

***Summations***

To find the sum of many terms of an infinite sequence, it is easy to express using ***summation notation***.





The index of summation runs through all integers starting with its ***lower limit*** and ending with its ***upper limit***.

The large uppercase Greek letter ***sigma***, **Σ**, is used to denote summation.

***Example***

Use the summation notation to express the sun of the first 100 terms of the sequence , where  for *j* = 1, 2, 3, …

***Solution***



***Example***

What is the value of 

***Solution***







***Example***

What is the value of 

***Solution***







***Theorem***

If *a* and *r* are real numbers and , then



***Proof***

Let 





 ***Shifting the index of summation with ***











If *r* = 1, then the 

***Double summations***

Double summations arise in many contexts (as in the analysis of nested loops in computer programs). An example of a double summation is



To evaluate the double sum, first expand the inner summation and then continue by computing the outer summation









|  |  |
| --- | --- |
| Some Useful Summation Formulae | |
| Sum | Closed Form |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***Example***

What is the value of 

***Solution***



***Example***

What is the value of 

***Solution***

***Exercises Section* 2.1 − Sequences and Summations**

1. Find these terms of the sequence , where 



1. What is the term  of the sequence , if  equals



1. What are the terms  of the sequence , if  equals





1. Find at least three different sequences beginning with the terms 1, 2, 4 whose terms are generated by a simple formula or rule.
2. Find at least three different sequences beginning with the terms 3, 5, 7 whose terms are generated by a simple formula or rule.
3. Find the first five terms of the sequence defined by each of these recurrence relations and initial conditions.
4. 
5. 
6. 
7. 
8. 
9. Find the first six terms of the sequence defined by each of these recurrence relations and initial conditions.
10. 
11. 
12. 
13. 
14. 
15. Let 
16. Find 
17. Show that , , and 
18. Show that  for all integers *n* with 
19. Is the sequence  a solution of the recurrence relation  if
20. 
21. 
22. 
23. 
24. 
25. 
26. 
27. 
28. Is the sequence  a solution of the recurrence relation  if
29. 
30. 
31. 
32. 
33. A person deposits $1,000.00 in an account that yields 9% interest compounded annually.
34. Set up a recurrence relation for the amount in the account at the end of *n* years.
35. Find an explicit formula for the amount in the account at the end of *n* years.
36. How much money will the account contain after 100 years?
37. Suppose that the number of bacteria in a colony triples every hour.
38. Set up a recurrence relation for the number of bacteria after *n* hours have elapsed.
39. If 100 bacteria are used to begin new colony, how many bacteria will be in the colony in 10 hours?
40. A factory makes custom sports cars at an increasing rate. In the first month only one car is made, in the second month two cars are made, and so on, with n cars made in the *n*th month.
41. Set up a recurrence relation for the number of cars produced in the first *n* months by this factory.
42. How many cars are produced in the first year?
43. Find an explicit formula for the number of cars produced in the first *n* months by this factory
44. For each of these lists of integers, provide a simple formula or rule that generates the terms of an integer sequence that begins with the given list. Assuming that your formula or rule is correct, determine the next three terms of the sequence.
45. 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, …
46. 1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 8, 8, …
47. 1, 0, 2, 0, 4, 0, 8, 0, 16, 0, …
48. 3, 6, 12, 24, 48, 96, 192, …
49. 15, 8, 1, −6, −13, −20, −27, …
50. 3, 5, 8, 12, 17, 23, 30, 38, 47, …
51. 2, 16, 54, 128, 250, 432, 686, …
52. 2, 3, 7, 25, 121, 721, 5041, 40321, …
53. 3, 6, 11, 18, 27, 38, 51, 66, 83, 102, …
54. 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, …
55. 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, …

***Section* 2.2**  **Algorithms**

***Introduction***

***Definition***

An ***algorithm*** is a finite sequence of precise instructions for performing a computation or for solving a problem.

* A program is one type of algorithm

O All programs are algorithms

O Not all algorithms are programs!

* Directions to somebody’s house is an algorithm
* A recipe for cooking a cake is an algorithm
* The steps to compute the cosine of 90° is an algorithm

**Properties of Algorithms**

***Input***: An algorithm has input values from a specified set.

***Output***: From each set of input values an algorithm produces output values from a specified set. The output values are the solution to the problem.

***Definiteness***: The steps of an algorithm must be defined precisely.

***Correctness***: An algorithm should produce the correct output values for each set of input values.

***Finiteness***: An algorithm should produce the desired output after a finite (but perhaps large) number of steps for any input in the set.

***Effectiveness***: It must be possible to perform each step of an algorithm exactly and in a finite amount of time.

***Generality***: The procedure should be applicable for all problems of the desired form, not just for a particular set of input values.

***Algorithm* 1 − Finding the Maximum Element in a Finite Sequence**

Given a list, how do we find the maximum element in the list?

To express the algorithm, we’ll use pseudocode

* Pseudocode is kinda like a programming language, but not really

***Example***

Show that Algorithm 1 for finding the maximum element in a finite sequence of integers has all the properties listed.

