1. **String Compression implement a method to perform string compressionThe code to implement this is https://www.educative.io/answers/string-compression-using-run-length-encoding**

**def string\_compression(string):**

**compressed\_string = ""**

**count = 1**

**# Add in first character**

**compressed\_string += string[0]**

**# Iterate through loop, skipping last one**

**for i in range(len(string)-1):**

**if(string[i] == string[i+1]):**

**count += 1**

**else:**

**if(count > 1):**

**# Add count to string**

**compressed\_string += str(count)**

**compressed\_string += string[i+1]**

**count = 1**

**# Add last count**

**if(count > 1):**

**compressed\_string += str(count)**

**return compressed\_string**

**print(string\_compression("AAABBBCCC"))**

**# Output: A3B3C3**

1. **LinkedList The link shows a program to find the nth element of a linked list https://www.geeksforgeeks.org/nth-node-from-the-end-of-a-linked-list/**

**The code will look something like this:**

**Node \* find\_nth\_element( LinkedList \*list, int n ) {**

**if (list == NULL)**

**return NULL;**

**// Move a pointer up n elements from head**

**Node \*nth\_node = list->head;**

**for(int i = 0; i < n; i++)**

**nth\_node = nth\_node->next;**

**// Now traverse list until nth node**

**Node \*traversal\_node = list->head;**

**while( nth\_node != NULL )**

**{**

**traversal\_node = traversal\_node->next;**

**nth\_node = nth\_node->next;**

**}**

**// Return nth element**

**return traversal\_node;**

**}**

**Stack minimum details of data structure is available in https://www.geeksforgeeks.org/stack-data-structure/ stach has function to push and pop can you also add function to minto the stack and it should also execute in O(1)**

This can be achieved by creating an additional stack to keep track of the minimum element currently in the primary stack. As each element is pushed onto the primary stack, the new element is compared with the element on the top of the ‘minimum’ stack and the smaller of the two is pushed onto the ‘minimum’ stack. When an element is popped off of the primary stack, the element on the top of the ‘minimum’ stack is also popped off. This operation of finding and removing the minimum element from the stack runs in O(1) time.

class Node:

# Constructor which assign argument to nade's value

def \_init\_(self, value):

self.value = value

self.next = None

# This method returns the string representation of the object.

def \_str\_(self):

return "Node({})".format(self.value)

# \_repr\_ is same as \_str\_

\_repr\_ = \_str\_

class Stack:

# Stack Constructor initialise top of stack and counter.

def \_init\_(self):

self.top = None

self.count = 0

self.minimum = None

# This method returns the string representation of the object (stack).

def \_str\_(self):

temp = self.top

out = []

while temp:

out.append(str(temp.value))

temp = temp.next

out = '\n'.join(out)

return ('Top {} \n\nStack :\n{}'.format(self.top, out))

# \_repr\_ is same as \_str\_

\_repr\_ = \_str\_

def getMin(self):

if self.top is None:

return "Stack is empty"

else:

print("Minimum Element in the stack is: {}" .format(self.minimum))

# Method to check if Stack is Empty or not

def isEmpty(self):

# If top equals to None then stack is empty

if self.top == None:

return True

else:

# If top not equal to None then stack is empty

return False

def \_len\_(self):

self.count = 0

tempNode = self.top

while tempNode:

tempNode = tempNode.next

self.count += 1

return self.count

def peek(self):

if self.top is None:

print("Stack is empty")

else:

if self.top.value < self.minimum:

print("Top Most Element is: {}" .format(self.minimum))

else:

print("Top Most Element is: {}" .format(self.top.value))

def push(self, value):

if self.top is None:

self.top = Node(value)

self.minimum = value

elif value < self.minimum:

temp = (2 \* value) - self.minimum

new\_node = Node(temp)

new\_node.next = self.top

self.top = new\_node

self.minimum = value

else:

new\_node = Node(value)

new\_node.next = self.top

self.top = new\_node

print("Number Inserted: {}" .format(value))

def pop(self):

if self.top is None:

print("Stack is empty")

else:

removedNode = self.top.value

self.top = self.top.next

if removedNode < self.minimum:

print("Top Most Element Removed :{} " .format(self.minimum))

self.minimum = ((2 \* self.minimum) - removedNode)

else:

print("Top Most Element Removed : {}" .format(removedNode))

if \_name\_ == '\_main\_':

stack = Stack()

