HKOI Junior Q3 (Shortest Path) Editorial

Alex Tung
alex20030190@yahoo.com.hk

27 January 2018

Table of Contents

Task Description

- Given a $N \times M$ grid. Also given a parameter K.

Task Description

- Given a $N \times M$ grid. Also given a parameter K.
- Alice's piece starts at cell (S_r, S_c) and she needs to bring the piece to cell (E_r, E_c).

Task Description

- Given a $N \times M$ grid. Also given a parameter K.
- Alice's piece starts at cell (S_r, S_c) and she needs to bring the piece to cell (E_r, E_c).
- For each move, she can move her piece X steps up, down, left, or right. X should equal 1 or a positive integer multiple of K.

Task Description

- Given a $N \times M$ grid. Also given a parameter K.
- Alice's piece starts at cell (S_r, S_c) and she needs to bring the piece to cell (E_r, E_c).
- For each move, she can move her piece X steps up, down, left, or right. X should equal 1 or a positive integer multiple of K.
- Of course, after each step, Alice's piece should remain on the board.

Task Description

- Given a $N \times M$ grid. Also given a parameter K.
- Alice's piece starts at cell (S_r, S_c) and she needs to bring the piece to cell (E_r, E_c).
- For each move, she can move her piece X steps up, down, left, or right. X should equal 1 or a positive integer multiple of K.
- Of course, after each step, Alice's piece should remain on the board.
- Output a shortest sequence of moves, which brings her piece from (S_r, S_c) to (E_r, E_c).

Sample IO

Sample Input 1

```
2 9 3
1 2
2 8
```

Sample Output 1

```
2
right 6
down 1
```

Sample Input 2

```
1 8 4
1 1
1 8
```

Sample Output 2

```
3
right 4
left 1
right 4
```

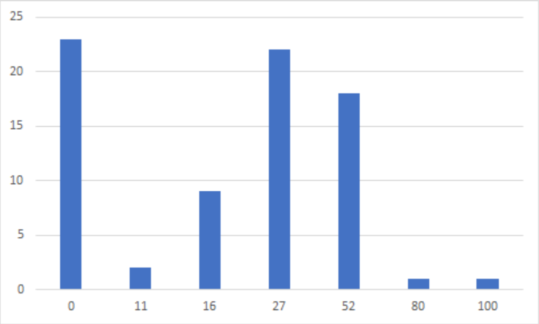
Constraints

For all cases:
$1 \leq N, M \leq 10^9$
$1 \leq K \leq 1000$

	Points	Constraints
1	16	(E_r, E_c) is reachable in one move
2	11	$K = 1$
3	25	$K = 2$
4	28	$(S_r, S_c) = (1, 1)$
5	20	No additional constraints

Statistics

Attempts: 76
Mean: 24.684
Stddev: 22.315
Top scores: 100 (ethening, 1:11), 80 (mtyeung1), 52 (18 contestants)
Score distribution:



Comments

- This is the hardest problem in HKOI 2017/18 Junior. (Far easier than J174 though ;))

Comments

- This is the hardest problem in HKOI 2017/18 Junior. (Far easier than J174 though ;))
- Ad-hoc problem

Comments

- This is the hardest problem in HKOI 2017/18 Junior. (Far easier than J174 though ;))
- Ad-hoc problem
- Getting $16 + 11 = 27$ points is easy, but full solution requires careful case handling.

Comments

- This is the hardest problem in HKOI 2017/18 Junior. (Far easier than J174 though ;))
- Ad-hoc problem
- Getting $16 + 11 = 27$ points is easy, but full solution requires careful case handling.
- It is not easy to code the solution (more on that later).

Subtask 1

Subtask 1 (16 points): (E_r, E_c) is reachable in one move.

Subtask 1

Subtask 1 (16 points): (E_r, E_c) is reachable in one move.

- That means, in particular, that (S_r, S_c) and (E_r, E_c) are on the same row/column.

Subtask 1

Subtask 1 (16 points): (E_r, E_c) is reachable in one move.

- That means, in particular, that (S_r, S_c) and (E_r, E_c) are on the same row/column.
- $S_r > E_r$, then move "up"
- $S_r < E_r$, then move "down"
- $S_c > E_c$, then move "left"
- $S_c < E_c$, then move "right"

Subtask 1

Subtask 1 (16 points): (E_r, E_c) is reachable in one move.

- That means, in particular, that (S_r, S_c) and (E_r, E_c) are on the same row/column.
- $S_r > E_r$, then move "up"
- $S_r < E_r$, then move "down"
- $S_c > E_c$, then move "left"
- $S_c < E_c$, then move "right"
- What should X be? X just equals the distance between the two cells.

