**Bottles problem**

**Part one**

**here are 2 bottles :** **A** - volume **m** liters

**B** - volume **n** liters

**A** **B**

volume **m** volume **n**

**m ≥ n**

**Start state : A i** liters **0 ≤ i ≤ m**

**B j** liters **0 ≤ j ≤ n**

**A** **B**

**i­**

**j**

**Start state : (i, j)**

**Final state : (?, ?)**

1. **(i, j) 🡪 (0, j)**

**i­**

**A A**

1. **(i, j) 🡪 (m, j)**

**m­**

**i­**

**A A**

1. **(i, j) 🡪 (i, 0)**

**B B**

**j**

1. **(i, j) 🡪 (i, n)**

**B B**

**n**

**j**

1. **(i, j) 🡪 (max(0, i + j - n), min(n, i + j))**

**A** **m** **B n**

**i­**

**j**

**5.1 i + j < n 🡪 A : 0 B : i + j**

**5.2 i + j > n 🡪 A : i + j – n B : n**

**max min**

1. **(i, j) 🡪 (min(m, i + j), max(0, i + j - m))**

**A** **m** **B** **n**

**i**

**j**

**6.1 i + j < m 🡪 A : i + j B : 0**

**6.2 i + j > m 🡪 A : m B : i + j - m**

**min max**

**Summary**

1. **(i, j) 🡪 (0, j) bottle A only**
2. **(i, j) 🡪 (m, j)**

**----------------------------------------------------------**

1. **(i, j) 🡪 (i, 0) bottle B only**
2. **(i, j) 🡪 (i, n)**

**----------------------------------------------------------**

1. **(i, j) 🡪 (max(0, i + j - n), min(n, i + j))**
2. **(i, j) 🡪 (min(m, i + j), max(0, i + j - m))**

**bottles A and B**

**Part Two**

1. **m = 2 n = 1**

**All modes : 0,0 0,1**

**1,0 1,1**

**2,0 2,1**

**--------------------------------------------------------------------------------------------------------------------------------------------------**

1. **m = 4 n = 2**

**All modes : 0,0 0,1 0,2**

**1,0 1,1 1,2**

**2,0 2,1 2,2**

**3,0 3,1 3,2**

**4,0 4,1 4,2**

1. **m = 2 n = 1**

**0,0 0,1**

**1,0 1,1**

**2,0 2,1**

**0,0 0,1 1,0 1,1 2,0 2,1 mode**

**0 1 2 3 4 5 index**

**m n (i,j) 🡪 index ??? formula**

**n = 1**

**0,0 0,1 1,0 1,1 2,0 2,1**

**1,0 1,1**

**2,0 2,1**

**0,0 0,1 1,0 1,1 2,0 2,1 mode**

**0 1 2 3 4 5 index**

1. **01 i = 0 \* x**

**0 1**

**10 11 i = 1 \* x**

**2 3**

**20 21 i = 2 \* x**

**4 5**

**x = 2**

**m = 2 n = 1**

**index = (n + 1) \* i + j**

1. **m = 4 n = 2**

**0,0 0,1 0,2 1,0 1,1 1,2 2,0 2,1 2,2 3,0 3,1 3,2 4,0 4,1 4,2**

**0 1 2 3 4 5 6 7 8 9 10 11 12 13 14**

m=4 n =2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | i j | 0 | 1 | 2 |
|  | 0 | 0,0 | 0,1 | 0,2 |
|  | 1 | 1,0 | 1,1 | 1,2 |
|  | 2 | 2,0 | 2,1 | 2,2 |
|  | 3 | 3,0 | 3,1 | 3,2 |
|  | 4 | 4,0 | 4,1 | 4,2 |

00 01 02 i = 0 \* x = 0

0 1 2

10 11 12 i = 1 \* x = 3

3 4 5

20 21 22 i = 2 \* x = 6

6 7 8

30 31 32 i = 3 \* x = 9

9 10 11

40 41 42 i = 4 \* x = 12

12 13 14

x = 3 = n + 1

**index = (n + 1) \* i + j**

**Part Three**

**m = 2**

**n = 1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **i,j** | **j** | 0,**0** | 0,**1** | 1,**0** | 1,**1** | 2,**0** | 2,**1** |
| **i** | **index** | **0** | **1** | **2** | **3** | **4** | **5** |
| **0**,0 | **0** | -- |  |  |  |  |  |
| **0**,1 | **1** |  | -- |  |  |  |  |
| **1**,0 | **2** |  |  | -- |  |  |  |
| **1**,1 | **3** |  |  |  | -- |  |  |
| **2**,0 | **4** |  |  |  |  | -- |  |
| **2**,1 | **5** |  |  |  |  |  | -- |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i,j | **j** | 0,**0** | 0,**1** | 1,**0** | 1,**1** | 2,**0** | 2,**1** |
| **i** | index | **0** | **1** | **2** | **3** | **4** | **5** |
| **0**,0 | **0** | -- | true |  |  | true |  |
| **0**,1 | **1** | true | -- | true |  |  | true |
| **1**,0 | **2** | true | true | -- | true | true |  |
| **1**,1 | **3** |  | true | true | -- | true | true |
| **2**,0 | **4** | true |  |  | true | -- | true |
| **2**,1 | **5** |  | true |  |  | true | -- |

m = 2

n = 1

//the index calculation for rows

**private** **static** **int** indexRow(**int** i, **int** j, **int** n){

**return** (n + 1) \* i + j;

}

**int** dim = (m+1)\*(n+1); // matrix dimension

**boolean** [][]mat = **new** **boolean**[dim][dim];

**for** (**int** i=0; i<=m; i++){

**for** (**int** j=0; j<=n; j++){

**…………………………………………………..**

**…………………………………………………..**

}

}

**…………………………………………………..**

**…………………………………………………..**

**int** index = *indexRow*(i,j,n);

mat[index][*indexRow*(0,j,n)] = **true**; //\*\*\*\*\* 1. (i, j) --> (0, j)

mat[index][*indexRow*(m,j,n)] = **true**; //\*\*\*\*\* 2. (i, j) --> (m, j)

mat[index][*indexRow*(i,0,n)] = **true**; //\*\*\*\*\* 3. (i, j) --> (i, 0)

mat[index][*indexRow*(i,n,n)] = **true**; //\*\*\*\*\* 4. (i, j) --> (i, n)

**int** indexTo = *indexRow*(Math.*max*(0,i+j-n),Math.*min*(n,i+j) ,n);

mat[index][indexTo] = **true**;

//\*\*\*\*\* 5. (i, j) --> (max(0, i + j - n), min(n, i + j))

indexTo = *indexRow*(Math.*min*(m,i+j),Math.*max*(0,j+i-m) ,n);

mat[index][indexTo] = **true**;

//\*\*\*\*\* 6. (i, j) --> (min(m, i + j), max(0, i + j - m))