|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hello, Welcome to Coding Challenges .co !  Coding Challenges .co is a collection of coding challenges completed by Max Plomer in the Matlab programming language as prep work for appacademy.io’s coding challenge. Feel free to contact Max at maxplomer@gmail.com or by phone at 203-945-8606 if you are aware of any opportunities in mechanical engineering.  DOWNLOAD SOLUTIONS ZIP FILE  **Coding Challenge 01**  -Write a function pow(base,exponent) that takes two integers greater or equal to zero, and returns base raised to the exponent power. Only use loops and arithmetic operators (no exponent operator, ^ in matlab).  **Coding Challenge 02**  -Write a function sumfun(array) which takes an array of numbers and returns the sum of the numbers. Only use loops and arithmetic operators (no prebuilt sum function).  **Coding Challenge 03**  -Write a function isprime(num) which takes a number and returns true (or 1) if it is prime and false (or 0) if not prime.  -You may want to use the modulo/remainder operator that returns the remainder after division (rem() in matlab). If num is divisible by i then the remainder is zero.  **Coding Challenge 04**  -Write a function primes(max) that takes an integer, max, greater or equal to zero and returns an array of all prime numbers less than max.  **Coding Challenge 05**  -Write a function nearest\_larger(array,i) which takes an array and an index. The function should return another index j that should satisfy  (a) array(i)<array(j)  (b) there is no j2 closer to i than j  (c) in case of tie, chose earlier/left index  (d) if no number in array is larger, then return [ ] for matlab, nil for ruby  Example input/output  array = [2,3,4,8] i = 3 j = 4  array = [2,8,4,3] i = 3 j = 2  array = [2,6,4,8] i = 3 j = 2  array = [2,6,4,6] i = 3 j = 2  array = [8,2,4,3] i = 3 j = 1  array = [2,4,3,8] i = 2 j = 4  array = [2,6,4,8] i = 4 j = [ ]  **Coding Challenge 06**  -Write a function no\_repeats(year\_start, year\_end), which takes a range of years and outputs those years which do not have any repeated digits.  -You should write a helper function called no\_repeat(year) which returns true/false if a single year doesn’t have a repeat  Example input/output  no\_repeats(1234,1234) result = [1234]  no\_repeats(1123,1123) result = [ ]  no\_repeats(1980,1987) result = [1980,1982,1983,1984,1985,1986,1987]  **Coding Challenge 07**  -Write a function letter\_count(word) that takes a string and returns for the case of matlab a cell array of letters and an array of each letter’s frequency. Do not include spaces.  Example input/output  word = 'moon cat' letters = { 'a' 'c' 'm' 'n' 'o' 't'} frequency = [1,1,1,1,2,1]  **Coding Challenge 08**  -Write a function orderedvowels(str) that takes a string of lowercase words and returns a string with just the words containing all their vowels (excluding y) in alphabetical order. Vowels may be repeated  -You will want a helper function ordered\_vowel\_word that returns true/false is a word’s vowels are in order.  Example input/output  orderedvowels('amends') result = 'amends'  orderedvowels('this is a test of the vowel ordering system')  result = 'this is a test of the system'  **Coding Challenge 09**  -A change machine takes any coin of value N and returns 3 new coins, valued at N/2, N/3 and N/4 (rounding down). There is a coin for zero!  -Write a function wonkycoin(n) that returns the number of coins you are left with if you take all the non-zero coins and keep feeding them back into the machine until you are left with only zero-value coins  Example input/output  wonkycoin(1) result = 3  wonkycoin(5) result = 11  wonkycoin(0) result = 1  **Coding Challenge 10**  -Write a function morse\_code(str) that takes in a string with no numbers or punctuation and outputs the morse code for it. Output two spaces between words and one space between letters. For Ruby use a hash map to map letters to codes. You will want a helper function to handle a single word.  Example input/output  morse\_code('q') result = '--.-'  morse\_code('cat') result = '-.-. .- -'  morse\_code('cat in hat') result = '-.-. .- - .. -. .... .- -'  **Coding Challenge 11**  -Write a function word\_unscrambler that takes two inputs: a scrambled word and a dictionary of real words. Your program must then output all words that our scrambled word can unscramble to.  -To see if a string is an anagram of another, split both strings into arrays of letters. Sort the two arrays, if they are equal then they are anagrams.  