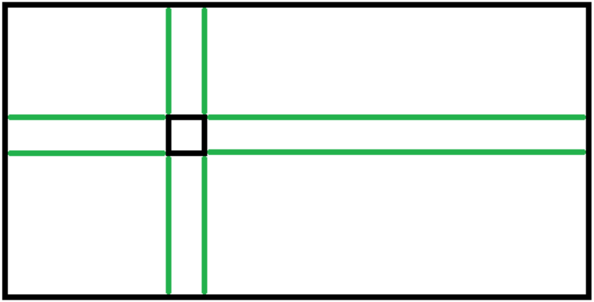
Subtask 2

Subtask 2 (11 points): $K = 1$

Subtask 2

Subtask 2 (11 points): $K = 1$

- You can divide the board into nine parts, using (S_r, S_c) as center.
- Then, write a bunch of 'if's :(



Subtask 2

Subtask 2 (11 points): $K = 1$

- You can divide the board into nine parts, using (S_r, S_c) as center.
- Then, write a bunch of 'if's :(
- To make life easier, observe that:

Observation 1

Horizontal (left/right) moves and vertical (up/down) moves are independent.

Subtask 3

Subtask 3 (25 points): $K = 2$

Subtask 3

Subtask 3 (25 points): $K = 2$

- In this subtask, there are finally some (non-obvious) decision-making.

Subtask 3

Subtask 3 (25 points): $K = 2$

- In this subtask, there are finally some (non-obvious) decision-making.
- By Observation 1, assume that $N = 1$ (so we are only concerned with horizontal moves).

Subtask 3

Subtask 3 (25 points): $K = 2$

- In this subtask, there are finally some (non-obvious) decision-making.
- By Observation 1, assume that $N = 1$ (so we are only concerned with horizontal moves).
- Further assume that $S_c < E_c$ (so we need to bring the piece to the right).

Subtask 3

Claim 1

Let $D := E_c - S_c$. Let $C := \lfloor \frac{D}{2} \rfloor$.

- If $D = 1$, the optimal solution is `right 1`.
- If $D > 1$ and is odd, an optimal solution is `right` $2C$; `right 1`.
- If D is even, the optimal solution is `right` $2C$.

Proof

It is obvious that we cannot do better.

Subtask 4

Subtask 4 (28 points): $(S_r, S_c) = (1, 1)$

Subtask 4

Subtask 4 (28 points): $(S_r, S_c) = (1, 1)$

- This is just a "safety net" for those who attempts the full solution :)

Subtask 4

Subtask 4 (28 points): $(S_r, S_c) = (1, 1)$

- This is just a "safety net" for those who attempts the full solution :)
- If you miss one case, you'll pass this subtask but not the next one.

Subtask 5

Subtask 5 (20 points): No additional constraints

Subtask 5

Subtask 5 (20 points): No additional constraints

- Same as for Subtask 3, we assume first that $N = 1$ and $S_c < E_c$.

Subtask 5

Subtask 5 (20 points): No additional constraints

- Same as for Subtask 3, we assume first that $N = 1$ and $S_c < E_c$.
- Again, let $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.

Subtask 5

Subtask 5 (20 points): No additional constraints

- Same as for Subtask 3, we assume first that $N = 1$ and $S_c < E_c$.
- Again, let $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.
- For each move `dir X`, if $X = 1$ we say it is a **small step**; otherwise we say it consists of $\frac{X}{K}$ **big step(s)**.

Subtask 5

Subtask 5 (20 points): No additional constraints

- Same as for Subtask 3, we assume first that $N = 1$ and $S_c < E_c$.
- Again, let $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.
- For each move `dir X`, if $X = 1$ we say it is a **small step**; otherwise we say it consists of $\frac{X}{K}$ **big step(s)**.

Note

One move may consist of one small step or **many** big steps. Do not confuse the notations!

Subtask 5

Claim 2

It is optimal to avoid moving left with big steps.

Idea of Proof

Otherwise, we may cancel a “left” big step with a suitable “right” big step or K suitable “right” small steps, without affecting the validity of the solution.
Such cancellation will not increase the number of moves.

Subtask 5

Claim 2

It is optimal to avoid moving left with big steps.

Idea of Proof

Otherwise, we may cancel a "left" big step with a suitable "right" big step or K suitable "right" small steps, without affecting the validity of the solution.
Such cancellation will not increase the number of moves.

- Therefore, if we make B big steps (to the right), we will need to make $|D - K \times B|$ small steps (could be to the left or to the right).

Subtask 5

Claim 3

It is optimal to take $B = C$ or $B = C + 1$ (recall that $C := \lfloor \frac{D}{K} \rfloor$).

Proof

If $B' < C$, compare with $B = C$. Number of small steps increases, while number of moves for the big steps decreases by at most one.
If $B' > C + 1$, compare with $B = C + 1$. Number of small steps increases, while number of moves for the big steps does not decrease.

Subtask 5

- Recall that $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.