# Function calls

stack.push(3)

stack.push(5)

stack.getMin()

stack.push(2)

stack.push(1)

stack.getMin()

stack.pop()

stack.getMin()

stack.pop()

stack.peek()

**D) given array of integer representing the elevation of a roof structure at various position each position is saparated by a unit length write a program to determine the amount of water that will be trapped on the roof after heavy rainfall example input:[2 1 3 0 1 2 3] Ans: ? go through the available link for this solution https://www.geeksforgeeks.org/trapping-rain-water/**

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

// Function to return the maximum

// water that can be stored

int maxWater(int arr[], int n)

{

// To store the maximum water

// that can be stored

int res = 0;

// For every element of the array

for (int i = 1; i < n - 1; i++) {

// Find the maximum element on its left

int left = arr[i];

for (int j = 0; j < i; j++)

left = max(left, arr[j]);

// Find the maximum element on its right

int right = arr[i];

for (int j = i + 1; j < n; j++)

right = max(right, arr[j]);

// Update the maximum water

res = res + (min(left, right) - arr[i]);

}

return res;

}

// Driver code

int main()

{

int arr[] = { 0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1 };

int n = sizeof(arr) / sizeof(arr[0]);

cout << maxWater(arr, n);

return 0;

}

**e) You will be given a list coin denominations that you can use to tender change to your customers, find the most optimum way to tender the exact change to your customers, the optimum is when you use the least number of coins.**

Greedy algorithm is an algorithmic technique in which current solutions are selected at each step in order to maximize the overall gain. It works on the principle of making the most optimal decision at every stage and thus, making an overall optimum solution. Dynamic programming is an approach used when the problem can be broken down into a series of overlapping sub-problems. It is used when multiple decisions at each point have to be considered and multiple solutions could be available. Thus, here the sub-problem of finding the least number of coins is broken down into a series of subproblems to make the solution most ideal.

The bonus question can be solved using a greedy algorithm. Greedy algorithms select the best solution at each step to give the overall most optimal solution. Therefore, in this case, the algorithm will take the current digit, iterate through the other digits and select the maximum number found in the iterated digits, leaving out the current digit. It will repeat this same step for every other digit until it is left with the maximum number possible. Therefore, the answer to this is yes, the above solution is part of a greedy algorithm.

// Recursive C++ program for

// coin change problem.

#include <bits/stdc++.h>

using namespace std;

// Returns the count of ways we can

// sum coins[0...n-1] coins to get sum "sum"

int count(int coins[], int n, int sum)

{

// If sum is 0 then there is 1 solution

// (do not include any coin)

if (sum == 0)

return 1;

// If sum is less than 0 then no

// solution exists

if (sum < 0)

return 0;

// If there are no coins and sum

// is greater than 0, then no

// solution exist

if (n <= 0)

return 0;

// count is sum of solutions (i)

// including coins[n-1] (ii) excluding coins[n-1]

return count(coins, n - 1, sum)

+ count(coins, n, sum - coins[n - 1]);

}

// Driver code

int main()

{

int i, j;

int coins[] = { 1, 2, 3 };

int n = sizeof(coins) / sizeof(coins[0]);

int sum = 4;

cout << " " << count(coins, n, sum);

return 0;

}

// This code is contributed by shivanisinghss2110

**What is dot product and cross product? Explain use cases of where dot product is used and cross product is used in graphics environment. Add links to places where you studied this information and get back with the understanding.**

**Dot and cross Products on Vectors:**

A quantity that is characterized not only by magnitude but also by its direction, is called a vector. Velocity, force, acceleration, momentum, etc. are vectors.