***Solution***

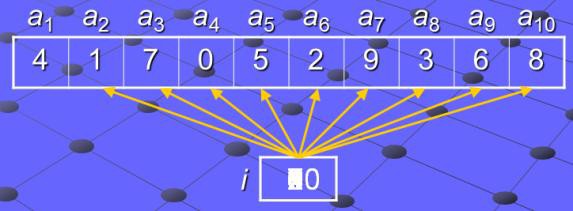
The input to Algorithm 1 is a sequence of integers. The output is the largest integer in the sequence. Each step of the algorithm is precisely defined, because only assignments, a finite loop, and conditional statements occur.

The values of the variable *max* equals the maximum terms when the algorithm terminates.

The initial value of max is the first term; as successive terms of the sequence are examined. This argument shows that when all the terms have been examined, max equal the value of the largest term and it will take *n* steps.

Algorithm 1 is general, because it can be used to find the maximum of any finite sequence of

integers.

**

***Procedure*** max 

*max***:=** *a*1

***for*** *i* := 2 to *n*

***if max***  **then** 

***return*** *max***{***max* is the largest element}

**Searching Algorithms**

Given a list, find a specific element in the list. There are two types:

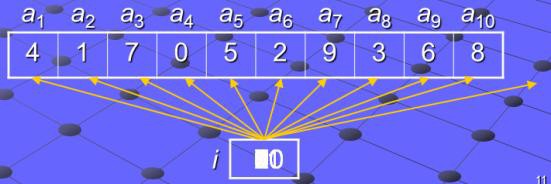
1. Linear search

2. Binary search

***Algorithm* 2 − Linear Search**

Given a list which does not have to be sorted, find element in the list

***procedure*** linear\_search (*x*: integer; : integers)

*i* := 1

***while*** ( *i* ≤ *n* and 



***if*** *i* ≤ *n* ***then*** *location* := *i*

***else*** *location* := 0

{*location* is the subscript of the term that equals *x*, or it is 0 if *x* is not found}

***Algorithm* 3 − Binary Search**

Given a list which *must* be sorted, find element in the list

***procedure*** linear\_search (*x*: integer; : increasing integers)

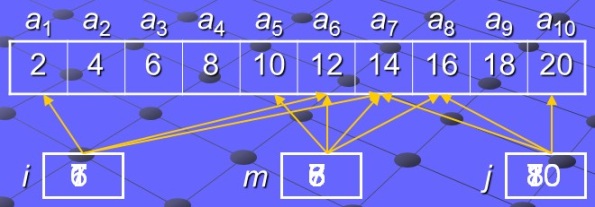
*i* := 1 { *i* is left endpoint of search interval }

*j* := *n* { *j* is right endpoint of search interval }

***while*** *i* < *j*

***begin***

 {*m* is the point in the middle }

***if ******then*** *i* := *m*+1

***else*** *j* := *m*

***end***

***if*** *x* =  ***then*** *location* := *i*

***else*** *location* := 0

{*location* is the subscript of the term that equals *x*, or it is 0 if *x* is not found}



***Sorting***

*Ordering* the elements of a list is a problem that occurs in many contexts. Suppose that we have a list of elements of a set. Suppose that we have a way to order elements of the set. ***Sorting*** is putting these elements into a list in which the elements are in increasing order.

There are two types:

* ***Bubble sort***
* ***Insertion sort***

**Bubble Sort**

The *bubble sort* is one of the simplest sorting algorithms, but not one of the most efficient. It takes successive elements and “***bubbles***” them up the list.

***Example***

Use the bubble sort to put 3, 2, 4, 1, 5 into increasing order.

***Solution***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| First Pass | 3 | 2 | 2 | 2 |
|  | 2 | 3 | 3 | 3 |
| 4 | 4 | 4 | 1 |
| 1 | 1 | 1 | 4 |
| 5 | 5 | 5 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Second Pass | 2 | 2 | 2 |
|  | 3 | 3 | 1 |
| 4 | 1 | 3 |
| 1 | 4 | 4 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Third Pass | 2 | 1 |
|  | 1 | 2 |
| 3 | 3 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Fourth Pass | 1 |
|  | 2 |
|  |
|  |
|  |

***Algorithm* 4 − Bubble**

***procedure*** bubblesort (: real numbers with )

***for*** *i* := 1 **to** *n* − 1

***for*** *j* := 1 **to** *n* − *i*

***if ***

***then*** interchange and 

**Bubble sort running time**

***Outer***for loop does *n* – 1 iterations

***Inner***for loop does:

*n* – 1 iterations the first time

*n* – 2 iterations the second time

***Total***: 

The bubble sort will take about  time.

**Insertion sort**

The *insertion sort* is another simple sorting algorithm, but inefficient. It starts with a list with one element, and inserts new elements into their proper place in the sorted part of the list

***Algorithm* 5 − Insertion sort**

***procedure*** insertion\_sort 

***for*** *j* := 2 **to** *n* ***take successive elements in the list***

***begin***

*i* := 1 ***find where that element should be***

***while ******in the sorted portion of the list***

*i* := *i* +1

** ***move all elements in the sorted portion of the list***

***for*** *k* := 0 **to** *j*-*i*-1 ***that are greater than the current element up by one***

******

** ***put the current element into it’s proper place***

***end *** ***in the sorted portion of the list***

The *insertion* sort will take about  time.

