5986. Minimum Cost to Set Cooking Time

My Submissions (/contest/biweekly-contest-71/problems/minimum-cost-to-set-cooking-time/submissions/)

Back to Contest (/contest/biweekly-contest-71/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

A generic microwave supports cooking times for:

- at least `1` second.
- at most `99` minutes and `99` seconds.

To set the cooking time, you push **at most four digits**. The microwave normalizes what you push as four digits by **prepending zeroes**. It interprets the **first** two digits as the minutes and the **last** two digits as the seconds. It then **adds** them up as the cooking time. For example,

- You push `9 5 4` (three digits). It is normalized as `0954` and interpreted as `9` minutes and `54` seconds.
- You push `0 0 0 8` (four digits). It is interpreted as `0` minutes and `8` seconds.
- You push `8 0 9 0`. It is interpreted as `80` minutes and `90` seconds.
- You push `8 1 3 0`. It is interpreted as `81` minutes and `30` seconds.

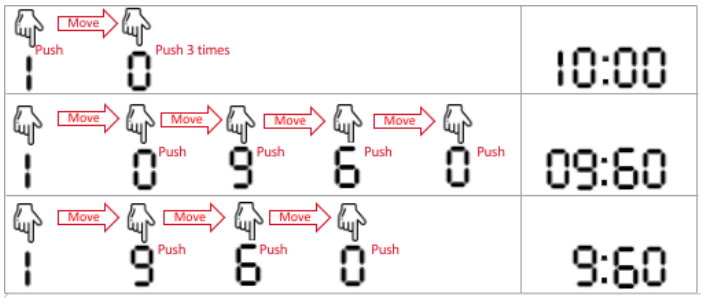
You are given integers `startAt`, `moveCost`, `pushCost`, and `targetSeconds`. **Initially**, your finger is on the digit `startAt`. Moving the finger above **any specific digit** costs `moveCost` units of fatigue. Pushing the digit below the finger **once** costs `pushCost` units of fatigue.

There can be multiple ways to set the microwave to cook for `targetSeconds` seconds but you are interested in the way with the minimum cost.

Return *the **minimum cost** to set* `targetSeconds` *seconds of cooking time*.

Remember that one minute consists of `60` seconds.

Example 1:




```
Input: startAt = 1, moveCost = 2, pushCost = 1, targetSeconds = 600
Output: 6
Explanation: The following are the possible ways to set the cooking time.
- 1 0 0 0, interpreted as 10 minutes and 0 seconds.
  The finger is already on digit 1, pushes 1 (with cost 1), moves to 0 (with cost 2), pushes 0 (with cost 1), pushes 0 (with
  The cost is: 1 + 2 + 1 + 1 + 1 = 6. This is the minimum cost.
- 0 9 6 0, interpreted as 9 minutes and 60 seconds. That is also 600 seconds.
  The finger moves to 0 (with cost 2), pushes 0 (with cost 1), moves to 9 (with cost 2), pushes 9 (with cost 1), moves to 6
  The cost is: 2 + 1 + 2 + 1 + 2 + 1 + 2 + 1 = 12.
- 9 6 0, normalized as 0960 and interpreted as 9 minutes and 60 seconds.
  The finger moves to 9 (with cost 2), pushes 9 (with cost 1), moves to 6 (with cost 2), pushes 6 (with cost 1), moves to 0
  The cost is: 2 + 1 + 2 + 1 + 2 + 1 = 9.
```

Example 2:

```
Input: startAt = 0, moveCost = 1, pushCost = 2, targetSeconds = 76
Output: 6
Explanation: The optimal way is to push two digits: 7 6, interpreted as 76 seconds.
The finger moves to 7 (with cost 1), pushes 7 (with cost 2), moves to 6 (with cost 1), and pushes 6 (with cost 2). The total
Note other possible ways are 0076, 076, 0116, and 116, but none of them produces the minimum cost.
```

Constraints:

- 0 <= startAt <= 9
- 1 <= moveCost, pushCost <= 10^5
- 1 <= targetSeconds <= 6039

JavaScript

```javascript
let start, mc, pc, res, sec;
const minCostSetTime = (startAt, moveCost, pushCost, targetSeconds) => {
    start = startAt, mc = moveCost, pc = pushCost, res = Number.MAX_SAFE_INTEGER, sec = targetSeconds;
    // pr(cal("1000"), cal("0960"), cal("960"))
    // pr(cal("0076"), cal("076"))
    dfs(0, '');
    return res;
};

const dfs = (idx, cur) => {
    if (cur.length > 4) return;
    // pr(idx, "cur", cur);
    if (cur.length == 1) {
        if (cur - '0' == sec) {
            // pr("length 1", cur, cal(cur))
            res = Math.min(res, cal(cur))
        }
    }
    if (cur.length == 2) {
        if (cur - '0' == sec) {
            // pr("length 2", cur, cal(cur))
            res = Math.min(res, cal(cur))
        }
    }
    if (cur.length == 4 || cur.length == 3) {
        let h, m;
        if (cur.length == 3) {
            h = cur.slice(0, 1) - '0';
            m = cur.slice(1) - '0';
        } else if (cur.length == 4) {
            h = cur.slice(0, 2) - '0';
            m = cur.slice(2) - '0';
        }
        if (h * 60 + m == sec) {
            // pr("length 3|4 ", cur, cal(cur), h * 60 + m, sec)
            res = Math.min(res, cal(cur))
        }
    }
    for (let i = 0; i < 10; i++) {
        cur += i + '';
        dfs(i, cur);
        cur = cur.slice(0, -1);
    }
};

const cal = (s) => {
    let sum = 0, pre = start;
    for (const c of s) {
        if (c - '0' != pre) {
            sum += mc;
        }
        sum += pc;
        pre = c - '0';
    }
    return sum;
```

```
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/635096388/) | More Details > (/submissions/detail/635096388/)

Share your acceptance!

1