Vectors can be multiplied in two ways:

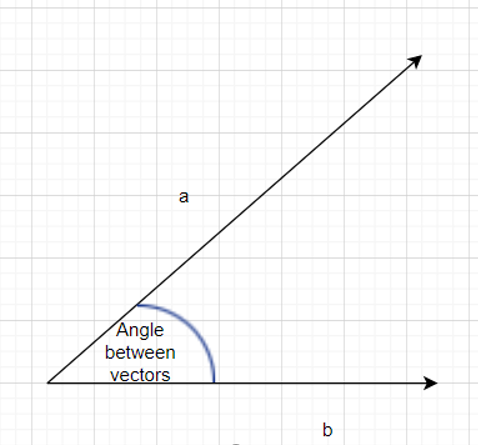
Scalar product or Dot product

Vector Product or Cross product

Scalar Product/Dot Product of Vectors

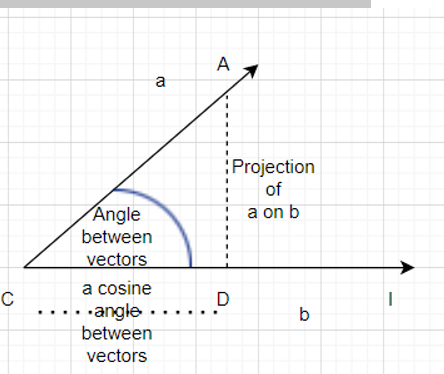
The resultant of scalar product/dot product of two vectors is always a scalar quantity. Consider two vectors a and b. The scalar product is calculated as the product of magnitudes of a, b, and cosine of the angle between these vectors.

Here, |a| = magnitude of vector a |b| = magnitude of vector b α = angle between the vectors



**Projection of one vector on other Vector**

Vector **a** can be projected on the line l as shown below:



It is clear from the above figure that we can project one vector over another vector. AC is the magnitude of vector A. In the above figure, AD is drawn perpendicular to line l. CD represents the projection of vector **a** on vector**b**.

Triangle ACD is thus a right-angled triangle, and we can apply trigonometric formulae.

*If α is the measure of angle ACD, then*

*cos α = CD/AC*

*Or,****CD = AC cos α***

From the figure, it is clear that CD is the projection of vector a on vector b

So, we can conclude that one vector can be projected over the other vector by the cosine of the angle between them.

**Properties of Scalar product:**

* Scalar product of two vectors is always a real number (scalar).
* Scalar product is commutative i.e. a.b =b.a= |a||b| cos α
* If α is 90° then Scalar product is zero as cos(90) = 0. So, the scalar product of unit vectors in x, y directions is 0.
* If α is 0° then the scalar product is the product of magnitudes of **a** and **b** |a||b|.
* Scalar product of a unit vector with itself is 1.
* Scalar product of a vector a with itself is |a|2
* If α is 1800, the scalar product for vectors a and b is -|a||b|
* Scalar product is distributive over addition

**a.**(**b** + **c**) = **a.b** + **a.c**

* For any scalar k and m then,

               l**a.**(m **b**) = km **a.b**

* If the component form of the vectors is given as:

**a**= a1x + a2y + a3z

**b**= b1x + b2y + b3z

           then the scalar product is given as

**a.b** = a1b1 + a2b2 + a3b3

* The scalar product is zero in the following cases:
  + The magnitude of vector a is zero
  + The magnitude of vector b is zero
  + Vectors a and b are perpendicular to each other

**Inequalities Based on Dot Product**

**Cauchy – Schwartz inequality**

According to this principle, for any two vectors **a** and **b**, the magnitude of the dot product is always less than or equal to the product of magnitudes of vector a and vector b

