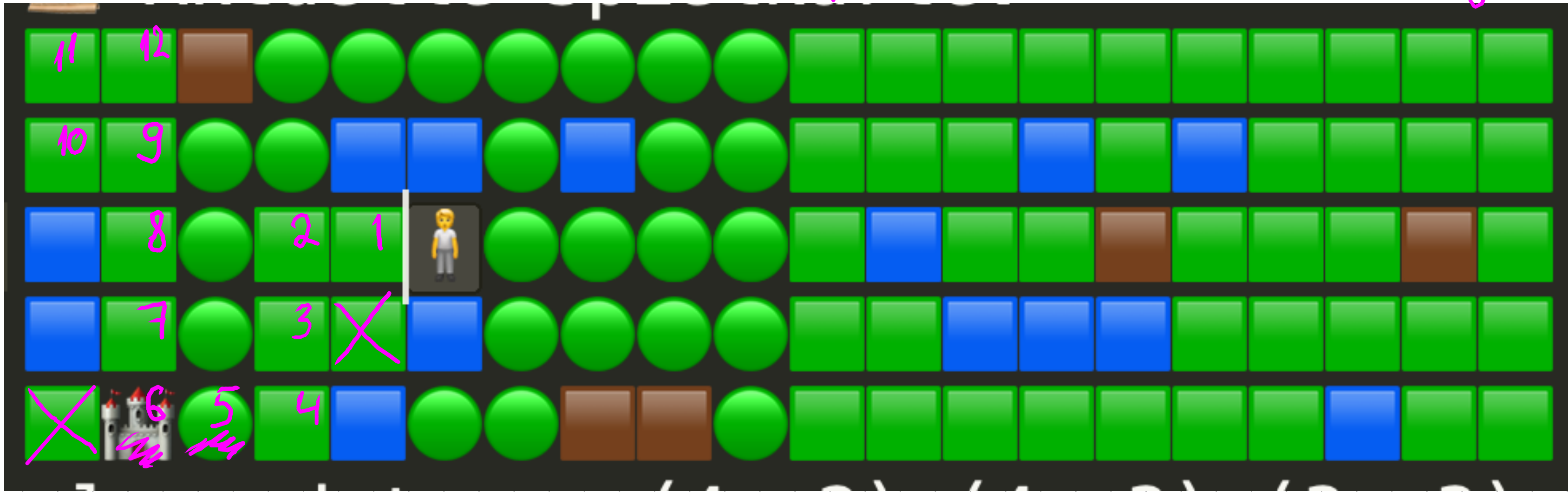
$12 \cdot 2 \simeq 24$ round

profit = 10 new

$\rho = \frac{10}{24} = 0,42$



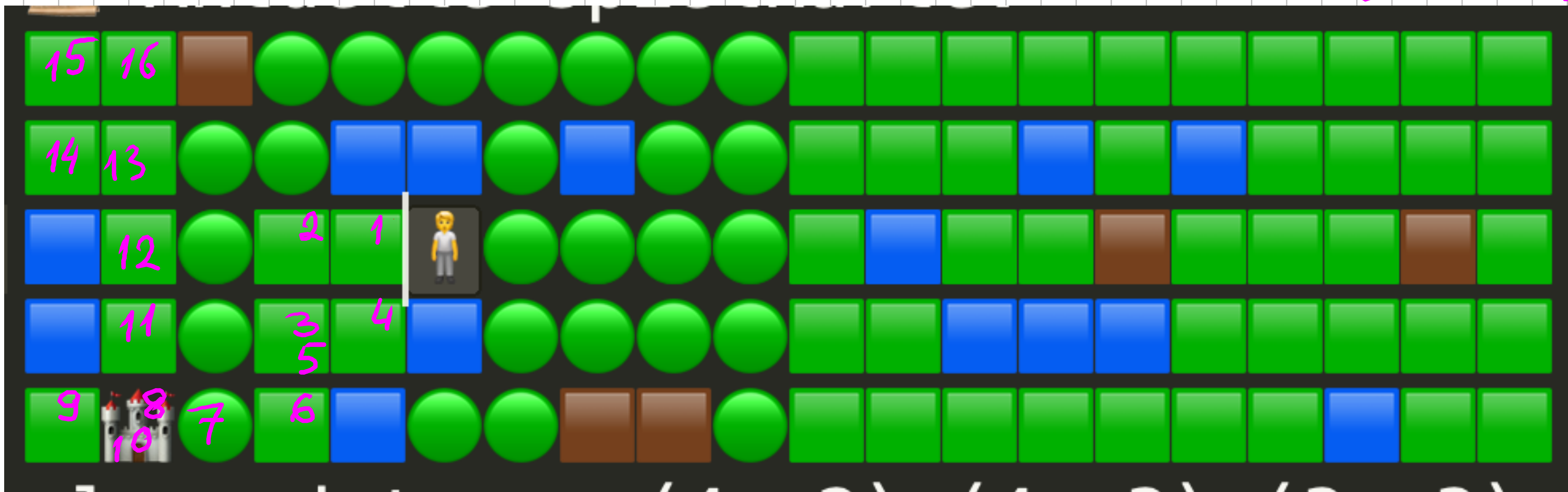
6 moves 4 new

8 moves 6 new

$16 \cdot 2 \simeq 32$ rounds

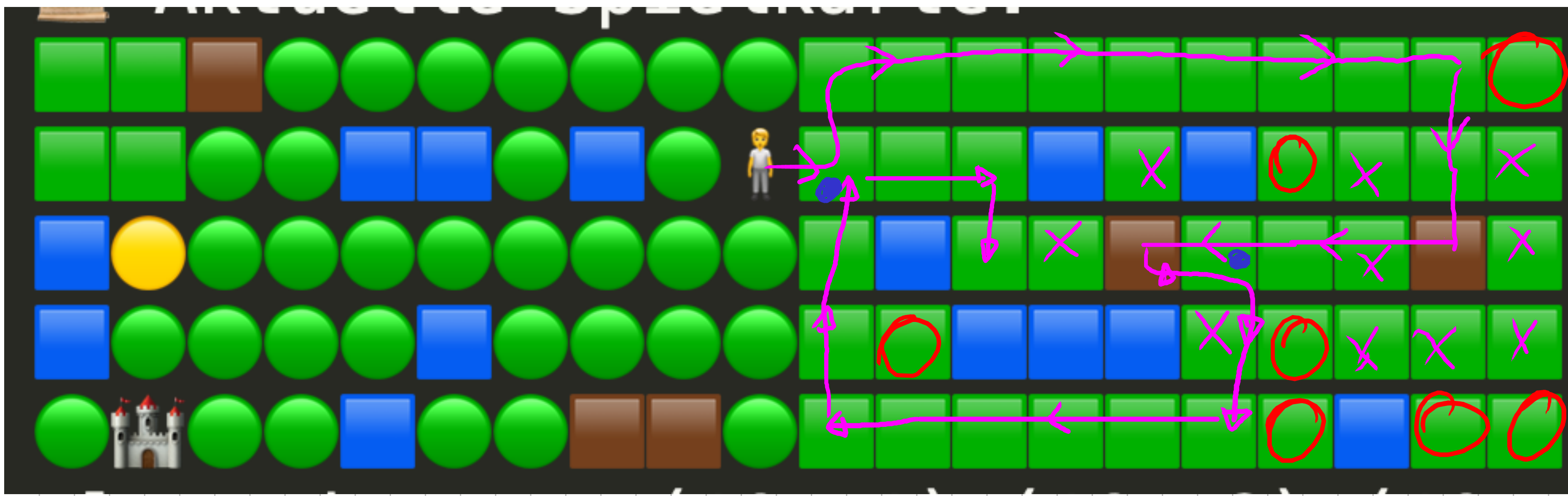
profit = 12 new

$\rho = \frac{\text{profit}}{\text{cost}} = \frac{12}{32} = 0,38$



6 moves 5 new

8 moves 5 new

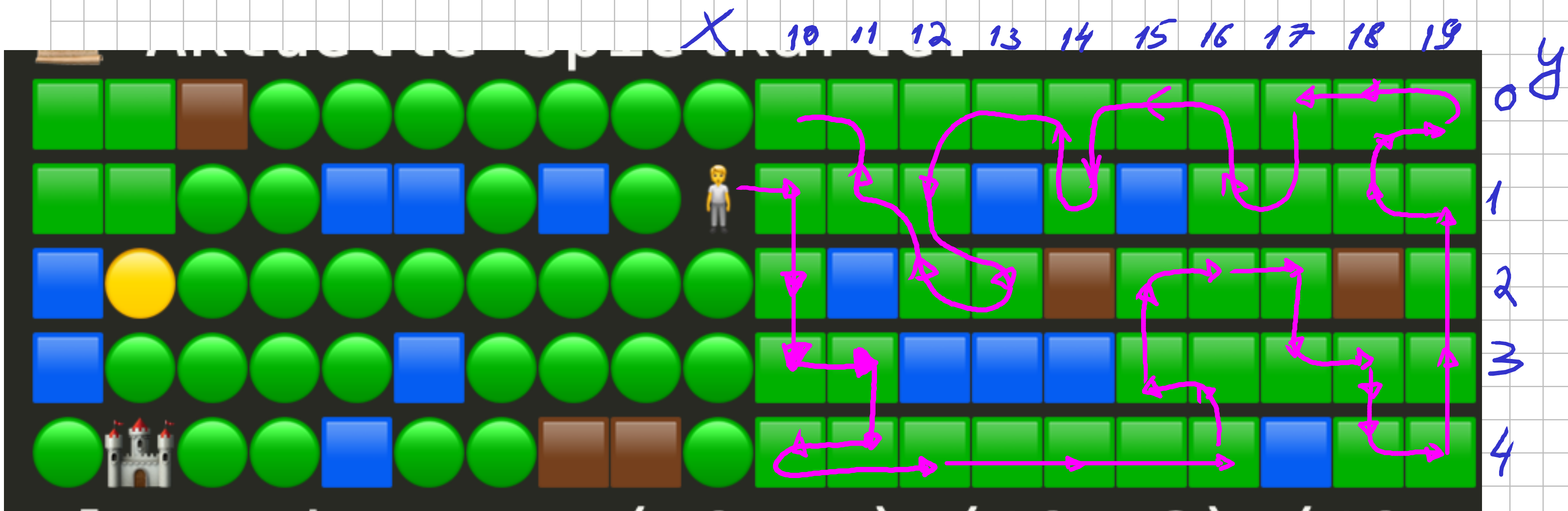


profit = 34

cost = 64

$\rho = 0{,}53$

horizon n moves $n \sim 5 \approx 10$



when optimizing for horizon: Pockets = bad, Mountains = good

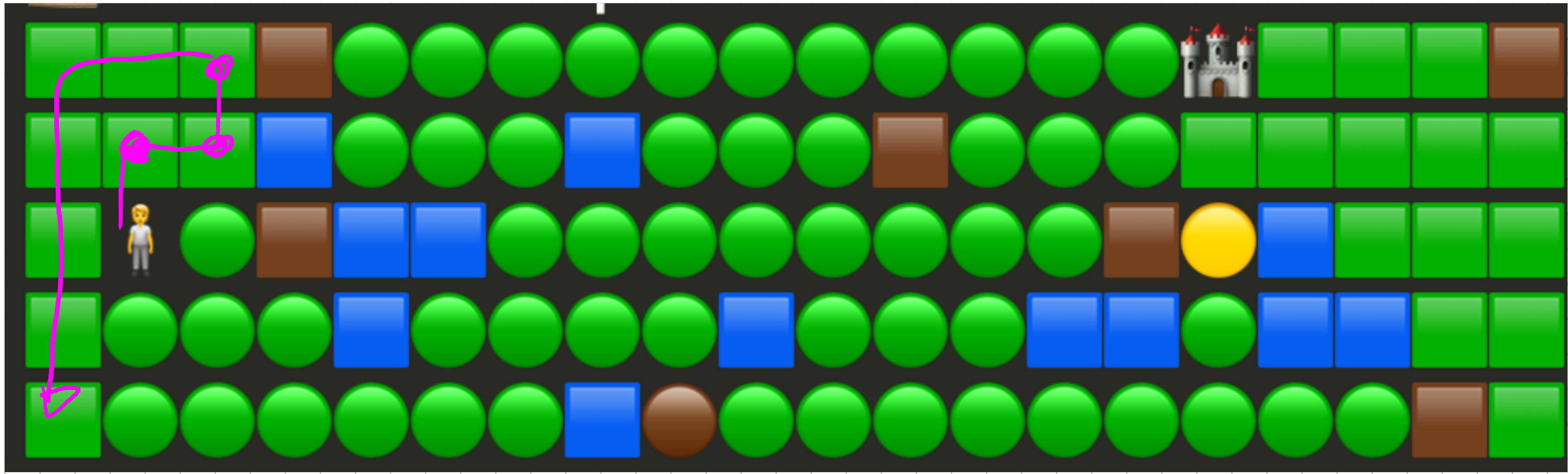
Metrics - numerical characteristics of something. often used for comparison

Pocket

0 1 2 3

$M = G + G$

