Cryptarithmetic

```
  S E N D
+ M O R E
---------
M O N E Y
```

digits 0 to 9

S -> 2
e -> 5
n -> 3
d -> 0
m -> 6
o -> 1
r -> 9
y -> 8

a-5 e-9 m-3 o-1 p-4 r-7 t-0

team
pep
toppr

t e a m

t p e p

t o p p r

0 1 4 4 7

=>

0 9 5 3

4 9 4

1 4 4 7

a -> 5

e -> 9

m -> 3

o -> 1

p -> 4

r -> 7

t -> 0

```
  send
+ more
------
 money
```

used -> digits

usn: sendmory

```
y
r
o
m
d
n
e
s
```

```java
public static void solution(String unique, int idx,
                            HashMap<Character, Integer> charIntMap, boolean[] usedNumbers,
                            String s1, String s2, String s3) {

  if(idx == unique.length()) {
      if(isValid(charIntMap,s1,s2,s3) == true) {
          //print the mapping
          for(int i=0; i < 26;i++) {
              char ch = (char)('a' + i);

              if(charIntMap.containsKey(ch) == true) {
                  System.out.print(ch + "-" + charIntMap.get(ch) + " ");
              }
          }

          System.out.println();
      }

      return;
  }

  char ch = unique.charAt(idx);

  for(int d=0;d <= 9;d++) {
      if(usedNumbers[d] == false) {
          usedNumbers[d] = true;
          charIntMap.put(ch,d);
          solution(unique,idx+1,charIntMap,usedNumbers,s1,s2,s3);
          usedNumbers[d] = false;
          charIntMap.put(ch,-1);
      }
  }

}
```