Example input/output  word\_unscrambler('cat', {'tac'}) result = {‘tac’}  word\_unscrambler('cat', {'tom'}) result = []  word\_unscrambler('cat', {'tic' 'toc' 'tac' 'toe'})  result = {'tac'}  word\_unscrambler('cat', {'scatter' 'tac' 'ca'})  result = {'tac'}  word\_unscrambler('turn', {'numb' 'turn' 'runt' 'nurt'})  result = {'turn' 'runt' 'nurt'}  **Coding Challenge 12**  -Write a function rec\_intersection(rect1,rect2) that returns the intersection of two rectangles. Rectangles are represented as a pair of coordinate-pairs: the bottom-left and top-right coordinates given in [x,y] notation.  -Left-most x coordinate of intersection is the minimum of the left-most coordinate of each rectangle. Top-most y coordinate of the intersection is the minimum of the top most y coordinate of each rectangle.  Example input/output  rec\_intersection({[0 0] [2 1]}, {[1 0] [3 1]}) result = {[1 0] [2 1]}  rec\_intersection({[1 1] [2 2]}, {[0 0] [5 5]}) result = {[1 1] [2 2]}  rec\_intersection({[1 1] [2 2]}, {[4 4] [5 5]}) result = {[ ] [ ]}  rec\_intersection({[1 1] [5 4]}, {[2 2] [3 5]}) result = {[2 2] [3 4]}  **Coding Challenge 13**  -Write a function bubble\_sort(array) which will sort an array of integers using bubble sort method. The bubble sorting method goes through an array and swaps numbers if they are out of order. This is repeated until there is no more swaps.  Example input/output  bubble\_sort([]) result = []  bubble\_sort([1]) result = [1]  bubble\_sort([5, 4, 3, 2, 1]) result = [1, 2, 3, 4, 5]  **Coding Challenge 14**  -Write a function towerofhanoi(n) which solves the tower of Hanoi puzzle output to screen the moves required to solve the puzzle in the most efficient way possible.  -From Wikipedia:  It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:  (1) Only one disk can be moved at a time.  (2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.  (3) No disk may be placed on top of a smaller disk.  **Coding Challenge 15**  -Write a function factorial(num) that calculates the factorial of num.  -factorial of 5 = 5x4x3x2x1  -Use of factorial? The number of different configurations of a 52-deck of cards is factorial of 52  **Coding Challenge 16**  -Write a function getwordlist that allows the user to input a list of words, and when then enter is a blank word the program quits and outputs the list.  **Coding Challenge 17**  -Write a function ask\_recursively(question) that takes in a string question and keeps asking until you answer yes or no, output either true or false  **Coding Challenge 18**  -Write a function helloname\_lovely that ask for your name and then says what a lovely name if you name is max or taylor  **Coding Challenge 19**  -Write a function landwater(map,x,y) that take a map input, a table of o’s (ocean) and M’s (land), and also x and y starting point and outputs the continent size.  -The function should recursively call itself.  Example input/output  world = [ 'ooooooooooo'  'ooooMMooooo'  'ooooooooMMo'  'oooMoooooMo'  'oooMoMMoooo'  'ooooMMMMooo'  'oooMMMMMMMo'  'oooMMoMMMoo'  'ooooooMMooo'  'oMoooMooooo'  'ooooooooooo']  x = 5  y = 6  size = 23  **Coding Challenge 20**  -Write a function remove\_dups(array) that reads a sequence of integers and remove duplicates  Example input/output  remove\_dups([617,201,415,201]) result = [617,201,415]  **Coding Challenge 21**  -Write a function report\_dups(array) that reads a sequence of integers and outputs one copy of only the duplicates in a list of elements.  Example input/output  report\_dups([617,201,415,201,201]) result = [201]  **Coding Challenge 22**  -Write a function report\_dups\_num(array) that reads a sequence of integers and outputs one copy of only the duplicates in a list of elements, but also count the number of times an elements appears in a list.  Example input/output  report\_dups\_num([617,201,415,201,201]) result = [201] num = [3]  **Coding Challenge 23**  -Write a function remove\_adjacent\_dups(array) that reads a sequence of integers and write out the sequence with adjacent duplicates removed.  Example input/output  array = [1,1,2,2,2,3,1,4,4] result = [1,2,3,1,4]  **Coding Challenge 24**  -Write a function options\_choices\_list(options,choices) that takes as an input a number of choices you have to make and the number of options for each choice. For example you might be splitting a 52-card deck of cards into 3 piles, therefore the number of options would be 3 and number of choices would be 52. The number of different configurations would be options^choices. Make a table where each row is a different configuration. Use recursion.  