0
1
2
3
4



For density metrics Pockets are good

For horizon metrics Pockets are bad

Q: Why does algoTSP prioritize Pockets when Pockets become available?

A: because is we want full global tour to be the shortest, we need to visit Pockets asap

Observation: When there is a fork with a Pocket, visiting the Pocket means having a shorter global tour. But ignoring the Pockets means picking up Gold faster.

Q: Why does it happen that if we ignore Pockets we find Gold faster (on average) ?

List of Tasks:

1) deadline on practical homework (diagrams)
   - we need to make up our mind about program architecture
2) Improving StrategyTSP
   - integrating Mountain terrains
   - better tour comparison (with respect to time horizon)

Plans:

1) Come up with an idea of how to integrate Mountains & Cost Comparison
2) Write down complete formal algorithm for StrategyTSP
3) Think about how to implement it in code
4) Make adjustments to the code structure (architecture)

5,6) Implement the algorithm

5,6) Create diagrams

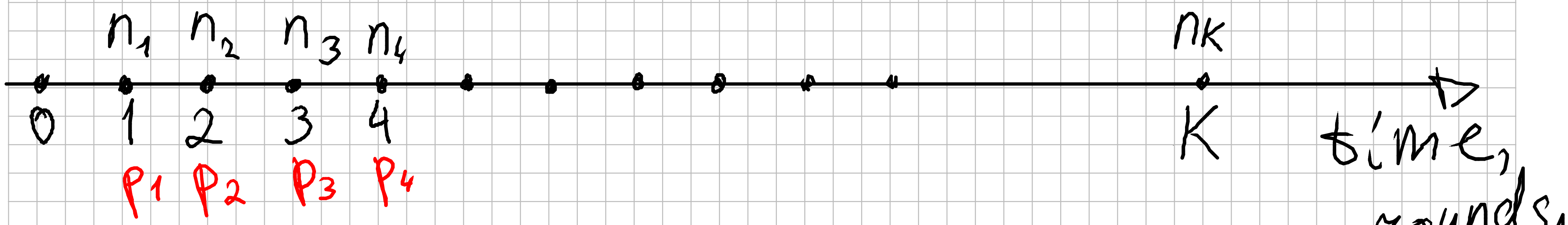
Better Cost comparison when choosing "the best" tour.

TSP generates lots of different tour candidates. How do we choose the best one among them?

Right now: we choose the shortest tour, a.k.a. lowest cost

Problems with this approach:
we do complete traversal, metrics = length of complete tour (# rounds)
What do we REALLY care about? Finding Gold asap.