a-2 c-7 e-9 p-8 t-0 u-1

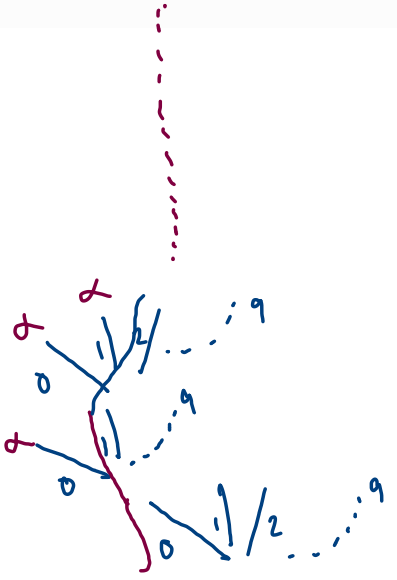
t->0 u->4
e->1 p->5
a->2
c->3

p u t
5 4 0

num = 540

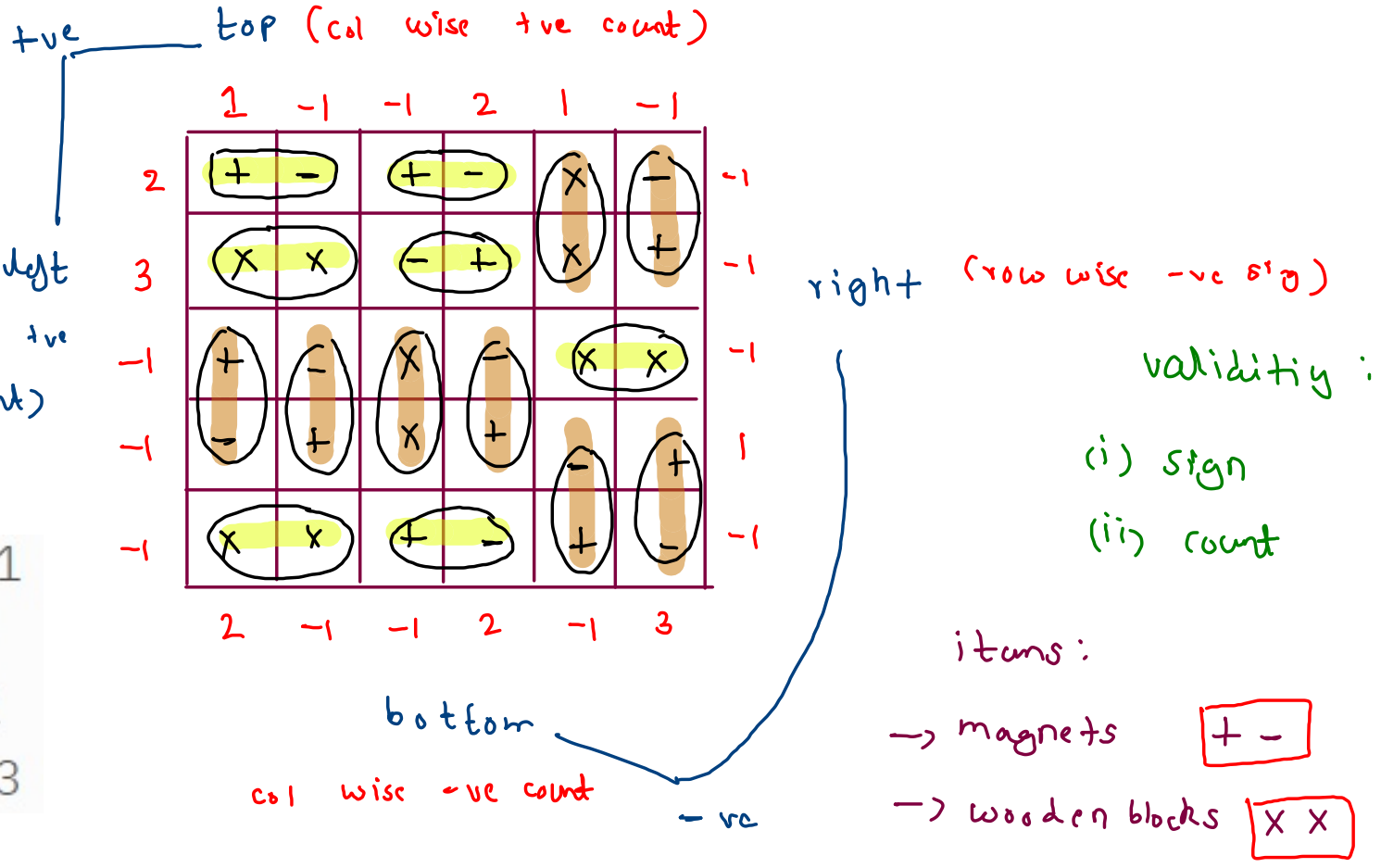
```java
public static int stringToInt(HashMap<Character,Integer>charIntmap,String s) {
    int num = 0;

    for(int i=0; i < s.length();i++) {
        char ch = s.charAt(i);
        int d = charIntmap.get(ch);

        num = num*10 + d;
    }

    return num;
}

public static boolean isValid(HashMap<Character,Integer>charIntmap,String s1,String s2,String s3) {
    int n1 = stringToInt(charIntmap,s1);
    int n2 = stringToInt(charIntmap,s2);
    int n3 = stringToInt(charIntmap,s3);

    return n1+n2 == n3;
}
```

12
345
540



Magnets

LRLRTT
LRLRBB
TTTTLR
BBBBTT
LRLRBB



top	1	-1	-1	2	1	-1
left	2	3	-1	-1	-1	
right	-1	-1	-1	1	-1	
bottom	2	-1	-1	2	-1	3

validity :