***Comparison* of Running Times**

***Searches***

 *Linear*: *n* steps

 *Binary*: steps

 *Binary* search is about as fast as you can get

**Sorts**

* *Bubble: * steps
* *Insertion: * steps
* There are other, more efficient, sorting techniques
* In principle, the fastest are heap sort, quick sort, and merge sort
* These each take  steps
* In practice, quick sort is the fastest, followed by merge sort

***Algorithm* 6 − Greedy Change-Making Algorithm**

***procedure*** change (: values of denominations of coins, where ; *n*: a positive integer)

***for*** *i* := 1 **to** *r*

  ***counts the coins of denomination *** ***used***

***While ***

 ***Add a coin of denomination ***



{ is the number of coins of denomination  in the change for *i* = 1, 2, …, *r*}

***Definition***

If *n* is a positive integer, then *n* cents in change using quarters, dimes, nickels, and pennies using the fewest coins possible has at most two dimes, at most one nickel, at most four pennies, and cannot have two dimes and a nickel. The amount of change in dimes, nickels, and pennies cannot exceed 24 cents

***Theorem***

The greedy algorithm (6) produces change using the fewest coins possible.

***Exercises Section* 2.2**  **Algorithms**

**1.** List all the steps used by the Algorithm 1 to find the maximum of the list

1, 8, 12, 9, 11, 2, 14, 5, 10, 4.

**2.** Devise an algorithm that finds the sum of all the integers in a list.

**3.** Describe an algorithm that takes as an input a list of *n* integers and produces as output the largest difference obtained by subtracting an integer in the list from the one following it.

**4.** Describe an algorithm that takes as an input a list of *n* integers in non-decreasing order and produces the list of all values that occur more than once.

**5.** Describe an algorithm that takes as an input a list of *n* integers and finds the location of the last even integer in the list or returns 0 if there are no even integers in the list.

**6.** Describe an algorithm that interchanges the values of the variables *x* and *y*, using only assignments. What is the minimum number of assignment statements needed to do this?

**7.** List all the steps used to search for 9 in the sequence 1, 3, 4, 5, 6, 7, 9, 11 using

***a***) a linear search *b*) a binary earch

1. Describe an algorithm that inserts an integer *x* in the appropriate position into the list  of integers that are in increasing order.

***Section* 2.3 – Divisibility and Modular Arithmetics**

***Division***

***Definition***

If *a* and *b* are integers with , we say that *a* divides *b* if there is an integer *c* such that , or equivalently, if  is an integers. When *a* divides *b* we say that *a* is a factor or divisor of *b*, and that *b* is multiple of *a*. The notation  denotes that *a* divides *b*. We write  when *a* does not divide *b*.

***Example***

Determine whether  and whether 3 | 12.

***Solution***

We see that , because 7/3 is not integer.

3 | 12 because 12/3 = 4.

***Example***

Let *n* and *d* be positive integers. How many positive integers not exceeding *n* are divisible by *d*?

***Solution***

The positive integers divisible by *d* are all the integers of the form *dk*, where *k* is a positive integer. Hence, the number of positive integers divisible by *d* that do not exceed *n* equals the number of integers *k* with . Therefore, there are  positive integers not exceeding *n* that are divisible by *d*.

***Theroem***

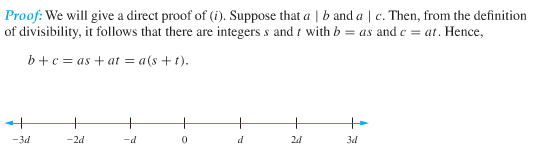
Let *a, b*, and *c* integers, where . Then

1. If *a* | *b* and *a* | *c,* then *a* | (*b + c*);
2. If *a* | *b,* then *a* | *bc* for all integers *c*;
3. If *a* | *b* and *b* | *c,* then *a* | *c*.

***Proof*** (***i***)

Suppose If *a* | *b* and *a* | *c.* Then, from the definition of divisibility, it follows that there are integers *s* and *t* with *b* = *as* and *c* = *at*. Hence,





Therefore, *a* divides *b* + *c*.

***Corollary***

If *a, b*, and *c* integers, where , such that *a | b* and *a | c*, then *a | mb + nc* whenever *m* and *n* are integers.

**The Division Algorithm**

***Theroem***

Let *a* be an integer and *d* a positive integer. Then there are unique integers *q* and *r*, with , such that .

***Definition***

In the equality given in the division algorithm,***d*** is called the ***divisor***, ***a*** called the ***dividend***, ***q*** is called the ***quotient***, and ***r*** is called the ***remainder***. This notation is used to express the quotient and remainder:



***Example***

What are the quotient and remainder when 101 is divided by 11?

***Solution***



Hence, the quotient when 101 is divided by 11 is 9 = 101 ***div*** 11,

and the remainder is 2 = 101 ***mod*** 11.