Example input/output  options = 2  choices = 3  configurations = [1 1 1  1 1 2  1 2 1  1 2 2  2 1 1  2 1 2  2 2 1  2 2 2]  **Coding Challenge 25**  -Redo Challenge 24 but use “binary and” to generate a list that is accurate if you only choose 2 options, then go back and expand the list for multiple choices again using your function that calls “binary and”.  -This will be much more complicated than challenge 24 but much faster computing time.  **Coding Challenge 26**  -Redo Challenge 24 but use the method of generating all the numbers 0 through (choices^options)-1 and then converting them to base “options”. Then can just add 1 to convert 0 choice to 1 choice. This will only work for options less than or equal to 10. And is very slow but very simple.  **Coding Challenge 27**  -Write a function number\_shuffle(num) that given a number with distinct digits, return a sorted array of all the unique numbers that can be found with those digits  Example input/output  num = 123 result = [123,132,213,231,312,321]  **Coding Challenge 28**  -Write a function tshirtprinter(jobs,Npresses) where job is an array with print jobs of various sizes, and Npresses is a number of presses. You must schedule each printing job with one of the operational printing machines in such a manner that all tshirts are printed in the least amount of time.  Example input/output  jobs = [10,15,20,24,30,45,75]  Npresses = 2  result = {[10 24 75] [15 20 30 45]}  **Coding Challenge 29**  -Write a function writepolynomial(coeffs) that reads in an array of polynomial coefficients and prints a function runpolynomial(x) that returns the value of the polynomial expression at x  -if a coefficient is 1, it doesn’t get printed  -if a coefficient is negative you have to display something like “-2\*x^3” not “+-2\*x^3”  -if a coefficient is zero, nothing gets added to the output  -for x^1 the ^1 part gets omitted  -x^0=1, so we don’t need to display it  Example input/output  coeffs = [-3,-4,1,0,6,1]  function result=runpolynomial(x)  result=-3\*x^5-4\*x^4+x^3+6\*x+1;  end  **Coding Challenge 30**  Macintosh HD:private:var:folders:yj:p2zstgbx7gb44h77h4dlctv80000gn:T:TemporaryItems:short_circuit.jpg  -Write a function shortcircuit(P,R)that determines the redundant/unused resistors (you can just output the path of least resistance). For the sake of simplicity assume that electricity flows through the path of lease resistance. This problem in essence becomes the “traveling salesman problem” is you assume the resistances are distances and want to find the shortest path between A and G  Example input/output  P = ['AB';'AD';'BC';'BE';'DF';'DC';'CE';'CF';'EG';'FG']  R = [50,150,250,250,400,50,350,100,200,100]  Start = 'A'  End = 'G'  Result = 'ADCFG'  **Coding Challenge 31**  -Make a function converttomilitary(normaltime) that converts normal time to military time. And another convertfrommilitary(militarytime) that converts military time to normal time  Example input/output  normaltime = {11 51 'pm'}  militarytime = {23 51}  **Coding Challenge 32**  -Make a function converttominute(militarytime) that converts military time to minute time. And another convertfromminute(minutetime) that converts minute time to military time.  Example input/output  militarytime = {23 51}  minutetime = 1431  **Coding Challenge 33**  -Make a function averagetime(times) that takes in an array of times and returns the averagetime.  -You will want to shift the times in a certain way as to make sure the average between 11:59pm and 12:01 am is midnight and not noon!  Example input/output  times = { {11 51 'pm'}  {11 56 'pm'}  {12 01 'am'}  {12 06 'am'}  {12 11 'am'}}  result = {12 1 'am'}  **Coding Challenge 34**  -Make a function businesshours(job,time,date) that takes in a job in minutes, time and date job is dropped off. And output the time and date job will be ready.  -Business is open 9am to 3pm, open Friday 10am to 5pm, closed Sunday Wednesday and Christmas, open Christmas eve 8am to 1pm  Example input/output  job = 7\*60  time = {6 44 'am'}  date = {'Dec' 24 2010}  donetime = { 11 0 'am'}  donedate = {'Dec' 27 2010}  **Coding Challenge 35**  -Make a function solvemaze(maze) that takes in a maze as input and outputs how many steps it will take to solve (only up down left right no diagonal), if the maze is not solvable output an empty array.  Example input/output  maze = [ '#######################'  '# A #'  '# ################### #'  '# # # #'  '# ################### #'  '# B #'  '#######################'];  result = 15  **Coding Challenge 36**  -Write a function gameoflife(height,width) that simulates the “Game of Life”. You have a grid of cells in 2 dimensions. Each cell has 2 possible states, alive or dead. Each cell has 8 neighbors: above, below, left right, and the 4 diagonals.  -any life cell <2 live neighbors dies  -any life cell >3 live neighbors dies  -any dead cell with 3 live neighbors comes alive  -create a random initial state  -output the final state  -pretend the board is folded on itself so that the edges touch eachother  Example input/output  height = 5  width = 5  result = [1 0 1 0 0  0 0 0 1 1  0 0 0 0 0  0 0 0 0 0  0 0 0 1 1]  **Coding Challenge 37**  -Make a function shiftsubtitle(shift) that reads a “subrip” format subtitle file (subtitle.txt) and shifts the times forward and prints out the corrected file to output.txt  Example input/output  shift = {0 0 02 500} (0 hours 0 minutes 2 seconds 500 miliseconds)  subtitle.txt file:  645  01:31:51,210 --> 01:31:54,893  the government is implementing a new policy...  646  01:31:54,928 --> 01:31:57,664  In connection with a dramatic increase  in crime in certain neighbourhoods,  output.txt file:  645  01:31:53,710 --> 01:31:57,393  the government is implementing a new policy...  646  01:31:57,428 --> 01:32:00,164  In connection with a dramatic increase  in crime in certain neighbourhoods,  **Coding Challenge 38**  -Write a function bucketsort(array,N) that sorts an array of numbers 0 through 1 using N number of buckets. Result will be desending (high to low)  Example input/output  N = 101  array = [0.9575  0.9649  0.1576  0.9706  0.9572  0.4854  0.8003  0.1419]  result = [4 2 5 1 7 6 3 8]  **Coding Challenge 39**  -Write a function bucketsort\_truncate(array,N) that truncates to the N’th decimal place. Therefore N=1 is the same as having 11 buckets.  **Coding Challenge 40**  -Write a function calculatecubemoves(sqnum,movepoints,obstacles) that calculates the possible squares that our “cube” character can move to on a 25x25 chess-like board.  -Each square has its own number x=1 y=1 sqnum=1, x=1 y=2 sqnum=26  -The “cube” character can move 1 square in the up/down/left/right directions as well as diagonals for each movepoint. You must respect the boarders of the board  -The piece is assumed to fly above obstacles, but cannot land on them, therefore remove obstacles from the results.  -The output is the possible moves (pm) for each step, no duplicates, and no moving backwards.  -You should have helper functions to check if x/y are on the board, and also to convert back and forth between sqnum and x/y  Example input/output  sqnum =105  movepoints = 3  obstacles = [133,158]  pm{1} = [129 130 131 79 80 81 104 106]  pm{2} = [153 154 155 156 157 53 54 55 56 57 78 103 128 82 107 132]  pm{3} = [177 178 179 180 181 182 183 27 28 29 30 31 32 33 52 77 102 127 152 58 83 108]  **Coding Challenge 41**  -Write a function calculatecylindermoves(sqnum,movepoints,obstacles) that calculates the possible squares that our “cylinder” character can move to on a 25x25 chess-like board.  -Each square has its own number x=1 y=1 sqnum=1, x=1 y=2 sqnum=26  -The “cylinder” character can move 1 square in the up/down/left/right directions but no diagonals for each movepoint. You must respect the boarders of the board  -The piece cannot fly above obstacles, therefore obstacles must not only be removed from the results, but will block other possible moves that obstacles block the path of.  -The output is the possible moves (pm) for each step, no duplicates, and no moving backwards.  -The output is also a path table of 2 columns and many rows, that will be used to calculate the path that our character must take. Obstacle spaces and duplicates can be left in the path table, because it doesn’t matter what specific path we take and we will never try to take a path to an obstacle.  -You should have helper functions to check if x/y are on the board, and also to convert back and forth between sqnum and x/y  Example input/output  sqnum = 105  movepoints = 3  obstacles = [106,130]  pm{1} = [80 104]  pm{2} = [55 79 81 103 129]  pm{3} = [30 54 56 78 82 102 128 154]  path = [105 106  105 104  105 80  105 130  80 81  80 79  80 55  80 105  104 105  104 103  104 79  104 129  55 56  55 54  55 30  55 80  79 80  79 78  79 54  79 104  81 82  81 80  81 56  81 106  103 104  103 102  103 78  103 128  129 130  129 128  129 104  129 154]  **Coding Challenge 42**  -Write a function findpath(path,start,end) using the path table from the previous challenge to create an array with the square numbers that our piece should travel to get from ‘start’ to ‘end’.  