Vim War

Ishaan Patel, Brogan Clements, Dominick DiMaggio

I pledge my honor that I have abided by the Stevens Honor System

Original Problem Statement

https://www.hackerrank.com/challenges/vim-war/problem

- Given: **N** soldiers and **M** skills
- Given: Target - a specific subset of **M** skills
- Each soldier can contain some subset of **M** skills
- Goal: Determine how many different combinations of soldiers that can meet the target skill set
- There can be no extra skills in the soldier set

Input/Output

Constraints:

1 <= **N** <= 100,000

1 <= **M** <= 20

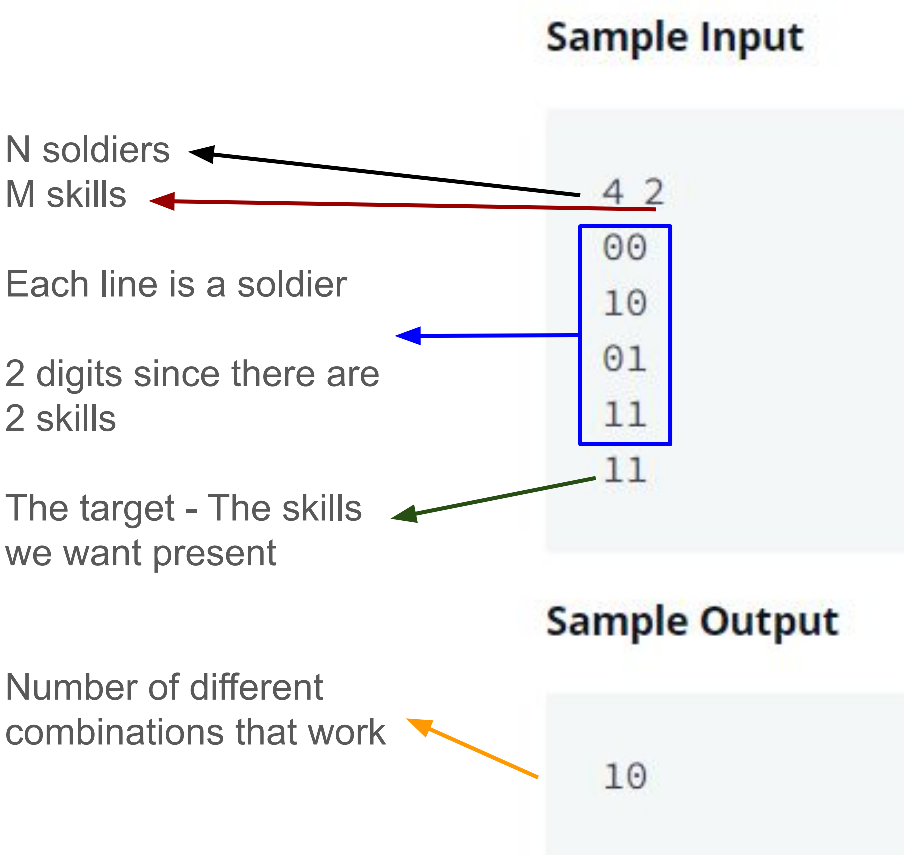
Input:

Each soldier is a binary value **M** digits long

0 = skill not present, 1 = skill is present

Output:

Integer of amount of possible combinations MODULO 10^9 + 7



N soldiers
M skills

Each line is a soldier

2 digits since there are 2 skills

The target - The skills we want present

Number of different combinations that work

Sample Input

```
4 2
00
10
01
11
11
```

Sample Output

```
10
```

Some test cases

```
3 2
11
00
00
11
```

Answer: 4

```
1 2
11
11
```

Answer: 1

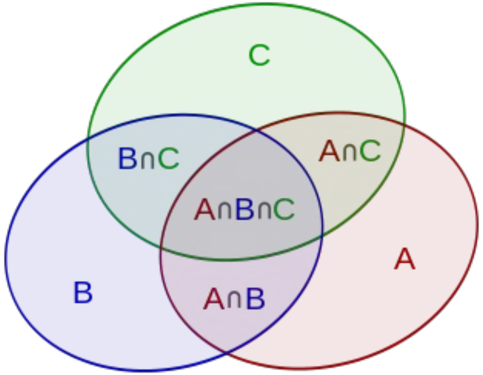
```
4 4
1010
0010
1100
1110
0001
```

Answer: 0

```
4 2
00
10
01
11
11
```

Answer: 10

Approaches/Realizations



Approaches Attempted:

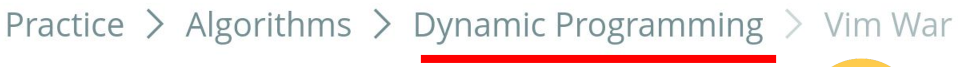
- Subset Sum
- Inclusion - Exclusion Principle (Final)

