# POST YOUR APPROACH



vlad_D
5 posts Wed, May 5, 2021 at 02:47 PM Marathon Match 126

My solution was a basic beam search with no real state scoring, but break ties of achieved score using some simple heuristics.

- try all the 1-move moves and expand (I wanted to add moves where I would take square and send it to the hole min-path, or take a segment and send all of them to hole. Was not successful initially, so I did not spend more time.)
- change the width of the BEAM dynamically depending on the time left.

It looked pretty decent, but watching the viz I could see some silly moves that I couldn't fix with the heuristics.

I suspect there would be some SA, random walks around greedy, etc solutions. I'm curious how the top pushed for the last 0.1% improvement

View: Threaded | Flat +4 Quote · Reply

## COMMENTS



Daiver19 Re: Post your approach (response to post by vlad_D)
1 posts Wed, May 5, 2021 at 03:19 PM

I believe the problem was well suited for greedy approaches: consider you have a sequence of D1 moves which put a value V1 into a hole and a sequence D2 which puts a value V2 into a hole and you want to order them. It's easy to prove, that we should start with V1 only if V1/D1 > V2/D2 (no matter the multiplier).

I just did randomized greedy algorithm multiple times. Basically, I iteratively picked the best move across 2 types of moves:
1) Run BFS to find all the possible singular value moves and pick one with maximum V/D (in case of a tie pick at random)
2) For every hole and every line leading directly to it look at all the move sequences which look like the following: N (possibly 0) slide moves leading directly to the hole, then K (possibly 0, N + K > 0) move pairs from

an orthogonal direction (i.e. we do a move to put a piece onto our line, then slide it to the hole). Since orthogonal slides are not always possible, also try adding a piece to make it possible (with at most 2 moves). Also for orthogonal moves it's better to just put 1-values aside with a single move (when possible) instead of putting them to the hole. Among all such sequences pick one with the highest sum(V)/sum(D) (random for ties) and perform the first couple of moves (usually putting just a single value in hole).

I've tried some things (including beam-search-like approach) to improve this but didn't have much success.

+3 | Quote · Reply

**nika** Re: Post your approach (response to post by **Daiver19**)
3 posts Wed, May 5, 2021 at 03:55 PM

Main solution consists of two parts
1) Greedy initial order. Trying to minimize same V/D ratio by completing single or multiple blocks in a simple chain. There is some penalty for removing the blocks that could potentially help others get optimal 2-move later.
2) Run Hill climbing (later SA but it was very slow and didn't have much impact) by picking some subsegment of order and moving it somewhere else.

Now for remaining 0.1%
1) Allow "move aside" moves instead of just moving everything to holes
2) Run Hill climbing again after
3) Rerun everything from scratch with added randomness until there is time

+2 | Quote · Reply

**sullyper** Re: Post your approach (response to post by **Daiver19**)
13 posts Wed, May 5, 2021 at 04:29 PM

It looks like greedy do well at first look, but if we look in more details it's not that simple:
- When you make a move it changes the distance of the next move (can increase or decrease it)
- Sometimes I saw my program delay move that looked better using V1/D1 > V2/D2 because it helped to make other moves faster.
- Sometimes I say it delaying the 'best' move, because D1 will be reduced later on.

Anyways, my soltion consisted in 2 phases:
1. for 2s I generate some greedy solution where each turn I compute the best move using (V+rng(C))/D, somehow incrementing the color V with a random number gave a big boost in the greedy solution. At the end I run a LocalOptimizer
2. Then I ran a SA (mostly HC though with the choice of temperature.
- The state is a list of {i, m} where i is the tile, m is either -1 meaning shortest path to the hole, or one of the 8 moves

- Mutations are:
1) I either move randomly one of the move somehwere else
2) I add a random move somewhere randomly
3) Pick a random segment of size [2..N] and apply a random permutation
- Then apply my LocalOptimizer

The LocalOptimizer extract the paths of a tiles and create a graph of dependencies (some path might depend on some other to be done before or after).
Then for each Path, I try to move it left at all the position, when I encouter a path it cannot swap with, I add that path to my block and try to move both path, 3 when I hit another dependencies etc.
As this solution does not change the length of the path, it's easy to keep track of the gain (or loss) by keeping track of the length of block and the sum C-1 in that block.

The downside of the LocalOptimizer is it depends a lot on the path chosen to go to a hole, many times there are many different paths that will lead to a Hole in the same number of moves, I tried to pick randomly such a path and recompute it frequently, the downside is it slow down the program and hence do fewer iterations, the positive side is now the path chosen is random, so the graph of dependencies changes and the LocalOptimizer might be able to make more progress.

There are seeds where my program probably find optimal (maybe I am naive) or is close to, but they are also some seeds where clearly more time provide a big boost:
Here is my first 10 seeds, using 10s, using 2h

Seed 1: 1224*
Seed 2: 1551325 1555198 (0.25%)
Seed 3: 412226 419547 (1.75%)
Seed 4: 10113*
Seed 5: 115664 118527 (2.5%)
Seed 6: 5703*
Seed 7: 665189 674967 (1.5%)
Seed 8: 42269 42271
Seed 9: 63092 63457 (0.6%)
Seed 10: 8448 8503 (0.6%)

- means I did not run for 2h

+2 | Quote · Reply

**survival07** Re: Post your approach (response to post by vlad_D)
5 posts Thu, May 6, 2021 at 06:45 AM

My provisional score is ~ 85.0, So my approach is not up to the mark. But here is how I improved my score from first submission score of 44.0

I used SA. To move number blocks, I mostly used 'slide' operation. I used 'move' operation when the board is almost full. In every iteration, I chose two random number blocks from an ordered list of number blocks and swapped their positions.

1. Initially I randomly ordered the number blocks and then moved them to the holes one by one with shortest moves.
2. Instead of always choosing two random number blocks I fixed one index, which is the last number block index in previous iteration where the process stopped due to Z <= 1 or no way to move that block
3. Instead of having a random initial order of blocks, I used a greedy approach to sort the block based on shortest moves first, then based on block value.

Is there a way to download other's code?

+0 | Edit · Quote · Reply

**simanman** Re: Post your approach (response to post by vlad_D)
6 posts Thu, May 6, 2021 at 12:08 PM

I wrote my approach using translator tools.

https://simanman.hatenablog.com/entry/2021/05/07/010550

this is original version (in japanese)

https://simanman.hatenablog.com/entry/2021/05/06/212713

+2 | Quote · Reply

## LEAVE A COMMENT

Comment ...

16000 character remaining

PREVIEW POST COMMENT