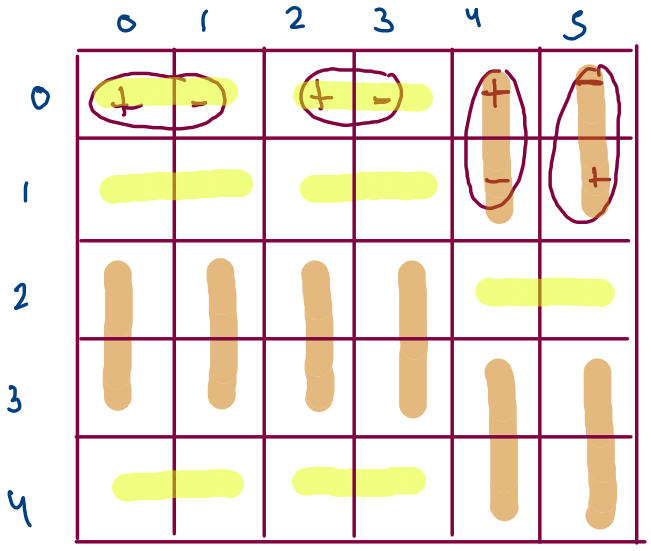
(i) sign

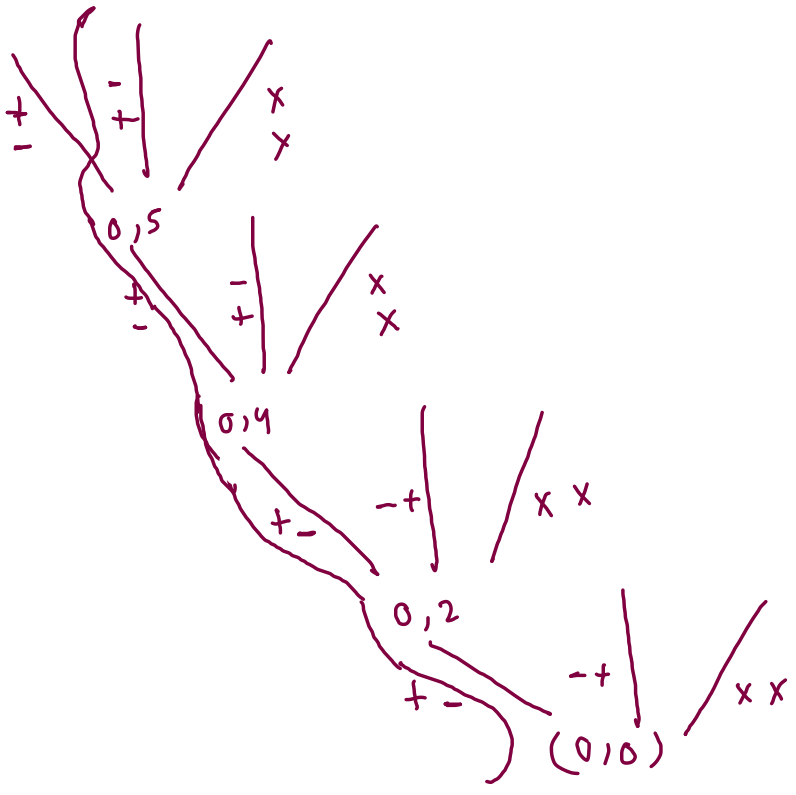
(ii) count

items :

