1a)
2-dimensional array

1b)
Input: n: positive integer
Input: m: positive integer

1. Algorithm: MemArray
2. S = n by m array
3. initialize array to -1
4. return MysteryRecursion (n,m)

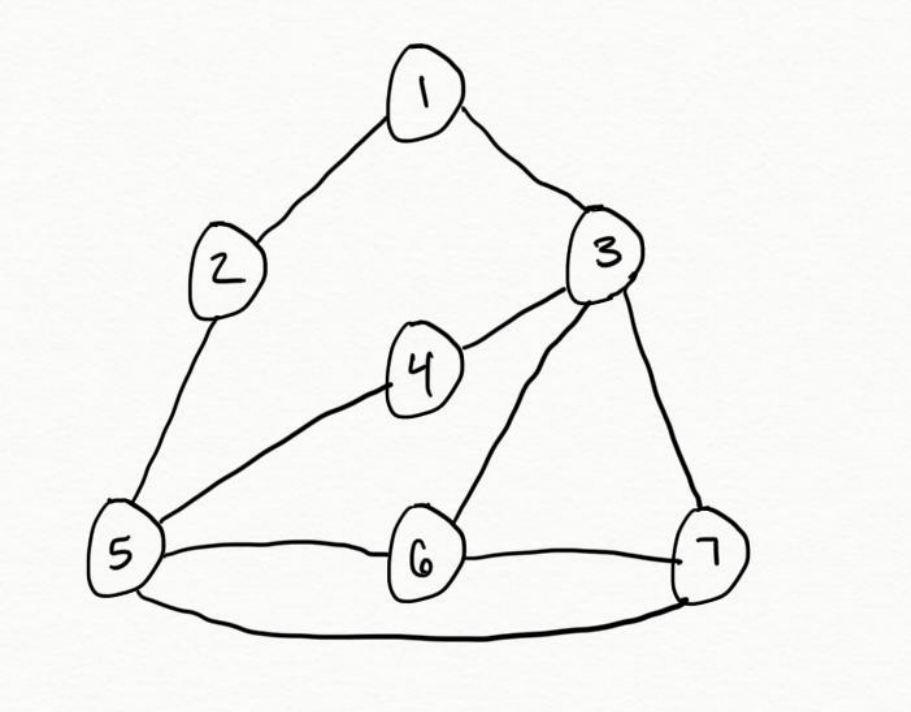
Input: n: positive integer
Input: m: positive integer

1. Algorithm: MysteryRecursion
2. if S[n,m]= -1
3. | if n and m = 1
4. | | S[n,m] =1
5. | else if n = 1 then
6. | | S[n,m] = m · MysteryRecursion(n, floor[m/2])
7. | else if m = 1 then
8. | | S[n,m] = n · MysteryRecursion(floor[n/2], m)
9. | else
10. | | S[n,m] = n·MysteryRecursion(floor[n/2],m) + m·MysteryRecursion(n, floor[m/2])
11. Return S[n,m]

1c)
A nested for loop would allow iterative dynamic programming to populate the 2d-array as such:

```
For i from 1 to n do
|       For j from 1 to m do
|       |       if i =1 and j =1
|       |       |       S[i,j] = 1
|       |       Else if i = 1
|       |       |       S[1,j] = j · S[ 1 , floor[j/2]]
|       |       Else if j = 1
|       |       |       S[i,1] = i · S[ floor[i/2] , 1]
|       |       Else
|       |       |       S[i,j] = i·S[ floor[i/2], j ] + j · S[ i , floor[j/2]]
```

2)



Vertex	Greedy	optimal
1	Red	Red
2	Blue	Blue
3	Blue	Green
4	Red	Red
5	Red	Green
6	Yellow	Blue
7	Green	Red

Since we found an example where greedy uses 4 colors, but we only need 3 colors, we have proven that the greedy algorithm is not optimal.