***Example***

What are the quotient and remainder when −11 is divided by 3?

***Solution***



Hence, the quotient when −11 is divided by 3 is −4 = −11 ***div*** 3,

and the remainder is 1 = −11 ***mod*** 3.

**Modular Arithmetic**

***Definition***

If *a* and *b* are integers and *m* is positive integer, then *a* is ***congruent*** to *b* ***modulo*** *m* if *m* divides *a* − *b*. We use the notation  to indicate that *a* is *congruent* to *b* *modulo* *m*. We say that  is a ***congruence*** and that m is its ***modulus*** (plural ***moduli***). If *a* and *b* are not congruent modulo *m*, we write 

***Theorem***

Let *a* and *b* be integers, and let *m* be a positive integer. Then  if and only if 

***Example***

Determine whether 17 is congruent to 5 modulo 6 and whether 24 and 14 are congruent modulo 6.

***Solution***

Because 6 divides 17 − 5 = 12, we see that .

24 − 14 = 10 is not divisible by 6, we see that 

***Theorem***

Let *m* be a positive integer. The integers *a* and *b* are congruent modulo *m* if and only if there is an integer *k* such that .

***Proof***

If  that implies by the definition of congruence to . Which is that there is an integer *k* such that .

Conversely, if there is an integer *k* such that , then . Hence, *m* divides *a* − *b*, so that 

***Theorem***

Let *m* be a positive integer. If  and , then



***Proof***

Using direct proof. Because  and , by the theorem that are integers *s* and *t* with . Hence,

And

***Corollary***

Let *a* and *b* be integers, and let *m* be a positive integer. Then



and



***Arithmetic Modulo m***

We define addition by:  and multiplication by 

***Exercises Section* 2.3 – Divisibility and Modular Arithmetics**

1. Does 17 divide each of these numbers?



1. Prove that if *a* is an integer other than 0, then



1. Show that if  and , where *a* and *b* are integers, then *a* = *b* or *a* = −*b*.
2. Show that if *a*, *b*, and *c* are integers, where  and , such that , then 
3. What are the quotient and remainder when
4. 19 is divided by 7?
5. −111 is divided by 11?
6. 789 is divided by 23?
7. 1001 is divided by 13?
8. 0 is divided by 19?
9. 3 is divided by 5?
10. −1 is divided by 3?
11. 4 is divided by 1?
12. What time does a 12-hour clock read
13. 80 hours after it reads 11:00?
14. 40 hours before it reads 12:00?
15. 100 hours after it reads 6:00?
16. What time does a 24-hour clock read
17. 100 hours after it reads 2:00?
18. 45 hours before it reads 12:00?
19. 168 hours after it reads 19:00?
20. Suppose *a* and *b* are integers, , and . Find the integer *c* with  such that
21. 
22. 
23. 
24. 
25. 
26. 
27. Suppose *a* and *b* are integers, , and . Find the integer *c* with  such that
28. 
29. 
30. 
31. 
32. Let *m* be a positive integer. Show that  if 
33. Show that if *n* and *k* are positive integers, then 
34. Evaluate these quantities
35. −17 ***mod*** 2
36. 144 ***mod*** 7
37. −101 ***mod*** 13
38. 199 ***mod*** 19
39. 13 ***mod*** 3
40. −97 ***mod*** 11
41. Find *a* **div** *m* and *a* **mod** *m* when
42. *a* = 228, *m* = 119
43. *a* = 9009, *m* = 223
44. *a* = −10101, *m* = 333
45. *a* = −765432, *m* = 38271
46. Find the integer *a* such that
47. 
48. 
49. 
50. 
51. 
52. Decide whether each of these integers is congruent to 5 modulo 17.



1. Find each of these values.
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 

***Section* 2.4 – Integer Representations and Algorithms**

**Representations of integers**

***Theorem***

Let *b* be an integer greater that 1. Then if *n* is a positive integer, it can be expressed uniquely in the form



Where *k* is a nonnegative integer  are nonnegative integers less than *b*, and 

***Example***

What is the decimal expansion of the integer that has  as its binary expansion?

***Solution***





**Octal and Hexadecimal Expansions**

Base 8 expansions are called ***octal*** expansions.

Base 16 expansions are called ***hexadecimal*** expansions.

***Example***

What is the decimal expansion of the number with octal expansion ?

***Solution***





***Example***

What is the decimal expansion of the number with hexadecimal expansion ?

***Solution***





**Base Conversion**

The algorithm for constructing the base *b* expansion of an integer *n*, divide *n* by *b* to obtain a quotient and remainder, that is,





***Example***

Find the octal expansion of 

***Solution***













***Example***

Find the hexadecimal expansion of 

***Solution***













***Example***

Find the bianry expansion of 

***Solution***



















|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Representation of the Integers* 0 *through* 15*.*** | | | | | | | | | | | | | | | | |
| ***Decimal*** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ***Hexadecimal*** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| ***Octal*** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| ***Binary*** | 0 | 1 | 10 | 11 | 100 | 101 | 110 | 111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

***Example***

Find the octal and hexadecimal expansions of 

***Solution***

Octal: 



Hexadecimal: 



***Example***

Find the binary expansions of  and 

***Solution***





**Algorithms for Integer Operations**

***Addition Algorithm***

To add *a* and *b*, first add their rightmost bits. This gives





***Example***

Add 

***Solution***

Therefore, 



***Example***

How many additions of bits are required to use Algorithm 2 to add two integers with *n* bits (or less) in their binary representations?