**|a.b|**≤**|a| |b|**

**Proof:**

*Since, a.b = |a| |b| cos α*

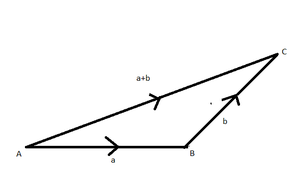
*We know that 0 < cos α < 1*

*So, we conclude that |a.b| ≤ |a| |b|*

**Triangle Inequality**

For any two vectors **a** and **b**, we always have

|**a**+ **b**| ≤ |**a**| + | **b**|



*Triangle inequality*

**Proof:**

*|****a****+****b****|2=|****a****+****b****||****a****+****b****|*

*=****a.a****+****a.b****+****b.a****+****b.b***

*= |****a****|2+ 2****a.b****+|****b****|2(dot product is commutative)*

*≤ |****a****|2+ 2|****a||b****| + |****b****|2*

*≤ (****|a****| + |****b|****)2*

*This proves that |****a****+****b****| ≤ |****a****| + |****b|***

**Examples of Dot Product of Vectors**

**Question 1. Consider two vectors such that |a|=6 and |b|=3 and α = 60°. Find their dot product.**

**Solution:**

***a.b****= |a| |b| cos α*

*So,****a.b****= 6.3.cos(60°)*

*=18(1/2)*

***a.b = 9***

**Question 2. Prove that the vectors a = 3i+j-4k and vector b = 8i-8j+4k are perpendicular.**

**Solution**:

*We know that the vectors are perpendicular if their dot product is zero*

***a.b****= (3i+j-4k)(  8i-8j+4k)*

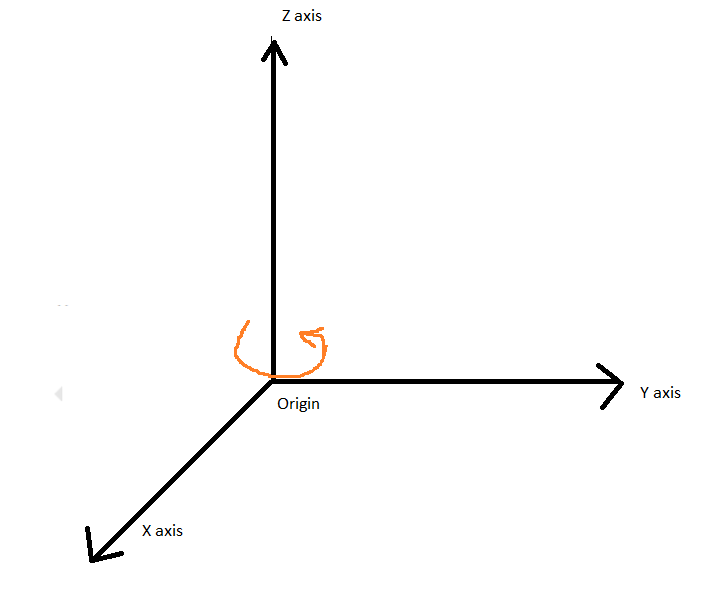
*= (3)(8) +(1)(-8)+(-4)(4)*

*=24-8-16 =0*

***Since the scalar product is zero, we can conclude that the vectors are perpendicular to each other.***

**Cross Product/Vector Product of Vectors**

Readers are already familiar with a three-dimensional right-handed rectangular coordinate system. In this system, a counterclockwise rotation of the x-axis into the positive y-axis indicates that a right-handed (standard) screw would advance in the direction of the positive z-axis as shown in the figure.



*3D Rectangular coordinate system*

The vector product of two vectors **a** and **b**with an angle α between them is mathematically calculated as

**a × b = |a| |b| sin α**

It is to be noted that the cross product is a vector with a specified direction. The resultant is always perpendicular to both a and b.

In case a and b are parallel vectors, the resultant shall be zero as sin(0) = 0

**Properties of Cross Product:**

* Cross Product generates a vector quantity. The resultant is always perpendicular to both a and b.
* Cross Product of parallel vectors/collinear vectors is zero as sin(0) = 0.

**i × i = j × j = k × k = 0**

* Cross product of two mutually perpendicular vectors with unit magnitude each is unity. (Since sin(0)=1)
* Cross product is not commutative.

