Question 1 (20%): Cycle Detection

a)by exploring the graph to remove one cycle at a time as well as the vertices on that cycle,
with base case of three vertices each of them have two neighbors, this can form a cycle
and then deleted so no vertex left
assuming there are n vertices already form some cycles with each of them is even degree
if now there exist one vertex left and this vertex also obey the property of even degree
then this vertex either be a vertex in some paths since it provide a condition of at least one in degree
and one corresponding out degree, hence with the assumption it must inside some cycles
formed by n vertices or this vertex would be a starting and ending vertex of a path which is a cycle
therefore it is true that by exploring the graph to remove one cycle at a time there
will be no cycle left so all vertex must inside some cycles

$Question\ 2\ (20\%)$: $All\ Pairs\ Shortest\ Paths$

$since\ this\ is\ a\ undirected\ same\ weight\ graph$

$we\ can\ use\ BFS\ to\ get\ the\ shortest\ distance\ to\ each\ other\ vertex\ from\ a\ particular\ vertex$

$by\ running\ BFS\ for\ each\ vertex,\ each\ vertex\ take\ time\ O(V + E)$

$as\ there\ are\ V\ vertices,\ so\ the\ total\ time\ would\ be\ O(V^2 + VE)$

$since\ maximum\ E\ for\ undirected\ graph\ is\ V(V - 1)/2\ then\ O(VE) = O(V^3) > O(V^2)$

$so\ we\ can\ simplify\ to\ O(V^2 + VE) = O(VE)$

$code\ implementation$:

$let\ D\ be\ a\ set\ storing\ all\ pair\ of\ vertices\ shortest\ path$

$function\ BFS(s)$

 $for\ each\ vertex\ u \in V - \{s\}.$

 $u.color \leftarrow white\ ,\ D(s, u) \leftarrow \infty$

 $s.color \leftarrow grays$

 $D(s, s) \leftarrow 0$

 $initialize\ an\ empty\ queue\ Q$

 $enqueue\ s\ in\ Q$

 $while\ Q\ not\ empty$

  $p \leftarrow dequeue\ Q$

  $for\ v\ in\ adj[p]$

   $if\ v\ is\ white$

    $v.color = gray$

    $D(s, v) = D(s, p) + 1$

    $enqueue\ v\ in\ Q$

$for\ each\ u \in V$

 $BFS(u)$

$after\ running\ BFS\ for\ each\ node,\ we\ can\ retrieve\ the\ all\ pairs\ shortest\ path\ in\ D$

$Question\ 3\ (30\%)$: $Currency\ Exchange$

$converting\ this\ problem\ into\ a\ finding\ negative\ cycle$
$we\ could\ use\ bellman - ford\ algorithm\ to\ find\ a\ shortest\ path$
$in\ this\ case\ each\ currency\ is\ vertex\ and\ the\ exchange\ rate\ is\ edge\ with\ weight$
$but\ edge\ weight\ use\ in\ algorithm\ is\ a\ bit\ different\ from\ exchange\ weight$
$so\ need\ to\ convert\ it\ by\ using\ the\ logarithmic\ property$

$code\ implementation$:

$first\ by\ choosing\ a\ start\ vertex\ s\ from\ anyone\ currency$
$Let\ D\ be\ a\ array\ storing\ shortest\ path\ from\ s\ to\ other\ currencies\ x_i$
$Let\ W\ be\ a\ n\ by\ n\ matrix\ storing\ the\ weight\ of\ each\ edge$
$transferring\ each\ rate\ pair\ (x_i, x_j)\ in\ M\ to\ W$

$for\ each\ pair(x_i, x_j)$
$\qquad W(x_i, x_j) = - log(M(x_i, x_j))$

$and\ now\ have\ the\ condition\ of\ negative\ cycle$

$for\ loop\ n - 1\ times\ \#check\ shortest\ path\ with\ at\ most\ n - 1\ vertices$
$\qquad for\ each\ pair(x_i, x_j)\ \#do\ relaxation\ on\ each\ pair$
$\qquad\qquad if\ D(s, x_i) + W(x_i, x_j) < D(s, x_j)$
$\qquad\qquad\qquad D(s, x_j) = D(s, x_i) + W(x_i, x_j)$

$now\ D\ storing\ the\ shortest\ path\ of\ every\ vertices\ from\ s$
$by\ doing\ relaxation\ one\ more\ time\ we\ can\ know\ whether\ there\ is\ a\ negative\ cycle\ exist$

$for\ each\ pair(x_i, x_j)$
$\qquad if\ D(s, x_i) + W(x_i, x_j) < D(s, x_j)$
$\qquad\qquad\qquad return\ true$
$return\ false$

$by\ checking\ the\ negative\ cycle\ we\ can\ easily\ know\ if\ there\ exist\ a\ path\ of\ currencies\ can$
$make\ profit\ after\ exchanging\ currencies\ a\ cycle$

$Question\ 4\ (30\%)$: $Maximum\ Flow$

by dividing the problem into a 2x2 matrix with entry $a_{11}, a_{12}, a_{21}, a_{22}$

it is easy to observe that with matrix A property

$a_{11} + a_{21} = b_{11},\ a_{11} + a_{12} = b_{12}$ and $a_{21} + a_{22} = b_{21}$ and $a_{12} + a_{22} = b_{22}$

and these bs are non negative integers

now constructing the matrix B from A by modifying $a_{11}, a_{12}, a_{21}, a_{22}$

let numbers $s_{11}, s_{12}, s_{21}, s_{22} \in [0, 1)$

defining each s_{ij} is use to compensate the decimal places from a_{ij} of being an integer

so $a_{ij} + s_{ij} \in$ non negative integers

by maintaining the same sum b_{ij} we like to use $a_{11} + s_{11} + a_{21} - s_{11} = b_{11}$

it is easily can see that since integers are closed under addition operation

so $a_{21} - s_{11}$ is also an non negative integer

this can be also apply in column and the diagonal entry can remain the same by adding s_{11}

and is a integer by the property illustrated above

then now in the general form of n by m matrix

this can divide into row $[a_1 \text{ to } a_m]$, column$[a_1 \text{ to } a_n]$ and a sub matrix $n - 1, m - 1$

so can be recursively divided to base case of 2x2 matrix

each time known that there exist a number $s \in [0, 1)$ so that

$a_1 + s + sum([a_2 \text{ to } a_n]) - s = sum([a_1 \text{ to } a_n])$

such that $sum([a_2 \text{ to } a_n]) - s$ is an integer and continuing

$a_2 + a_3 ... + a_n - s + s_2$ for making a_2 to be an integer

$a_2 + s_2 - s + sum([a_3 \text{ to } a_n]) - s_2 = sum([a_2 \text{ to } a_n])$ is also an integer by the property proved above

then column can be apply the same method

such that all the entries in n by m matrix A can be changed to integers and making the same sum in every rows and column

therefore the matrix B must exist