***Solution***

Two integers are added by successively adding pairs of bits. Adding each pair of bits and the carry requires two additions of bits. Thus, the total number of additions of bits used is less than twice the number of bits in the expansion. Hence, the number of additions of bits used by Algorithm 2 to add two *n*-bit integers is *O*(*n*).

**Multiplication Algorithm**





***Algorithm*: Multiplication of Integers**

**for***j* := 0 ***to*** *n* − 1

**if ** **then**  shifted *j* places

**else** 

*p* := 0

**for** *j* := 0 **to** *n* − 1

****

**Return** *p* {*p* is the value of *ab*}

***Example***

Find the product of  and 

***Solution***



**Modular Exponential**

It is important to find  efficiently, where *b, n* and *m* are large integers.



***Example***

Compute 

***Solution***











***Example***

Use Algorithm 5 to find 

***Solution***

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

This shows that following steps of Algorithm 5 produces the result 

***Exercises Section* 2.4 – Integer Representations and Algorithms**

1. Convert the decimal expansion of each of these integers to a binary expansion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. 321 | 1. 1023 | 1. 100632 | 1. 231 | 1. 4532 |

1. Convert binary the expansion of each of these integers to a decimal expansion

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Convert the binary expansion of each of these integers to an octal expansion

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Convert the octal expansion of each of these integers to a binary expansion

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

1. Convert the hexadecimal expansion of each of these integers to a binary expansion

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. Show that the binary expansion of a positive integer can be obtained from its hexadecimal expansion by translating each hexadecimal digit into a block of four binary digits.
2. Show that the binary expansion of a positive integer can be obtained from its octal expansion by translating each octal digit into a block of three binary digits.
3. Explain how to convert from binary to base 64 expansions and from base 64 expansions to binary expansions and from octal to base 64 expansions and from base 64 expansions to octal expansions
4. Find the sum and product of each of these pairs of numbers. Express your answers as a base 3 expansions

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Find the sum and product of each of these pairs of numbers. Express your answers as an octal expansion.

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Find the sum and product of each of these pairs of numbers. Express your answers as an hexadecimal expansion.

|  |  |
| --- | --- |
|  |  |
|  |  |

***Section* 2.5 – Primes and Greatest Common Divisors**

**Primes**

***Definition***

An integer *p* greater than 1 is called ***prime*** of the only positive factors of *p* are 1 or *p*.

A positive integer that is greater than 1 and is not prime is called composite.

***Example***

The integer 7 is prime because its only positive factors are 1 and 7.

The integer 9 is composite because its is divisible by 3.

***Theorem* − The Fundamental Theorem of Arithmetic**

Every integer greater than 1 can be written uniquely as a prime or as the product of two or more primes where the prime factors are written in order of nondecreasing size.

***Example***

Find the prime factorization of 100, 641, 999, and 1024.

***Solution***









**Trial Division**

***Theroem***

If *n* is a composite integer, then *n* has a prime divisor less than or equal to 

***Proof***

If *n* is composite, then it has a factor *a* (by definition of a composite integer) with . Hence, by the definition of a factor, we have .

If , then , which is a contradiction. Consequently,

. Because both *a* and *b* are divisors of *n*, we see that *n* has a positive divisor not exceeding . This divisor is either prime or, by the fundamental theorem of arithmetic, has a prime divisor less than itself. In either case, *n* has a prime divisor less than or equal to .

***Example***

Show that 101 is prime

***Solution***

The only primes not exceeding  are 2, 3, 5, 7. Because 101 is not divisible by 2, 3, 5, or 7 (the quotient of 101 and each of these integers is not an integer). It follows that 101 is prime.

***Example***

Find the prime factorization of 7007

***Solution***

None of the primes 2, 3, and 5 divides 7007. However, 7 divides 7007, with , and . Because 13 is prime, the procedure is completed.

It follows that the prime factorization is

**The Sieve of *Eratosthenes***

The ***Sieve******of******Eratosthenes***can be used to find all primes not exceeding a specified positive integer. For example, begin with the list of integers between 1 and 100.

* 1. Delete all the integers, other than 2, divisible by 2.
  2. Delete all the integers, other than 3, divisible by 3.
  3. Next, delete all the integers, other than 5, divisible by 5.
  4. Next, delete all the integers, other than 7, divisible by 7.
  5. Since all the remaining integers are not divisible by any of the previous integers, other than 1, the primes are: {2,3,7,11,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89, 97}

If an integer *n* is a composite integer, then it has a prime divisor less than or equal to .

To see this, note that if *n* = *ab*, then  .