Subtask 5

- Recall that $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.
- Two cases to consider: $B = C$ big steps, or $B = C + 1$ big steps.

Subtask 5

- Recall that $D := E_c - S_c$, and $C := \lfloor \frac{D}{K} \rfloor$.
- Two cases to consider: $B = C$ big steps, or $B = C + 1$ big steps.

Case 1 ($B = C$)

- If $B = 0$, an optimal solution is `right 1` (D times).
- Otherwise, an optimal solution is `right` $K \times B$; `right 1` ($(D - K \times B)$ times).

Subtask 5

Case 2 ($B = C + 1$)

- If $K \geq M$, we should disregard this case. Otherwise,
- if $K \times B < M$, we need $1 + (K \times B - D)$ moves;
- if $K \times B \geq M$, we need $2 + (K \times B - D)$ moves.

Subtask 5

Case 2 ($B = C + 1$)

- If $K \geq M$, we should disregard this case. Otherwise,
- if $K \times B < M$, we need $1 + (K \times B - D)$ moves.
- if $K \times B \geq M$, we need $2 + (K \times B - D)$ moves.

Again, it is obvious that the number of moves is optimal.
So, it remains to construct a solution with the given number of moves (not easy!).

Subtask 5

Case 2a ($B = C + 1$, $K \times B < M$)

Set $REMAIN := (K \times B - D)$, $LOC := S_c$, $GOAL := E_c$.
Then perform the following:

1. while $REMAIN > 0$ and $LOC > 1$

   `move left 1`

   $REMAIN := REMAIN - 1$

   $LOC := LOC - 1$

2. `move right` $K \times B$; $LOC := LOC + K \times B$
3. while $LOC > E_c$

   `move left 1`

   $LOC := LOC - 1$

Subtask 5

Case 2b ($B = C + 1$, $K \times B \geq M$)

Set $REMAIN := (K \times B - D)$, $LOC := S_c$, $GOAL := E_c$.
Then perform the following:

1. while $REMAIN > 0$ and $LOC > 1$
   `move left 1`
   $REMAIN := REMAIN - 1$
   $LOC := LOC - 1$
2. `move right` K
3. while $REMAIN > 0$
   `move left 1`
   $REMAIN := REMAIN - 1$
4. `move right` $K \times (B - 1)$

Subtask 5

Finally, choose the case with fewer moves, and find a sequence of moves as described.

Note

From the construction above, we see that the number of moves is $O(K)$ with a reasonably small constant.

Subtask 5

Here are some examples. Let's dry-run them!

M	K	S_c	E_c
10	10	1	10
8	4	1	8
18	6	2	17
19	6	2	17
10	8	4	9
10	8	5	9

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.
- The key is to handle all directions using *one single function*!

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.
- The key is to handle all directions using *one single function*!
- Use a function like `void solve_1D(int dim, int S, int E, int K, string move_front, string move_back)`.

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.
- The key is to handle all directions using *one single function*!
- Use a function like `void solve_1D(int dim, int S, int E, int K, string move_front, string move_back)`.
- Then, if $S_r < E_r$, call `solve_1D(`N`, `S_r`, `E_r`, `K`, "right", "left")`; otherwise, call `solve_1D(`N`, `$N+1-S_r$`, `$N+1-E_r$`, `K`, "left", "right")`.

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.
- The key is to handle all directions using *one single function*!
- Use a function like `void solve_1D(int dim, int S, int E, int K, string move_front, string move_back)`.
- Then, if $S_r < E_r$, call `solve_1D(`N`, `S_r`, `E_r`, `K`, "right", "left")`; otherwise, call `solve_1D(`N`, `$N+1-S_r$`, `$N+1-E_r$`, `K`, "left", "right")`.
- Similarly, if $S_c < E_c$, call `solve_1D(`M`, `S_c`, `E_c`, `K`, "down", "up")`; otherwise, call `solve_1D(`M`, `$M+1-S_c$`, `$M+1-E_c$`, `K`, "up", "down")`.

But this is TOO HARD to code...

- My C++ solution has about 70 lines, including 15 lines of headers.
- The key is to handle all directions using *one single function*!
- Use a function like `void solve_1D(int dim, int S, int E, int K, string move_front, string move_back)`.
- Then, if $S_r < E_r$, call `solve_1D(`N`, `S_r`, `E_r`, `K`, "right", "left")`; otherwise, call `solve_1D(`N`, `$N+1-S_r$`, `$N+1-E_r$`, `K`, "left", "right")`.
- Similarly, if $S_c < E_c$, call `solve_1D(`M`, `S_c`, `E_c`, `K`, "down", "up")`; otherwise, call `solve_1D(`M`, `$M+1-S_c$`, `$M+1-E_c$`, `K`, "up", "down")`.
- Essentially, we are flipping the board.

The End

- Questions?