***Machine and Workers***

C371\_Coding\_March2023

**Topic**: Searching Algorithms

**Difficulty Level:** Hard

**Question / Problem Statement**:

Eleanor loves playing games. Her current favourite game is CandyMaker, where the goal is to make candies.

Eleanor just started a level in which she must accumulate **N** candies starting with **M** machines and **W** workers. In a single pass, she can make **M**\***W** candies. After each pass, she can decide whether to spend some of her candies to buy more machines or hire more workers. Buying a machine or hiring a worker costs **P** units, and there is no limit to the number of machines she can own or workers she can employ.

Eleanor wants to minimise the number of passes to obtain the required number of candies at the end of a day. Determine that number of passes.

For example, Karl starts with **M** = 1 machine and **W** = 2 workers. The cost to purchase or hire, **P =** 1and she needs to accumulate 60 candies. Shе executes the following strategy:

1. Make **M** x **W** = 1 x 2 = 2 candies. Purchase two machines.

2. Make 3 x 2 = 6 candies. Purchase 3 machines and hire 3 workers.

3. Make 6 x 5 = 30 candies. Retain all 30 candies.

4. Make 6 x 5 = 30 candies. With yesterday's production, Eleanor has 60 candies.

It took 4 passes to make enough candies.

Write a program to determine the minimum number of passes required to accumulate at least **N** candies.

**Note**

In a single pass, she can make **M**\***W** candies.

**Function Description**

In the provided code snippet, implement the provided **machineAndWorkers(...)** method using the variables to determine the minimum number of passes required to accumulate at least **N** candies. You can write your code in the space below the phrase **“WRITE YOUR LOGIC HERE”**.

There will be multiple test cases running so the Input and Output should match exactly as provided.  
The base Output variable **result** is set to a default value of **-404** which can be modified. Additionally, you can add or remove these output variables.

**Input Format**

A single line consisting of four space-separated integers describing the values of **M**, **W**, **P** and **N**, the starting number of machines and workers, the cost of a new machine or a new

hire, and the number of candies Eleanor must accumulate to complete the level.

**Sample Input**

3 1 2 12 – denotes M, W, P and N.

**Constraints**

1 <= **M, W, P, N** <= 10^12.

**Output Format**

Output should return a long integer denoting the minimum number of passes required to accumulate at least **N** candies.

**Sample Output**

3

**Explanation**

Eleanor makes three passes:

1. In the first pass, she makes M\*W = 3\*1=3 candies. She then spends P=2 of them hiring another worker, so W=2 and she has one candy left over.

2. In the second pass, she makes 3\*2=6 candies. She spends 2\*p=4 of them on another machine and another worker, so W=3 and M=4 and she has 3 candies left over.

3. In the third pass, Eleanor makes 4\*3=12 candies. Because this satisfies his goal of making at least N=12 candies, we print the number of passes (i.e., 3) as our answer.

**Solution Steps**

1. This problem can be solved by using a binary search. Basically we binary search on the answer (number of passes) and check if the current round number is enough to make the required number of candies. If we can do so, we try to minimise our answer by decreasing the number of passes, else we increase the number of passes.

2. Two important observations to make when solving this problem are:

1. Always buy machines and hire workers as early as possible.  
   If we do so, then the subsequent operations will lead to a larger product everytime and hence more candies.
2. To maximise the number of candies made during each round, the numbers of workers and machines should be either equal or as close to each other as possible. Hence, we must always invest in whichever resource we have less of.

This is because let's say the current number of workers and machines are x and y respectively.

And we decide to spend a z number of candies for the current operation to increase x and y in

some ratio. Obviously, our aim is to maximise the product x' \* y’ where x' and y’ are the new

x and y after the increment respectively. We know that their sum is x + y + z. Let's denote this

sum by S. So we have, x' + y’ = S and we need to maximise x' \* y' which is maximum when x’=y’=S/2. Hence x’ and y’ should be as close as possible.

3. We write a check function having the following parameters: M, W, ,P, N and mid. This function returns a boolean value denoting whether or not it's possible to produce the required number of candies, N, in less than or equal to mid number of passes if we currently have M machines and W workers where each new machine or worker costs P candies.

4. If we can make candies with the current resources in the remaining number of passes, then the function returns true; otherwise, we must try to make candies and buy new resources (i.e., if there are more machines than workers, hire more workers or if both are equal increase both of them in equal proportion). Compare the remaining number of passes with the number of passes needed to make candies for buying additional resources and, if the remaining number is less, return false.

**Running Solution in C++** :

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

bool check(ll machines, ll workers, ll price, ll target, ll rounds) {

if (machines >= (target+workers-1)/workers) return true;

ll cur = machines\*workers;

rounds--;

if (rounds == 0) return false;

while (1) {

ll rem = target - cur;

ll rnds = (rem + machines\*workers - 1) / (machines\*workers);

if (rnds <= rounds) return true;

if (cur < price) {

rem = price - cur;

rnds = (rem + machines\*workers - 1) / (machines\*workers);

rounds -= rnds;

if (rounds < 1) return false;

cur += rnds \* machines \* workers;

}

cur -= price;

if (machines > workers) {

workers++;

} else {

machines++;

}

}

return false;

}

int main(){

ios::sync\_with\_stdio(0);

cin.tie(0);

ll m, w, p, n;

cin >> m >> w >> p >> n;

ll a = 1, b = 1000000000000LL;

while (a < b) {

ll mid = (a + b) >> 1;

if (check(m, w, p, n, mid)) {

b = mid;

} else {

a = mid + 1;

}

}

cout << a << "\n";

return 0;

}

Input:

1 1 6 45

Output:

16

**Test Cases [ Qty: 12 ]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case No** | **Input** | **Output** | **Score** |
| 1 | 3 1 2 12 | 3 | 0 |
| 2 | 1 1 6 45 | 16 | 0 |
| 3 | 5184889632 5184889632 20 10000 | 1 | 1 |
| 4 | 1 1 1000000000000 1000000000000 | 1000000000000 | 1 |
| 5 | 31745 17538 37467 3109235745 | 4 | 1 |
| 6 | 1 1 1 1 | 1 | 1 |
| 7 | 1 100 10000000000 1000000000000 | 617737754 | 1 |
| 8 | 5 2 10302 9133131738 | 6583 | 1 |
| 9 | 2 2 39708 5068524053 | 45473 | 1 |
| 10 | 13 2 11216 9767611550 | 3584 | 1 |
| 11 | 4408 8432 19546 2872106926 | 9 | 1 |
| 12 | 38 25 22 21 | 1 | 1 |

Plagiarism found – No

Clarity of the problem statement - Yes

Clarity of the example in the problem statement - Yes

Clarity of sample test cases - Yes

Clarity of test cases (Dual output) – Yes

Clarity of explanations - Yes

Provided Solution running – Yes

EEOC complaint (using abusive words/Indian Names/) - No

Similar Question in System - No

Difficulty Level – Hard

Question w.r.t Searching algorithms concepts- Yes

Final Comment: **Accepted**