The Lanternfish Problem Solved in $O(M^3 \log T)$ Time

Saito Kengo*

January 19, 2024

Problem Statement

You are given a sequence $A = (A_0, A_1, ..., A_{N-1})$ of length N.

The operation is defined as follows:

$A_i \leftarrow A_i - 1$

if $A_i < 0, A_i \leftarrow 6$, and add $A_{|A|+1} := 8$

You are given a positive integer T.

Output count of each integer from 0 to 8 after T operations.

Since the answer can be enormous, find it modulo 998244353.

Constraints

· $N, A_i, T \in \mathbb{N}$

· $0 \leqslant N \leqslant 100$

· $0 \leqslant A_i \leqslant 8 \; (0 \leqslant i < N)$

Subtask

1. $T = 18, 80$ (10pt)
2. $0 \leqslant T \leqslant 10^5$ (30pt)
3. $0 \leqslant T \leqslant 10^{18}$ (60pt)

Sample I/O

Input	Output
5 800	216570217
1 2 3 3 4	281664774
	587649421
	336942175
	61904212
	338823687
	388245435
	5657419
	839448772

Input	Output
7 1000000000000000000	837491177
1 2 3 4 5 6 7	132616921
	129466346
	442563120
	32132233
	42266735
	389177929
	290261401
	5867217

* National College of Technology, Ariake College

1 $O(T)$ *solution*

Define $V(=\mathbb{R}^9)$ s.t. $V_i :=$ count of occurrences of i in A and $Ans_T(=\mathbb{R}^9) :=$ the answer for the T-th day.

The transitions of Ans are represented as follows:

$$
\begin{aligned}
Ans_k \quad \leftarrow \quad & [Ans_{k-1}[1], Ans_{k-1}[2], Ans_{k-1}[3], Ans_{k-1}[4], Ans_{k-1}[5] \\
& , Ans_{k-1}[6], Ans_{k-1}[7] + Ans_{k-1}[0], Ans_{k-1}[8], Ans_{k-1}[0]]
\end{aligned} \tag{1}
$$

Taking note that $Ans_0 = V$, it can be observed that Ans_T is obtained by performing T operations mentioned above.

The time complexity of this solution is $O(9T) \simeq O(T)$.

Python

```python
N, T = map(int,input().split())
A = list(map(int,input().split()))

MOD = 998244353

Ans = [0] * 9
for i in range(N):
    Ans[A[i]] += 1

for _ in range(T):
    Ans_before = Ans[:]
    Ans[0] = Ans_before[1] % MOD
    Ans[1] = Ans_before[2] % MOD
    Ans[2] = Ans_before[3] % MOD
    Ans[3] = Ans_before[4] % MOD
    Ans[4] = Ans_before[5] % MOD
    Ans[5] = Ans_before[6] % MOD
    Ans[6] =(Ans_before[7] + Ans_before[0]) % MOD
    Ans[7] = Ans_before[8] % MOD
    Ans[8] = Ans_before[0] % MOD

for i in range(9):
    print(Ans[i])
```

2 $O(M^3 \log T)$ *solution*

Define $V(=\mathbb{R}^{9\times 1})$ s.t. $V_i :=$ count of occurrences of i in A and $Ans_T(=\mathbb{R}^{9\times 1}) :=$ the answer for the T-th day.

Find a matrix $DP(=\mathbb{R}^{9\times 9})$ that satisfies the following:

$$Ans_T = DP \cdot Ans_{T-1} \tag{2}$$

$$\begin{bmatrix} Ans_{T_0} \\ Ans_{T_1} \\ Ans_{T_2} \\ Ans_{T_3} \\ Ans_{T_4} \\ Ans_{T_5} \\ Ans_{T_6} \\ Ans_{T_7} \\ Ans_{T_8} \end{bmatrix} = \begin{bmatrix} DP_{0,0} & DP_{0,1} & \cdots & DP_{0,7} & DP_{0,8} \\ DP_{1,0} & & & & DP_{1,8} \\ & & & & \\ & & & & \\ \vdots & & \ddots & & \vdots \\ & & & & \\ & & & & \\ DP_{7,0} & & & & DP_{7,8} \\ DP_{8,0} & DP_{8,1} & \cdots & DP_{8,7} & DP_{8,8} \end{bmatrix} \begin{bmatrix} Ans_{T_8} \\ Ans_{T_0} \\ Ans_{T_1} \\ Ans_{T_2} \\ Ans_{T_3} \\ Ans_{T_4} \\ Ans_{T_5} \\ Ans_{T_6} - Ans_{T_8} \\ Ans_{T_7} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} Ans_{T_0} \\ Ans_{T_1} \\ Ans_{T_2} \\ Ans_{T_3} \\ Ans_{T_4} \\ Ans_{T_5} \\ Ans_{T_6} \\ Ans_{T_7} \\ Ans_{T_8} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Ans_{T_8} \\ Ans_{T_0} \\ Ans_{T_1} \\ Ans_{T_2} \\ Ans_{T_3} \\ Ans_{T_4} \\ Ans_{T_5} \\ Ans_{T_6} - Ans_{T_8} \\ Ans_{T_7} \end{bmatrix} \tag{4}$$