*Trial division*, a very inefficient method of determining if a number *n* is prime, is to try every integer *i* ≤ and see if *n* is divisible by *i*.

|  |
| --- |
| ***The Sieve of Eratosthenes*** |
| table32.jpg |

**The infinitude of Primes**

It has long been known that there are infinitely many primes. This means that whenever  are the *n* smallest primes, we know there is a larger.

***Theorem***

There are infinitely may primes.

***Proof***: Assume finitely many primes: 

Let . Either *q* is prime or by the fundamental theorem of arithmetic it is a product of primes. But none of the primes **divides *q* since if | *q*, then  divide  *.*Hence*,* there is a prime not on the list It is either *q*, or if *q* is composite, it is a prime factor of *q*. This contradicts the assumption that  are all the primes. Consequently, there are infinitely many primes.

***Mersenne* Primes**

***Definition***

Prime numbers of the form  *,* where*p* is prime, are called ***Mersenne primes***.

***Example***

 are Mersenne primes.

 is not a Mersenne prime since 2047 = 23∙89.

There is an efficient test for determining if  is prime. The largest known prime numbers are Mersenne primes. 47 Mersenne primes were known, the largest is 243,112,609 *−* 1, which has nearly 13 million decimal digits.

**Distribution of Primes**

Mathematicians have been interested in the distribution of prime numbers among the positive integers. In the nineteenth century, the *prime number theorem* was proved whichgives an asymptotic estimate for the number of primes not exceeding *x*.

***Theorem*** − **Prime Number**

The ratio of the number of primes not exceeding *x* and approaches 1 as *x* grows without bound. (ln*x* is the natural logarithm of *x*),

The theorem tells us that the number of primes not exceeding *x*, can be approximated by .

The odds that a randomly selected positive integer less than *n* is prime are approximately 

**Greatest Common Divisor**

***Definition***

Let *a* and *b* be integers, not both zero. The largest integer *d* such that *d* | *a* and also *d* | *b* is called the greatest common divisor of *a* and *b*. The greatest common divisor of *a* and *b* is denoted by gcd(*a,b*).

One can find greatest common divisors of small numbers by inspection.

***Example***

What is the greatest common divisor of 24 and 36?

***Solution***

gcd(24,26) = 12

***Example***

What is the greatest common divisor of 17 and 22?

***Solution***

gcd(17,22) = 1

***Definition***

The integers *a* and *b* are *relatively prime* if their greatest common divisor is 1.

***Example*** 17 and 22

***Definition***

The integers are *pairwise* *relatively prime* if  whenever .

***Example***

Determine whether the integers 10, 17 and 21 are pairwise relatively prime.

***Solution***

Because gcd(10,17) = 1, gcd(10,21) = 1, and gcd(17,21) = 1, 10, 17, and 21 are pairwise relatively prime.

***Example***

Determine whether the integers 10, 19, and 24 are pairwise relatively prime.

***Solution***

Because gcd(10,24) = 2, 10, 19, and 24 are not pairwise relatively prime.

**Least Common Multiple**

***Definition***

The least common multiple of the positive integers *a* and *b* is the smallest positive integer that is divisible by both *a* and *b*. It is denoted by lcm(*a*,*b*).

The least common multiple can also be computed from the prime factorizations.



This number is divided by both *a* and *b* and no smaller number is divided by *a* and *b*.

***Example***





***Theorem***

Let *a* and *b* be positive integers. Then 

**Euclidean Algorithm**

The Euclidian algorithm is an efficient method for computing the greatest common divisor of two integers. It is based on the idea that gcd(*a*,*b*) is equal to gcd(*a*,*c*) when *a* > *b* and *c* is the remainder when *a* is divided by *b*.

***Lemma* 1**

Let , where *a, b, q*, and *r* are integers. Then 

***Proof***

Suppose that *d* divides both *a* and *b*. Then *d* also divides *a* − *bq* = *r*. Hence, any common divisor of *a* and *b* must also be any common divisor of *b* and *r*. Suppose that *d* divides both *b* and *r*. Then *d* also divides *bq* + *r* = *a*. Hence, any common divisor of *a* and *b* must also be a common divisor of *b* and *r*. Therefore, .

***Example***

Find *gcd* (91, 287)

***Solution***

287 = 91 ∙ 3 + 14 ***Divide* 287 *by* 91**

91 = 14 ∙ 6 + 7 ***Divide* 91 *by* 14**

14 = 7 ∙ 2 + 0 ***Divide* 14 *by* 7**

*gcd* (287, 91) = 7

***Example***

Find *gcd* (414, 662)

***Solution***

662 = 414 ∙ 1 + 248 ***Divide* 662 *by* 414**

414 = 248 ∙ 1 + 166 ***Divide* 414 *by* 248**

248 = 166 ∙ 1 + 82 ***Divide* 248 *by* 166**

166 = 82 ∙ 2 + 2 ***Divide* 166 *by* 82**

82 = 2 ∙ 41 + 0 ***Divide* 82 *by* 2**

*gcd*(414, 662) = 2

**Euclidean Algorithm**

**procedure** *gcd*(*a, b*: positive integers)

*x* := *a*

*x* := *b*

**while** *y ≠* 0

*r* := *x* **mod** *y*

*x* := *y*

*y* := *r*

**return** *x* {gcd(*a*,*b*) is *x*}

***GCDs* as Linear Combinations**

***Bézout’s Theorem***

If *a* and *b* are positive integers, then there exist integers *s* and *t* such that gcd(*a*,*b*) = *sa* + *tb*.