$n_i = 0, 1, 2, \ldots$ – amount of new grass explored per round



$$\sum_i n_i = n_1 + n_2 + n_3 + \ldots + n_K = \#\text{Grass on the map} =$$

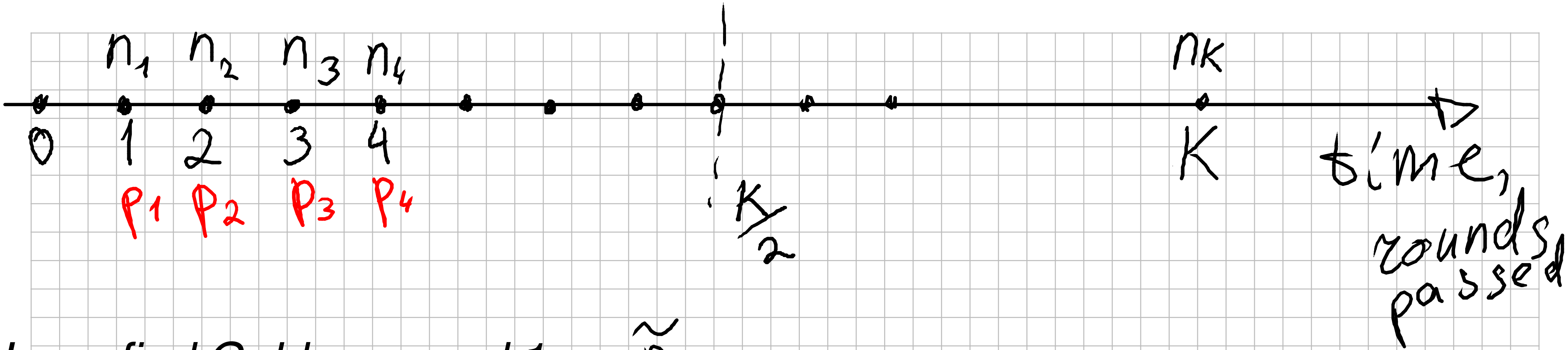
$$= N_G \lesssim K$$

We have a complete tour with length of K rounds
It explores all the grass N_G nodes

if $n_j > 0 \Rightarrow$ *there is a chance that we found (observed) gold during round j*

p_j – *probabily of finding gold during round j* $= \dfrac{n_j}{N_G - \sum_{i<j} n_i} = \dfrac{n_j}{\sum_{i \geq j} n_i}$

$$\tilde{p}_j = (1-p_1)(1-p_2)\ldots(1-p_{j-1}) \cdot p_j$$



I can find Gold on round 1 $\tilde{p}_1$

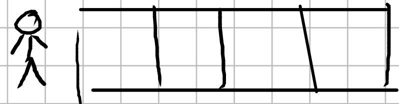
I can find Gold on round 2 $\tilde{p}_2$

I can find Gold on round 3 $\tilde{p}_3$

I can find Gold on round 4 $\tilde{p}_4$

...

I can find Gold on round K $\tilde{p}_K$



$\tilde{p}_1 = 25\% = 1/4$

$\vdots$

$\tilde{p}_4 = 25\% = 1/4$

$$\Sigma = 1\cdot\frac{1}{4} + 2\cdot\frac{1}{4} + 3\cdot\frac{1}{4} + 4\cdot\frac{1}{4} = \frac{1}{4}\left[1+2+3+4\right] = \left[4 = K\right] = \frac{1}{K}\cdot\sum_{K} i = \frac{1}{K}\cdot\frac{K(K-1)}{2} = \frac{K-1}{2}$$

On average $1\cdot\tilde{p}_1 + 2\cdot\tilde{p}_2 + 3\cdot\tilde{p}_3 + 4\cdot\tilde{p}_4 + \ldots + K\cdot\tilde{p}_K$

Math Expectation shows how much time we need on average to find gold tour 1 1 1 1 1 1 0 0 0 0 1

tour 2 0 0 1 0 1 0 0 1 1 1

Tour 1 is preferable because it has non-zeros closer to start

$$ppos, \underbrace{[g_1, g_2, \ldots, g_n]}_{goals}$$

$$Ppos \rightarrow g_1 \xrightarrow{cost_{12}} g_2 \rightarrow \ldots$$

$n_1 \quad n_2 \quad n_3 \quad n_4 \quad \ldots \quad n_k$

$\alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \alpha_4 \quad \ldots \quad \alpha_k \qquad \sum_{i=1}^{k} n_i = N_G$

$\alpha_i \geq \alpha_j \quad i < j$ *non-ascending sequence*

$$\sum_{i=1}^{k} \alpha_i \cdot n_i$$

$$\alpha_m = \gamma^m, \quad 0 < \gamma \leq 1$$

Formal algorithm for tour with *good* profit comparison

0. Generate a continuous tour (complete tour over all nodes)

1. Suppose we have a continuous tour (all nodes in tour are neighbours)

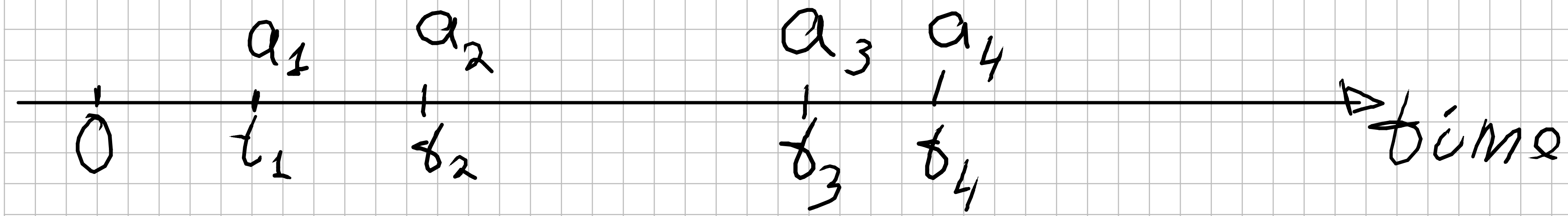
(0,0) (0,1) (0,2) (1,2)

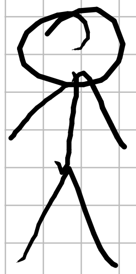
2. Calculate number of newly explored grass per round

n1, n2, n3, n4, ...