Our Realizations:

- Problem is NP complete
- It’s a math problem

Attempt: Adapting Subset Sum

- Our first attempt
- Problem conceptually maps well to the subset sum problem we’ve recently seen
- Worked well with the trivial branching recursive solution
  - SLOW
- Next optimization: **Memoization** (lazy improvement)
  - New problem: Maximum recursion depth (input sizes are extremely large)
- Final pivot: **Dynamic programming**



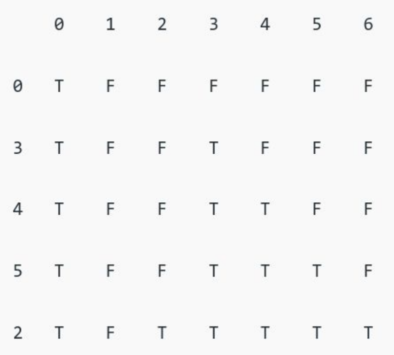
Practice > Algorithms > Dynamic Programming > Vim War

🤔

Subset Sum DP

Differences between DP for subset sum and Vim War:

- Table size
- Absence of starting values
- No analogous concept of a “previous state”



	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
4	T	F	F	T	T	F	F
5	T	F	F	T	T	T	F
2	T	F	T	T	T	T	T

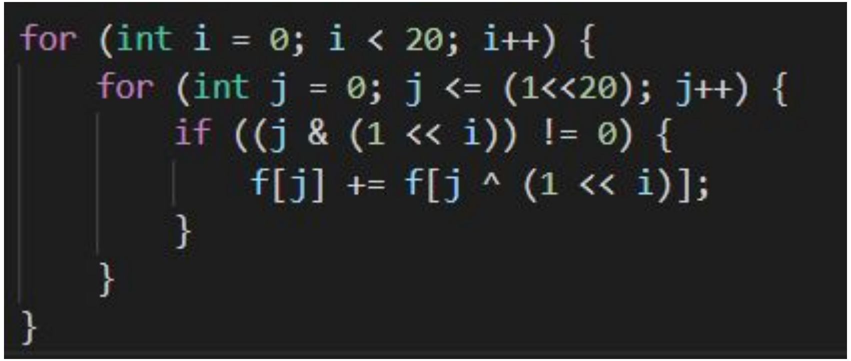
- Would need to be adapted further to account for more than a boolean answer

Attempt 2: Inclusion-Exclusion

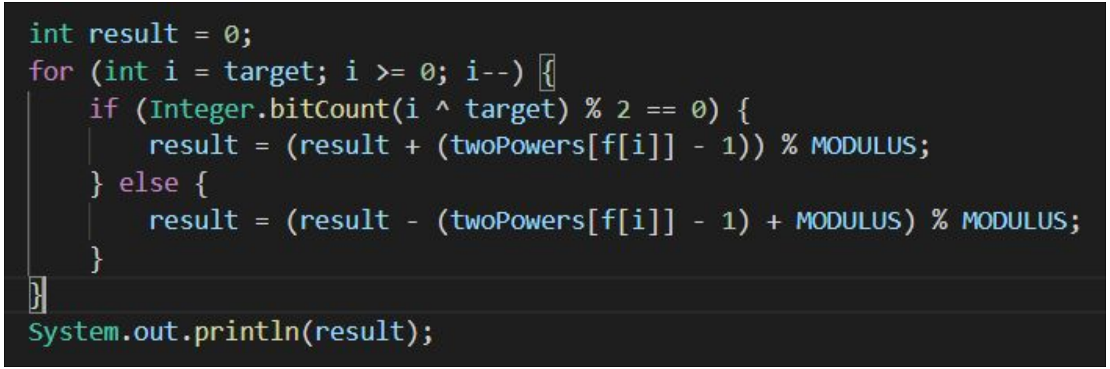
- Math of a much higher level than we ourselves completely understand

1. Calculating f(i)
    - f(i) = The number of numbers that equal i or are 1 bit away
    - What’s the purpose?
2. Calculating all subsets
    - 2^f(i) - 1 is all subsets that produce i
    - Why not just 2^f(target) - 1?
    - Add subsets 0 bits away, subtract subsets 1 bit away, add subsets 2 bits away, and so on

Mildly Interesting Code



```
for (int i = 0; i < 20; i++) {
    for (int j = 0; j <= (1<<20); j++) {
        if ((j & (1 << i)) != 0) {
            f[j] += f[j ^ (1 << i)];
        }
    }
}
```



```
int result = 0;
for (int i = target; i >= 0; i--) {
    if (Integer.bitCount(i ^ target) % 2 == 0) {
        result = (result + (twoPowers[f[i]] - 1)) % MODULUS;
    } else {
        result = (result - (twoPowers[f[i]] - 1) + MODULUS) % MODULUS;
    }
}
System.out.println(result);
```