Example input/output  path output from challenge 41  start = 105  finish = 154  result = [105, 104, 129, 154]  **Coding Challenge 43**  -Create a selection sort <http://en.wikipedia.org/wiki/Selection_sort>  -In selection sort you go through the list from i=1 to i=length(list)-1  -For each i, go through the list j=i to j=length(list) find the lowest or same value as i  -if i is not the lowest or there is an element that is same value, swap them  **Coding Challenge 44**  -Create an insertion sort <http://en.wikipedia.org/wiki/Insertion_sort>  -In insertion sort you go through the list from i=2 to i=length(list)  -save list(i) to ‘value’ variable  -start with j=i-1 you then go in reverse, if list(j)>value move list(j) to list(j+1)  -j=j-1, quit when j=0 or list(j)<value  -put ‘value’ into list(j+1)  **Coding Challenge 45**  -Write a function called shuffleletters(word) that shuffles the letters of a word.  Example input/output  word = 'Solarcity'  result = 'iyrtlacoS'  **Coding Challenge 46**  -Write a function called shufflewords(sentence) that shuffles the words of a sentence.  Example input/output  sentence = 'Will I get the Solarcity job without a year of intense CAD experience'  result = 'a without experience the intense year CAD Will job Solarcity I get of'  **Coding Challenge 47**  -Create a word sorting function wordsort\_word2num(list) that sorts words alphabetically by converting them to a number, return just the indexes of the sorted list  -The word 'cab' converted to a fraction would be 0.030102  -By sorting the number in ascending order you will sort alphabetically  Example input/output  list = {'cat', 'dog', 'bird', 'snake', 'elephant', 'tiger'}  result = [ 3 1 2 5 4 6 ]  **Coding Challenge 48**  -Modify selectionSort to create a word sorting function selectionSort\_word(list) with only using the fact that ‘a’<’b’  -You should create a helper function to check if one word comes before another  -Also if two words have the same starting letters, but one is longer, for example ‘cat’ and ‘cats’, ‘cat’ comes before ‘cats’ alphabetically  **Coding Challenge 49**  -Create a sort that recursively sorts numbers by taking the minimum of the array and sending the rest to be sorted  **Coding Challenge 50**  Macintosh HD:Users:maxplomer:Desktop:Screen Shot 2014-05-31 at 10.08.09 PM.png  -There is a pattern so that you go right 1 and up 1, then left 2 and down 2, then right 3 and up 3, etc. The origin is square zero and then each step is a square higher as in the figure above.  -Make a function primepattern(x,y) that determines the number of steps you have to take to find the closest prime square. Steps can only be up/down/left/right.  -You will need a helper function that determines the square number for a given x and y  Example input/output  x = 0  y = -3  result = 2  **Coding Challenge 51**  Macintosh HD:Users:maxplomer:Desktop:Screen Shot 2014-06-01 at 10.12.55 AM.png  -Write a function hexagonpuzzle(vertexstart) that solves the tri-hexagonal puzzle  -There are 13 vertexes ‘0’ means white ‘1’ means black, they are numbered top-down left-right  -Upper left hexagon is hex1, upper right is hex2 and bottom is hex3, than can be rotated 1 CW or 0 CCW  -Input is the vertex state of the starting position  -Output is a list of moves with each row a hexagon number and then rotation direction  Example input/ouput (for example above)  vertexstart = [0 0 0 0 0 0 0 1 0 1 0 1 1]  pathoutput = [3 0  3 0  2 1]  **Coding Challenge 52**  -Write a function that lays mines in a field ‘xlen’ wide and ‘ylen’ tall, divided into unit-square cells. Maximum of one mine per cell, and no adjacent mines (diagonals are ok)  -‘blocked’ input variable is squares where a mine can not be planted. Each row of the blocked input is the row # and column # of a blocked spaced for the example blow ‘o’ is open and ‘b’ is blocked  example field:  o b b  o o o  o o o  -Output is the maximum number of mines and the number of configurations that can get you that maximum number  Example input/output  xlen = 3  ylen = 3  blocked = [1 2  1 3]  max\_mines = 4  num\_configs = 1  **Coding Challenge 53**  -Make a code that takes in a list of circle centers and output the area that is intersected by all circles  -Solve numerically by splitting area into dx and dy (easier but very slow)  -Solve by converting circles to polygons and take intersection that way  Example input/output  circs = [1 0  0 0  0 1  1 1]  areatot = 0.3151467  **Coding Challenge 54**  -Create a list of words in the document  -sort the list alphabetically  -remove duplicate words  Example input/output  document=['internet libertarians calling for the equal treatment of all '...  