3. Calculate probabilities of finding gold at each round
and calculate Math Expectation of number round until gold is found

Or as alternative calculate gamma-discounted summ of rewards





1 25%	2 25%
4 25%	3 25%

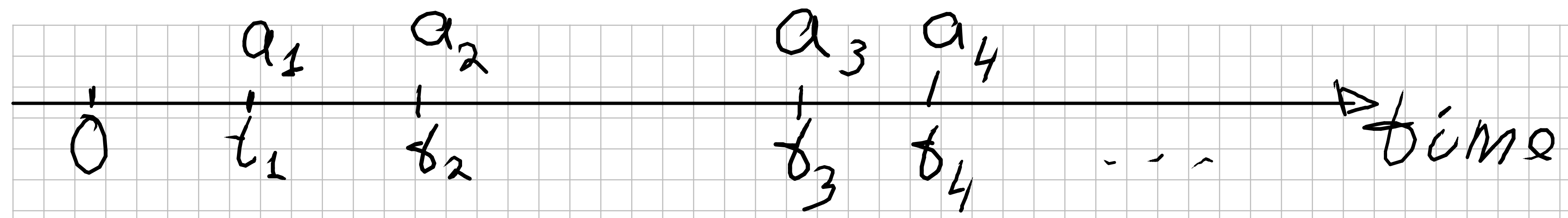
$$p_1 = \frac{1}{4} \qquad \tilde{p}_1 = p_1 = \frac{1}{4}$$

$$p_2 = 1/3 \qquad \tilde{p}_2 = \left(1 - \frac{1}{4}\right) \cdot \frac{1}{3} = \frac{3}{4} \cdot \frac{1}{3} = 1/4$$

$$p_3 = 1/2 \qquad \tilde{p}_3 = \frac{3}{4} \cdot \frac{2}{3} \cdot \frac{1}{2} = 1/4$$

$$p_4 = 1$$

$$\tilde{p}_4 = \frac{3}{4} \cdot \frac{2}{3} \cdot \frac{1}{2} \cdot 1 = 1/4$$



a_j — How much grass was discovered at timestamp j

$$\sum_j a_j = \#Grass$$

$$E(t) = \sum_j t_j \cdot \tilde{p}_j = \left[\tilde{p}_j = \frac{a_j}{\#Grass}\right] = *$$

$\tilde{p}_j$ — prob. to observe gold at timestamp j

$$T = \max_j (t_j) \leq \#G$$

$$* = \sum_j t_j \cdot \frac{a_j}{\#G} = \sum_j a_j \cdot \frac{t_j}{\#Grass} =$$

$$\frac{t_j}{\#Grass} = \alpha(t)$$

$$= \sum_t a(t) \cdot \alpha(t)$$

$$\sum_j a_j \cdot \frac{\delta_j}{\#G} = \left[T = \max_j \delta_j \right] =$$

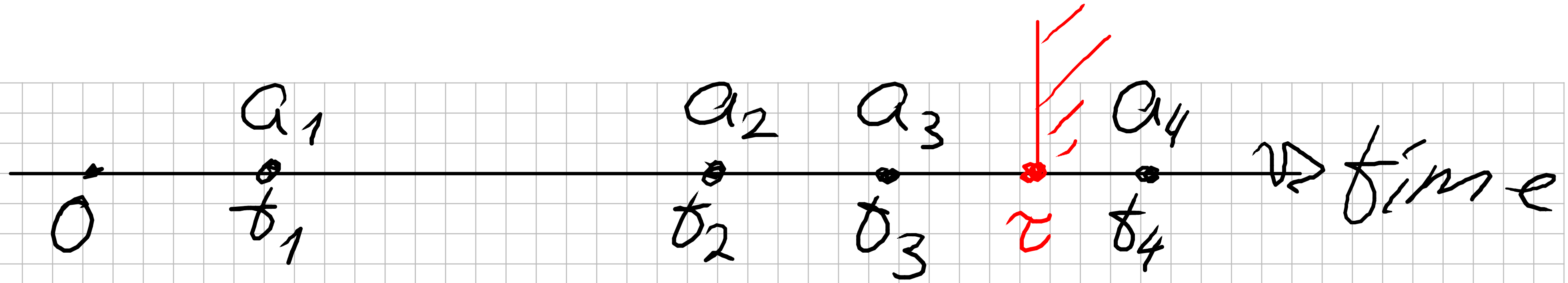
$$= \sum_j a_j \frac{T - T + \delta_j}{\#G} = \sum_j a_j \frac{T}{\#G} - \sum_j a_j \frac{T - \delta_j}{\#G} =$$

$$= \frac{T}{\#G} \underbrace{\sum_j a_j}_{= \#G} - \sum_j a_j \frac{T - \delta_j}{\#G} =$$

$$= T - \sum_j a_j \frac{T - \delta_j}{\#G} = \left[\frac{T - \delta_j}{\#G} \text{ non-negative descending } = \lambda(t) \right]$$

$$* = T - \sum_t a(t) \cdot \lambda(t) \longrightarrow \text{minimize}$$

$$\sum_t a(t) \cdot \lambda(t) \longrightarrow \text{maximize}$$



Assume our enemy finds his gold at time τ

Then myreward after τ *are worth nothing*

$$\alpha(t) = \begin{cases} \dots, & t < \tau \\ 0, & t \geq \tau \end{cases}$$

$$\beta(\tau) = 1$$

Generalisation:

$$\sum_t a(t) \cdot \left(1 - \beta(t)\right), \quad \beta(t)$$

probability that enemy wins the game by time t

AC tours generation:
1. Ppos, map, Set visited
2. find unvisited Grass closest to Ppos (BFS)
3. move to that node
4. repeat from (1)