**a × b is not equal to b × a**

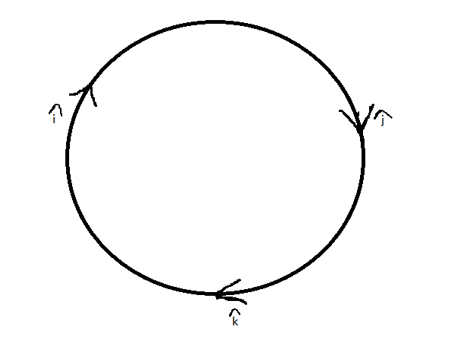
* Cross product is distributive over addition

**a ×** (**b**+ **c**) = **a** **× b**+ **a** **×** **c**

* If k is a scalar then,

**k(a × b) = k(a) × b = a × k(b)**

* On moving in a clockwise direction and taking the cross product of any two pair of the unit vectors we get the third one and in an anticlockwise direction, we get the negative resultant.



*Cross product in clockwise and anticlockwise direction*

           The following results can be established:

           i × j = k     j × k = i       k × i = j

           j × i = -k   i × k= -j       k × j = -i

**Cross product in Determinant Form**

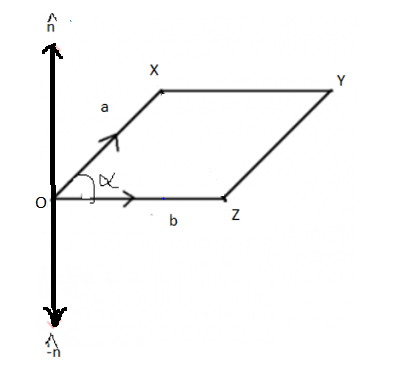
If the vector**a** is represented as **a = a1x + a2y + a3z** and vector **b** is represented as **b = b1x + b2y + b3z**

Then the cross product **a × b**can be computed using determinant form

**a × b** = x(a2b3 – b2a3) + y(a3b1 – a1b3) + z(a1b2 – a2b1)

If a and b are the adjacent sides of the parallelogram OXYZ and α is the angle between the vectors a and b.

Then the area of the parallelogram is given by |**a × b**| = |a| |b|sin.α



*Vectors a and b as adjacent sides of a parallelogram*

**Examples of Cross product of Vectors**

**Question 1. Find the cross product of two vectors a and b if their magnitudes are 5 and 10 respectively. Given that angle between then is 30°.**

**Solution:**

***a × b****= a.b.sin (30) = (5) (10) (1/2) = 25 perpendicular to****a****and****b***

**Question 2. Find the area of a parallelogram whose adjacent sides are**

**a = 4i+2j -3k**

**b= 2 i +j-4k**

**Solution**:

*The area is calculated by finding the cross product of adjacent sides*

*a × b = x(a2b3 – b2a3) + y(a3b1 – a1b3) + z(a1b2 – a2b1)*

*= i(-8+3) + j(-6+16) + k(4-2)*

*= -5i +10j + 2k*

*Therefore, the magnitude of area is*

**Application:**Dot products and cross products are extensively useful for engineering applications.

**Reference:** [**https://www.geeksforgeeks.org/dot-and-cross-products-on-vectors/**](https://www.geeksforgeeks.org/dot-and-cross-products-on-vectors/)

**g) Explain a piece of code that you wrote which you are proud of? If you have not written any code, please write your favorite subject in engineering studies. We can go deep into that subject**

Machine Learning

**h) Random crashes you are given a source code to test and it randomly crashes and it never place (you have attached a debugger and you find this). Explain what pect and how would you go about with isolating the cause.**

If the source code is randomly crashing and we are not able to pinpoint the exact cause of the crash, then it is necessary to narrow down the potential cause of the crash by utilizing a debugger. Through utilising a debugger, we are able to look at the state of the system at the time of the crash and see which parts of the code were executing at the time of the crash. This allows us to trace back the specific portion of code that could be causing the random crashing. Additionally, the debugger can be used to set break points and enable other features that would help isolate the cause of the crash. By doing these, we can analyze the possible causes of the crash more thoroughly and test to determine what can be done to prevent it from happening again.

**I declare that I have done the above work by myself and not worked with anyone or got help from any individual on the internet.**