SOL POUR

Đề bài yêu cầu rót 1 số lít nước từ bình này qua bình khác để ít nhất 1 bình có lượng nước bằng m. và số lần rót là ít nhất.

Không tính trạng thái ban đầu. ta có sau mỗi lần rót. Sẽ phải tồn tại 1 bình có lượng nước bằng $p[i]$ $(i = 1 \ldots . . n)$ hoặc bằng 0 (theo đề bài).

Vậy ta có bao nhiêu trạng thái?

Với mỗi trạng thái khác trạng thái ban đầu. sẽ tồn tại 1 bình có lượng nước bằng $p[i]$ $(i = 1 \ldots . . n)$ hoặc bằng 0 (theo đề bài) và 2 bình còn lại có tổng lượng nước là $v - p[i]$. vậy nên với mỗi trạng thái ta sẽ có dạng $(p[i], x, k)$ với $p[i] + x + k = v$. Ta có trạng thái $(p[i], x, k)$ với $x + k$ dựa vào $p[i]$ nên ta chỉ cần lưu $(p[i], min(x, k))$ có nghĩa là trạng thái $(p[i], x, k) = (p[i], min(x, k))$. Với trạng thái $(p[i], min(x, k))$ tiếp tục tối ưu ta lưu trạng thái là $(i, min(x, k))$. Và trường hợp bằng 0 thì ta lưu là $(0, min(x, k))$.

Vậy ta có $(n + 1) * v/2$ trạng thái. Nên ta có thể làm theo cách bfs duyệt hết trạng thái và tìm số lần rót ít nhất của mỗi trạng thái.

Đáp an là số lần rót ít nhất của trạng thái mà tồn tại ít nhất một bình đang có lưu lượng $= m$.

Dpt$((n + 1) * v/2)$.

Code tham khảo

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define fi first
#define se second
typedef pair <int, int> ii;

int v, n, m, p[25];
int f[25] [50099], ans;
queue <ii> q;

// Donuoc(int a, int b, int val): Đổ nước từ cốc có mực nước a sang cốc có mực nước b
HOẶC đổ nước từ cốc có mực nước b sang cốc có mực nước a sao cho cốc có mực
nước a trở về vạch p[j], lượng nước đổ đi là a-p[j], bình b sẽ có lượng nước là b+a-
p[j].

Số phép biến đổi của trạng thái hiện tại là Val
```

Lúc này mực nước 3 bình là (p[j], b+a-p[j], V-a-b)

```
void Donuoc(int a, int b, int val)
{
 //Xét hai tình huống đổ từ a sang b lượng nước a-p[j] để bình a có vạch p[j] khi đó điều kiện đổ là if(a >= p[j])
// Hoặc đổ từ b sang a lượng nước p[j] – a để bình a có vạch p[j] khi đó điều kiện đổ là if(a < p[j] && b > p[j] - a)
//Gộp 2 tình huống lại vì đều đưa về trạng thái có 1 bình là p[j] và bình nhỏ nhất là min(b + a - p[i], v - a - b)

 for (int j = 0; j <= n; j++)
   {
      if (p[j] > a + b) break;
      if (f[j][min(a + b - p[j], v - a - b)] == -1)
      {
         f[j][min(a + b - p[j], v - a - b)] = val + 1;
         q.push(j,{min(a + b - p[j], v - a - b)});
      }
   }


}
//f[i][x]: Số lần đổ nước ít nhất đến trạng thái cả 3 bình có 1 bình mực nước p[i], 2 bình còn lại có lượng nước x
void solve()
{
   memset(f, -1, sizeof f); //Xác đinh trạng thái chưa đến
   f[0][0] = 0;
   q.push({0, 0});

   while(q.size())
   {
      auto [id,x] = q.front();
      q.pop();

      int y = v - x - p[id]; //Tính lượng nước bình thứ 3

      if (x == m || y == m || p[id] == m) //Tính lượng nước bình thứ 3
      {
         ans = f[id][x];
         return;
      }
```

```
        Donuoc(x, y, f[id][x]);
        Donuoc(x, p[id], f[id][x]);
        Donuoc(y, x, f[id][x]);
        Donuoc(y, p[id], f[id][x]);
        Donuoc(p[id], x, f[id][x]);
        Donuoc(p[id], y, f[id][x]);
    }
}

int main()
{
    ios_base::sync_with_stdio(NULL);
    cin.tie(NULL);
    cout.tie(NULL);

    freopen("POUR.inp", "r", stdin);
    freopen("POUR.out", "w", stdout);

    cin >> v >> n >> m;
    for (int i = 1; i <= n; i++)
    {
        cin >> p[i];
    }

    ans = -1;
    solve();
    cout << ans;

}
```