TSP tours generation (Gold):
1. Ppos, map, Set visited
2. Goals = all Grass + on my side + not visited
3. choose 'g' from G
- *a) let 'n' = 'g'*
- *b) find node 'n2' from G closest to 'n' (BFS)*
- *c) remove 'n' from Goals*
- *d) let 'n' = 'n2', repeat from (b) until Goals are not exhausted*

4. repeat (3) for each goal in Goals
*5. among all generated tours choose the best**

TSP tours generation (Gold) + Mountains:

1. Ppos, map, Set visited

2. Goals = all Grass + on my side + not visited,
and all Mountains + oms + nv.

3. choose 'g' from G

a) let 'n' = 'g', let tour = List<node>,

a0) if 'g' is a bad Mountain, skip the whole tour.

aa) assume 'n' is Grass

ab) assume 'n' is Mountain

```
for all neighbours:
    Goals.remove(neighbour)
if nothing was removed:
    Goals.remove('n')
    let 'n' = tour.last()
```

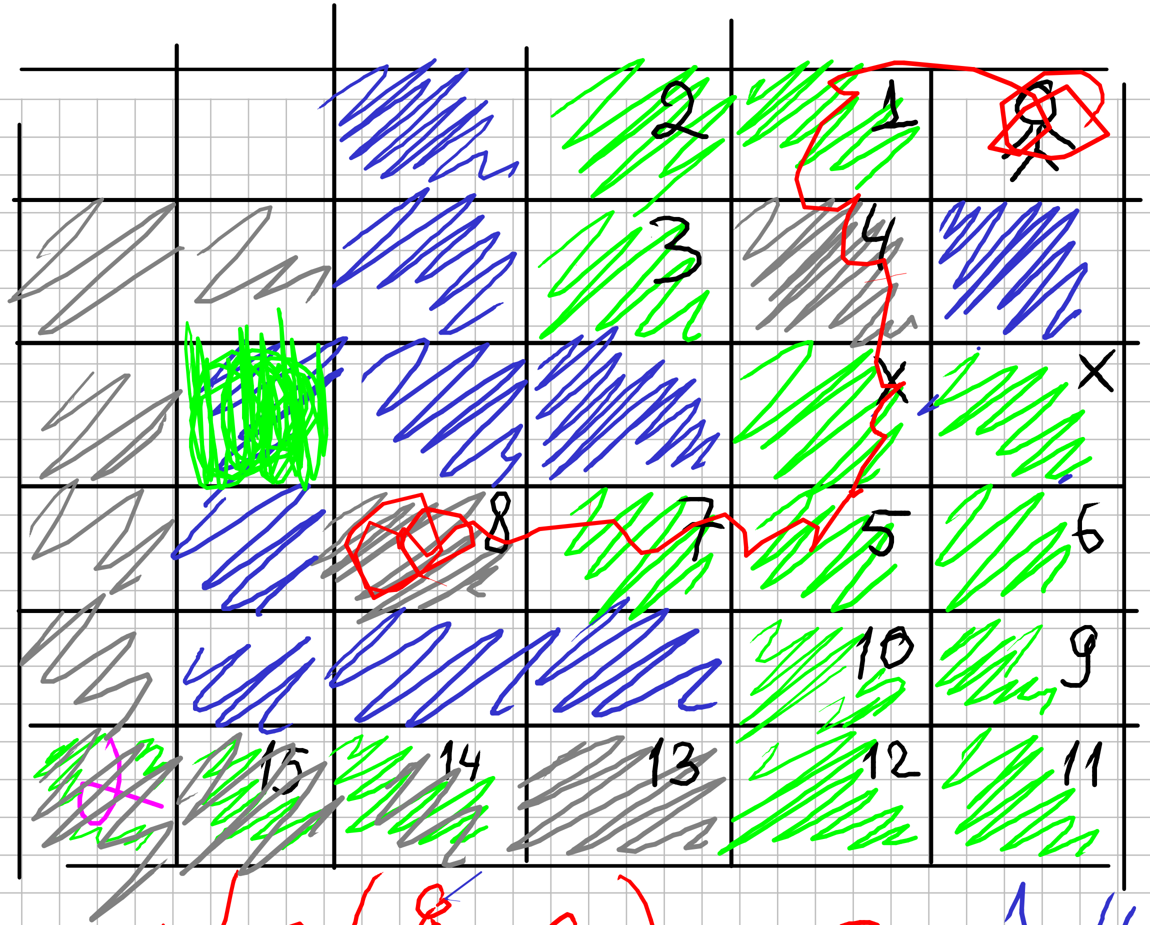
b) find node 'n2' from G closest to 'n' (BFS)

c) remove 'n' from Goals, tour.add('n')