Following the equation 2, we can get k-th terms of the sequence Ans.

$$\begin{aligned} Ans_k &= DP \cdot Ans_{k-1} && (5)\\ &= DP \cdot DP \cdot Ans_{k-2} && (6)\\ \cdots & && (7)\\ &= DP^k \cdot Ans_0 && (8)\\ &= DP^k \cdot V && (9) \end{aligned}$$

So, Ans_T is

$$Ans_T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \end{bmatrix} \tag{10}$$

The N-th power of a real number can be computed using repeated squaring (like doubling technique) in $O(\log T)$ time. Similarly, repeated squaring can be applied to matrices. With the size of matrix DP being M $(= 9)$, the bottleneck arises from matrix multiplication, resulting in a time complexity of $O(M^3 \log T)$. In this problem, M is fixed at 9, making it sufficiently fast.
By addressing this, you can achieve 60 pt out of 100 pt for subtask 3.

C++

```cpp
#include <iostream>
#include <vector>
#include <cmath>
#define int long long
using namespace std;

const int MOD = 998244353;
const int M = 9;

vector<vector<int>> DP = {
    {0, 1, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 1, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 1, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0, 0},
    {1, 0, 0, 0, 0, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 1},
    {1, 0, 0, 0, 0, 0, 0, 0, 0}
};

vector<vector<int>> matrixMult(vector<vector<int>>& A, vector<vector<int>>& B) {
    int Bx = B.size();
    vector<vector<int>> result(M, vector<int>(Bx, 0));
    for (int i = 0; i < M; ++i) { // A_row
        for (int j = 0; j < Bx; ++j) { // B_col
            for (int k = 0; k < M; ++k) { // A_col = B_row
```

```
                result[i][j] += (A[i][k] % MOD * B[k][j] % MOD) % MOD;
                                result[i][j] %= MOD;
            }
        }
    }

    return result;
}

signed main() {
    int N, T;
    cin >> N >> T;

        // Ans_0 (= V)
    vector<vector<int>> Ans(M, vector<int>(1,0));
    for (int i = 0; i < N; ++i) {
        int A; cin >> A;
        Ans[A][0] += 1;
    }

    while (T) {
        if (T & 1) Ans = matrixMult(DP, Ans);
        DP = matrixMult(DP, DP);
        T >>= 1;
    }

    for (int i = 0; i < 9; ++i) {
        cout << Ans[i][0] << endl;
    }

    return 0;
}
```

C++ with AtCoder Library; ACL / modint (recommend)

```cpp
#include <iostream>
#include <vector>
#include <cmath>
// Ref : https://github.com/atcoder/ac-library
#include <atcoder/modint>
#define int long long

using namespace std;
using namespace atcoder;
using mint = modint998244353;

const int M = 9;

vector<vector<mint>> DP = {
    {0, 1, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 1, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 1, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0, 0},
    {1, 0, 0, 0, 0, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 1},
    {1, 0, 0, 0, 0, 0, 0, 0, 0}
};

vector<vector<mint>> matrixMult(vector<vector<mint>>& A, vector<vector<mint>>& B)
     {
    int Bx = B.size();

    vector<vector<mint>> result(M, vector<mint>(Bx, 0));

    for (int i = 0; i < M; ++i) { // A_row
        for (int j = 0; j < Bx; ++j) { // B_col
            for (int k = 0; k < M; ++k) { // A_col = B_row
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    return result;
}

signed main() {
    int N, T;
    cin >> N >> T;

```

```
        // Ans_0 (= V)
    vector<vector<mint>> Ans(M, vector<mint>(1,0));
    for (int i = 0; i < N; ++i) {
        int A; cin >> A;
        Ans[A][0] += 1;
    }

    while (T) {
        if (T & 1) Ans = matrixMult(DP, Ans);
        DP = matrixMult(DP, DP);
        T >>= 1;
    }

    for (int i = 0; i < 9; ++i) {
        cout << Ans[i][0].val() << endl;
    }

    return 0;
}
```