'internet data have camped out in front of the federal communications '...  'commission in washington saying they will not quit their '...  'occupy style protest until the regulator stands up for net neutrality']  result = { 'all' 'calling' 'camped' 'commission' 'communications' 'data' 'equal' 'federal' 'for' 'front' 'have' 'in' 'internet' 'libertarians' 'net' 'neutrality' 'not' 'occupy' 'of' 'out' 'protest' 'quit' 'regulator' 'saying' 'stands' 'style' 'the' 'their' 'they' 'treatment' 'until' 'up' 'washington' 'will'}  **Coding Challenge 55**  -Create a list of words in the document  -find words that appear more than once  -output these most used words alphabetically as well as an array with how many times they are used  Example input/output  document variable from above  topwords = { 'for' 'in' 'internet' 'of' 'the'}  times\_used = [ 2 2 2 2 3]  **Coding Challenge 56**  -calculate  -use formula  -There is a fast technique that does not use any calls to a factorial function or calculate the exponent term , by saving the previous term and simply multiplying the top by x and dividing the bottom by n.  **Coding Challenge 57**  -calculate cos(x)  -use formula  **Coding Challenge 58**  -calculate sin(x)  -use formula  **Coding Challenge 59**  -Write function to read two integers m and n and produce the desired result without using multiplication and division. Use repeated additions and subtractions instead.  -calculate m \* n (the result of multiplication)  -m and n might be positive or negative  **Coding Challenge 60**  -Write function to read two integers m and n and produce the desired result without using multiplication and division. Use repeated additions and subtractions instead.  -calculate m div n (the result of integer division), and also m mod n (the remainder after integer division)  -m and n are both positive  **Coding Challenge 61**  -Repeat coding challenge 60, but this time m and n can be negative, will make things more complicated.  **Coding Challenge 62**  -Implement addition of complex numbers in terms of operations on real numbers  **Coding Challenge 63**  -Implement multiplication of complex numbers in terms of operations on real numbers  **Coding Challenge 64**  -Implement floating point addition in terms of integer arithmetic operations (int32 converts reals into integers and idivide does integer division in matlab). Represent floating point numbers as pairs of integers (m,n), where m is a four-digit integer; that is, either 1000 m 9999, or -9999 m -1000. The pair (m, n) represents m \* . For example, (3142, 1) represents 3142 \* = 3.142.  -Make sure output does not need to be in (m,n) format.  -You can use, < >, or ‘while’ operators  **Coding Challenge 65**  -Direct relation graph is a table of values that shows how removing one variable from a system will cause error in other variables.  -For a transient chemical system, , is the error caused in the production rate of species A when removing species B  II = number of reactions  = Stoichiometric coefficient of A in reaction i  = rate of reaction i  = 1 if B is in reaction i, else = 0   |  |  |  |  | | --- | --- | --- | --- | |  | A | B | C | | A | 1 |  |  | | B |  | 1 |  | | C |  |  | 1 |   -Implement direct relation graph using 3 for-loops, in the inner most for loop which tests if species B is in reaction i .  -Avoid division by zero.  Example input/output  KK = 3  II = 2  netk = [1  1]  NU = [-1 0  1 -1  0 1]  rtable = [1.0000 1.0000 0  0.5000 1.0000 0.5000  0 1.0000 1.0000]  **Coding Challenge 66**  -Implement direct relation graph using a cell array of participating reactions for each species, as well as a cell array of participating species for each reaction.  **Coding Challenge 67**  -Create a recursive function to search the direction relation graph results  -sort the rab values from largest to smallest  -mark the starting (most important) species error as 1  -go through all edges A->B (B causes error in A)  -if A is marked and B is not marked: make threshold = edge value, run search(B)  -search(B): mark B, set B’s error as threshold, go through all edges, B->C, if C is not marked and edge B->C is greater than threshold then search(C)  -The algorithm is saying: if A is important, but C doesn’t cause a lot of error in A, but it causes more error in B than B causes in A, we will set the error of C to the error that B causes in A  Example input/output  KK  rtable from above  starting\_species=1 (species A)  error = [1.0000  1.0000  0.5000]  **Coding Challenge 68**  -Create a non-recursive version of search algorithm |