d) let 'n' = 'n2', repeat from (b) until Goals are not exhausted

4. repeat (3) for each goal in Goals

*5. among all generated tours choose the best**



$g = 1$

tour =
1-2-3-4-5

1-4-5

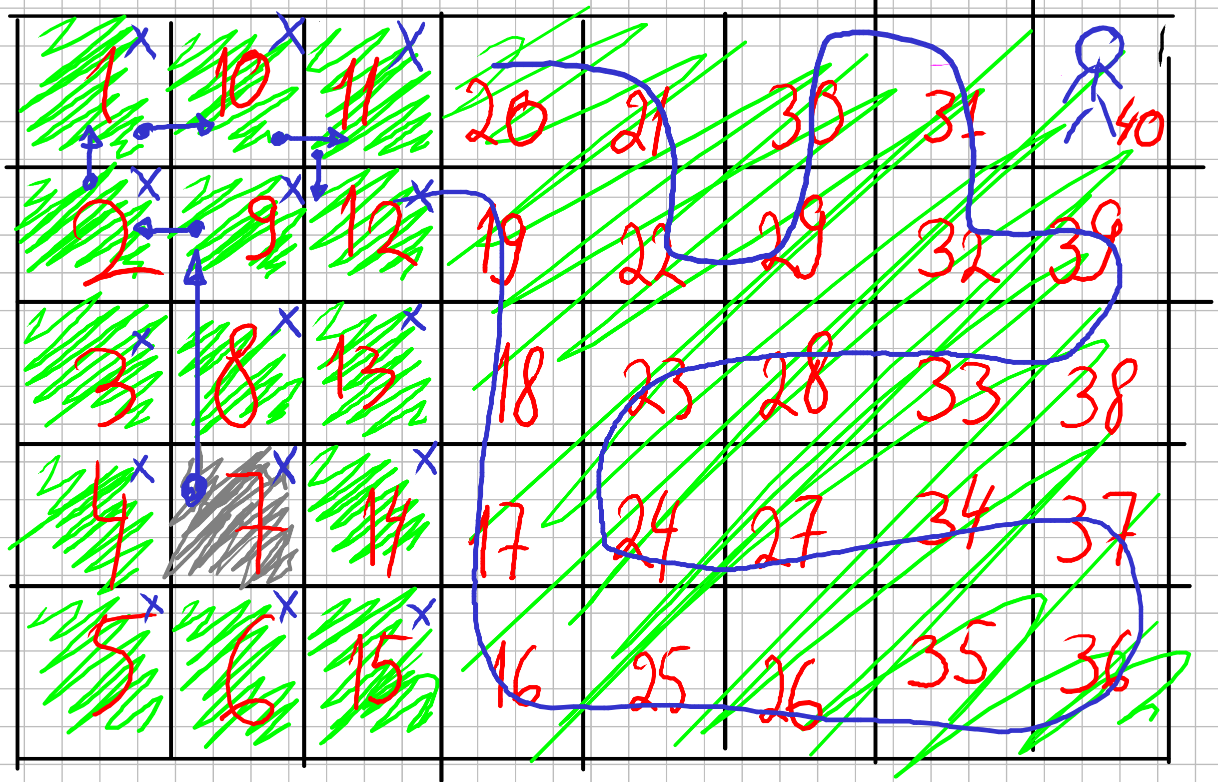
$g = 2$

$$\text{route}(\;,\,8) = [\;,\overset{1}{1},\overset{4}{4},\overset{0}{x},\overset{1}{5},\overset{1}{7},\overset{1}{8}]$$

$2 \in \text{Goals}$

-7 -BFS-

Goals = {1, . . . 39}



1
4 ∘ 2
3

$g = 7$

```
route = continuous_path(40, 7)
for node in route:
        Goals.remove(node)
```

$n = g = 7$

tour = { }

$n2 = 9$

tour = {7, 9}

$n = 9 \quad n2 = 2$

tour = {7, 9, 2)

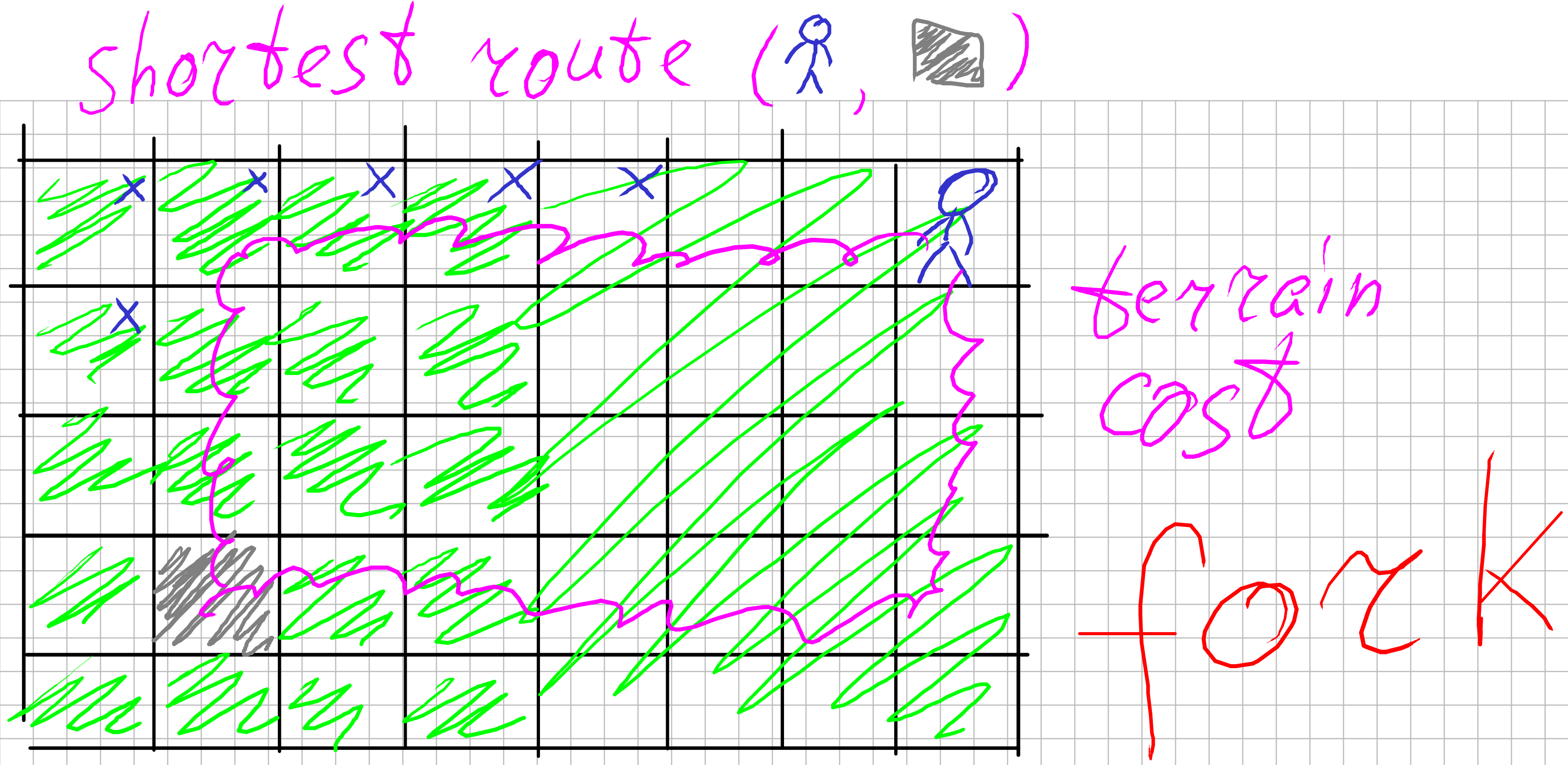
$n = 2 \quad n2 = 1 \quad t = (7, 9, 2, 1)$