***Definition***

If *a* and *b* are positive integers, then integers *s* and *t* such that gcd(*a*,*b*) = *sa* + *tb* are called ***Bézout coefficients***of *a* and *b.* The equation gcd(*a*,*b*) = *sa* + *tb* is called***Bézout’s identity****.*

***Example***

Express gcd(252,198) = 18 as a linear combination of 252 and 198.

***Solution***

First use the Euclidean algorithm to show gcd(252,198) = 18

1. 252 = 1∙198 + 54
2. 198 = 3 ∙54 + 36
3. 54 = 1 ∙36 + 18
4. 36 = 2 ∙18

Now working backwards, from  *iii* and *i* above

18 = 54 − 1 ∙36

36 = 198 − 3 ∙54

Substituting the 2nd equation into the 1st yields:

18 = 54 − 1 ∙(198 − 3 ∙54 )= 4 ∙54 − 1 ∙198

Substituting 54 = 252 − 1 ∙198 (from *i*)) yields:

18 = 4 ∙(252 − 1 ∙198) − 1 ∙198 = 4 ∙252 − 5 ∙198

This method illustrated above is a two pass method. It first uses the Euclidian algorithm to find the gcd and then works backwards to express the gcd as a linear combination of the original two integers. A one pass method, called the ***extended Euclidean algorithm***, is developed in the exercises.

***Lemma* 2**

If *a*, *b*, and *c* are positive integers such that gcd(*a*, *b*) = 1 and *a* | *bc*, then *a* | *c*.

***Proof***

Assume gcd(*a*, *b*) = 1 and *a* | *bc*

Since gcd(*a*, *b*) = 1, by Bézout’s Theorem there are integers *s* and *t* such that  *sa* + *tb* = 1.

Multiplying both sides of the equation by *c*, yields *sac + tbc = c. a | tbc* and  *a* divides *sac + tbc* since *a | sac* and *a|tbc.* We conclude *a | c,* since *sac + tbc = c.*

***Lemma* 3**

If *p* is prime and , then for some *i*.

* Lemma 3 is crucial in the proof of the uniqueness of prime factorizations.

**Uniqueness of Prime Factorization**

We will prove that a prime factorization of a positive integer where the primes are in nondecreasing order is unique.

***Proof* (*by contradiction*)**

Suppose that the positive integer *n* can be written as a product of primes in two distinct ways:

 and 

Remove all common primes from the factorizations to get



By Lemma 3, it follows that  divides , for some *k,* contradicting the assumption that  and are distinct primes.

Hence, there can be at most one factorization of *n* into primes in nondecreasing order.

**Dividing Congruences by an Integer**

***Theorem***

Let m be a positive integer and let *a*, *b*, and *c* be integers. If *ac* ≡ *bc* (mod *m*) and gcd(*c,m*) = 1, then *a* ≡ *b* (mod *m*).

***Proof***

Since *ac* ≡ *bc* (mod *m*), *m* | *ac* − *bc* = *c*(*a* − *b*) by Lemma 2 and the fact that gcd(*c*,*m*) = 1, it follows that *m* | *a* − *b.* Hence, *a* ≡ *b* (mod *m*).

***Exercises Section* 2.5 – Primes and Greatest Common Divisors**

1. Determine whether each of these integers is prime.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  | 1. 93 | 1. 101 |
| 1. 107 | 1. 113 |  |  |  |

1. Find the prime factorization of each these integers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  | 1. 101 | 1. 143 |
| 1. 289 | 1. 899 |  |  |  |

1. Find the prime factorization of 10!
2. Show that if  is composite if *a* and *m* are integers greater than 1 and *m* is odd. [*Hint*: Show that *x* + 1 is a factor of the polynomial  if *m* is odd]
3. Show that if  is an odd prime, then  for some nonnegative integer *n*. [*Hint*: First show the polynomial identity  holds, where  and *t* is odd]
4. Which positive integers less than 12 are relatively prime to 12?
5. Which positive integers less than 30 are relatively prime to 30?
6. Determine whether the integers in each of these sets are pairwise relatively prime.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1. 14, 17, 85 | 1. 25, 41, 49, 64 | 1. 17, 18, 19, 23 |
| 1. 11, 15, 19 | 1. 14, 15, 21 | 1. 12, 17, 31, 37 | 1. 7, 8, 9, 11 |

1. We call a positive integer ***perfect*** if it equals the sum of its positive divisors other than itself
2. Show that 6 and 28 are perfect.
3. Show that  is a perfect number when  is prime
4. Show that if  is prime, then *n* is prime. *Hint*: Use the identity 
5. Determine whether each of these integers is prime, verifying some of Mersenne’s claims

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

1. What are the greatest common divisors of these pairs of integers?
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 
11. 
12. 
13. What is the least common multiple of each pair
14. 
15. 
16. 
17. 
18. 
19. 
20. 
21. 
22. 
23. 
24. 
25. Find gcd(1000, 625) and lcm(1000, 625) and verify that 
26. Find gcd(92928, 123552) and lcm(92928, 123552) and verify that 
27. Use the Euclidean algorithm to find

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Prove that the product of any three consecutive integers is divisible by 6.
2. Show that if *a, b*, and *m* are integers such that  and , then 
3. Prove or disprove that  is prime whenever *n* is a positive integer.