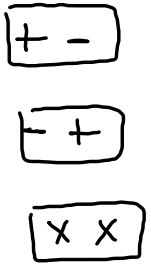
→ magnets + -

→ wooden blocks X X

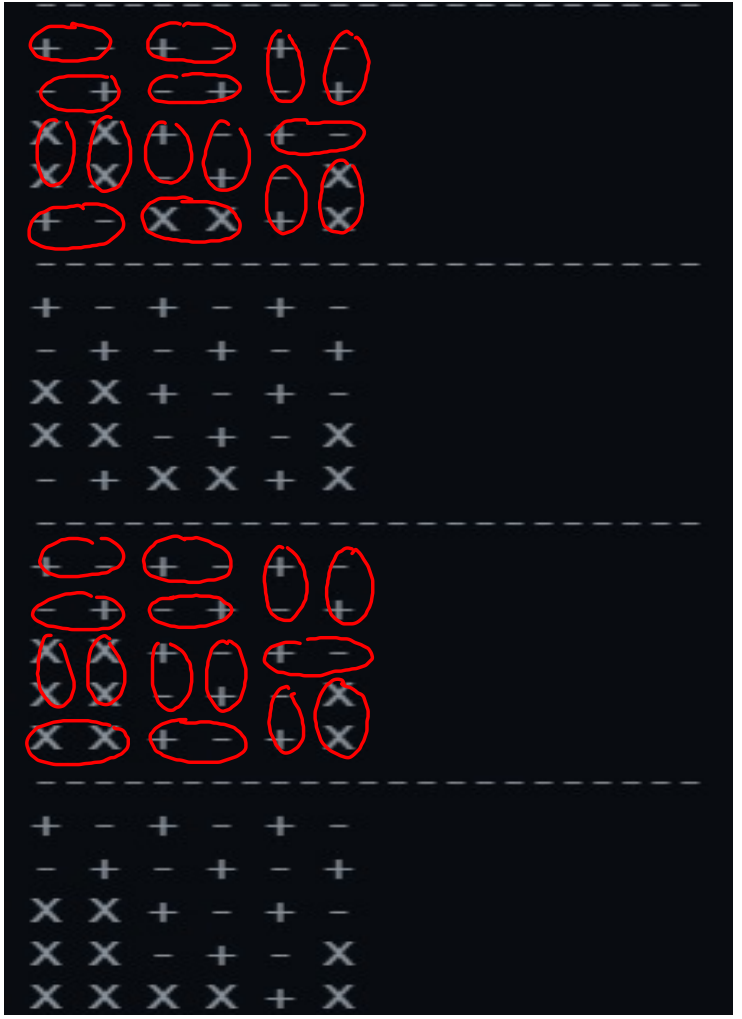




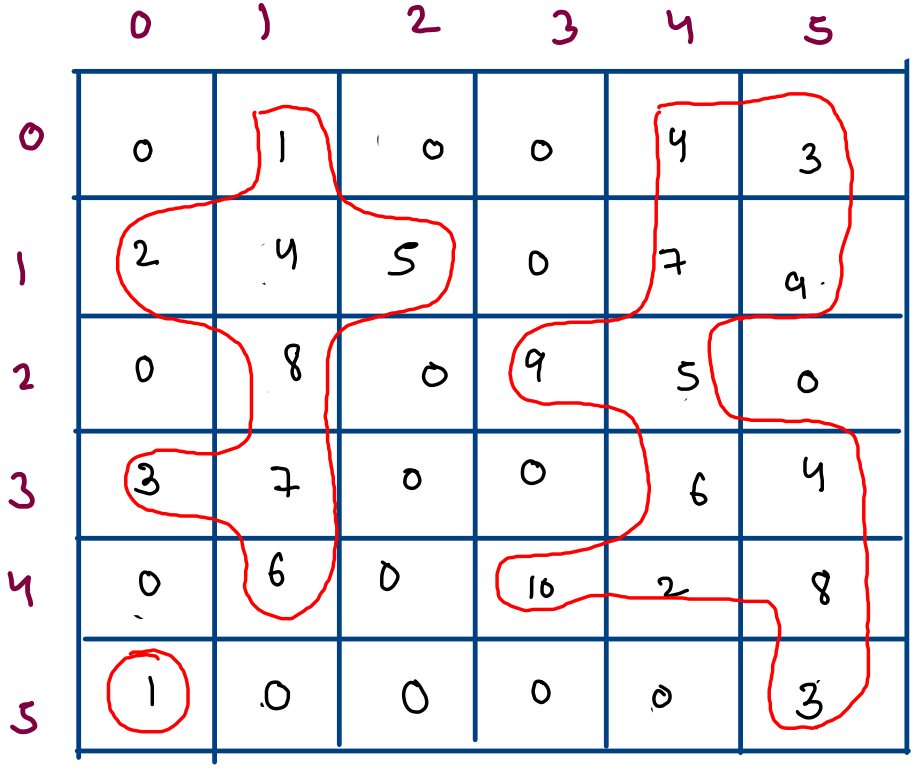
items:



LRLRTT
LRLRBB
TTTTLR
BBBBTT
LRLRBB

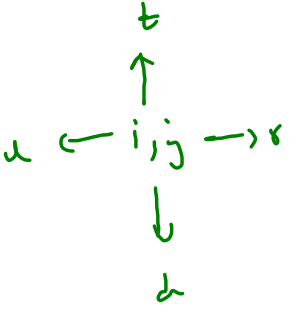


Gold Mine - 2



islands

0 -> block



```
public static void getMaxGold(int[][] arr){
    //write your code here

    for(int i=0; i < arr.length;i++) {
        for(int j=0; j < arr[0].length;j++) {
            if(arr[i][j] != 0) {
                sca = 0;
                dfs(arr,i,j);

                if(max < sca) {
                    max = sca;
                }
            }
        }
    }
}
```

```
public static void dfs(int[][]arr,int i,int j) {
    if(i < 0 || i >= arr.length || j < 0 || j >= arr[0].length || arr[i][j] == 0) {
        return;
    }

    sca += arr[i][j];

    arr[i][j] = 0;

    //top
    dfs(arr,i-1,j);

    //left
    dfs(arr,i,j-1);

    //down
    dfs(arr,i+1,j);

    //right
    dfs(arr,i,j+1);
}
```

max = ~~0~~ 34

sca = 3 + 8 + 4 + 7 + 2 + 5