***Section* 2.6 – Applications of Congurences**

**Hashing Functions**

***Definition***

A *hashing function h* assigns memory location *h*(*k*) to the record that has *k* as its key.

A common hashing function is *h*(*k*) = *k* **mod** *m*, where *m* is the number of memory locations. Because this hashing function is onto, all memory locations are possible.

***Example***

Find the memory locations assigned by the hashing function *h*(*k*) = *k* **mod** 111 to the records of customers with Social Security numbers 064212848, 037149212, and 107405723.

***Solution***

This hashing function assigns the records of customers with social security numbers as keys to memory locations in the following manner:

*h*(064212848) = 064212848 **mod** 111 = 14

*h* (037149212) = 037149212 **mod** 111 = 65

*h* (107405723) = 107405723 **mod** 111 = 14,

But since location 14 is already occupied, the record is assigned to the next available position, which is 15.

The hashing function is not one-to-one as there are many more possible keys than memory locations. When more than one record is assigned to the same location, we say a *collision* occurs. Here a collision has been resolved by assigning the record to the first free location. For collision resolution, we can use a *linear probing function*:

, where *i* from 0 to m – 1.

There are many other methods of handling with collisions. You may cover these in a later CS course.

**Pseudorandom Numbers**

Randomly chosen numbers are needed for many purposes, including computer simulations.

***Pseudorandom numbers*** are not truly random since they are generated by systematic methods.

The ***linear congruential*** *method* is one commonly used procedure for generating pseudorandom numbers. Four integers are needed: the *modulus* *m*, the *multiplier* *a*, the *increment* *c*, and *seed*, with , 0 ≤ *c* < *m*, *.* We generate a sequence of pseudorandom numbers , with  for all *n*, by successively using the recursively defined function



***Example***

Find the sequence of pseudorandom numbers generated by the linear congruential method with modulus *m* = 9, multiplier *a* = 7, increment *c* = 4, and seed .

***Solution***

Compute the terms of the sequence by successively using the congruence , with .



















The sequence generated is 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, 7, 8, 6, 1, 2, 0, 4, 5, 3, … It repeats after generating 9 terms.

* Commonly, computers use a linear congruential generator with increment *c* = 0. This is called a *pure multiplicative generator*. Such a generator with modulus 231 − 1 and multiplier 75 = 16,807 generates 231 − 2 numbers before repeating.

**Check Digits: UPCs**

A common method of detecting errors in strings of digits is to add an extra digit at the end, which is evaluated using a function. If the final digit is not correct, then the string is assumed not to be correct.

***Example***

Retail products are identified by their *Universal Product Codes* (*UPC*s). Usually these have 12 decimal digits, the last one being the check digit. The check digit is determined by the congruence:



* + - 1. Suppose that the first 11 digits of the UPC are 79357343104. What is the check digit?
      2. Is 041331021641 a valid UPC?

***Solution***

* + - 1. 



. So, the check digit is 2.

* + - 1. 



Hence, 041331021641 is not a valid UPC.

***Exercises Section* 2.6 – Applications of Congurences**

1. Find the memory locations assigned by the hashing function *h*(*k*) = *k* **mod** 97 to the records of customers with Social Security numbers?

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

1. A parking lot has 31 visitor spaces, numbered from 0 to 30. Visitors are assigned parking spaces using the hashing function *h*(*k*) = *k* **mod** 31, where *k* is the number formed from the first three digits on a visitor’s license plate.
2. Which spaces are assigned by the hashing function to cars that have these first three digits on their license plates: 317, 918, 007, 100, 111, 310
3. Describe a procedure visitors should follow to find a free parking space, when the space they are assigned is occupied.
4. Find the sequence of pseudorandom numbers generated by the linear congruential generator
5.  with seed .
6.  with seed .
7. Find the sequence of pseudorandom numbers generated by using the pure multiplicative generator  with seed .
8. The first nine digits of the ISBN-10 of the European version of the fifth edition of this book are . What is the check digit for that book?
9. The ISBN-10 of the sixth edition of Elementary Number Theory and Its Applications is , where *Q* is a digit. Find the value of *Q*.
10. The USPS sells money orders identified by 11-digit number . The first ten digits identify the money order:  is a check digit that satisfies . Find the check digit for the USPS money orders that have identification number that start with these ten digits

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

1. Determine which single digit errors are detected by the USPS money order code.
2. Determine which transposition errors are detected by the USPS money order code.