$n = 1 \quad n2 = 10 \quad t = (7, 9, 2, 1, 10)$

$n = 1 \quad n2 = 11 \quad t = (\ldots, 1, 10, 11)$

$n = 11 \quad n2 = 12 \quad t = (\ldots 11, 12)$

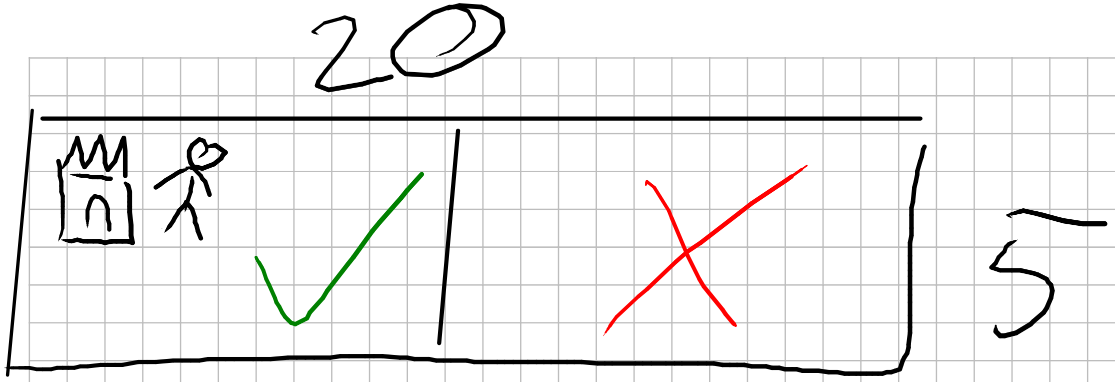
continuous



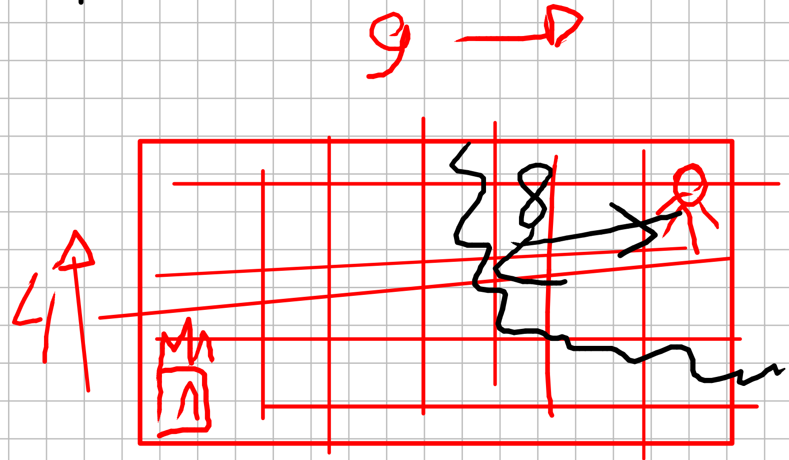
We want pathfinder to return shortest continuous route, but if there are alternavite shortest routes then return the one with most unexplored nodes (unexplored ~ Goals)

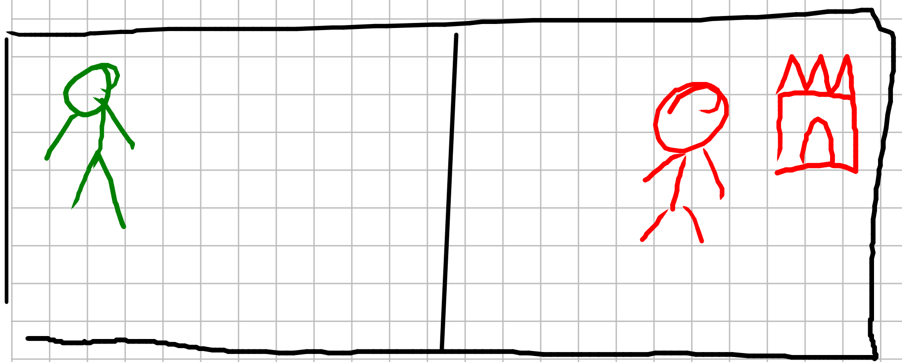
new_terrain_cost = terrain_cost - a * (if node unexplered) + b * (noise)

new_route_cost = fair_cost - a * (number_of_explored_goals)



$4 \cdot 2 + 9 \cdot 2 = 26$

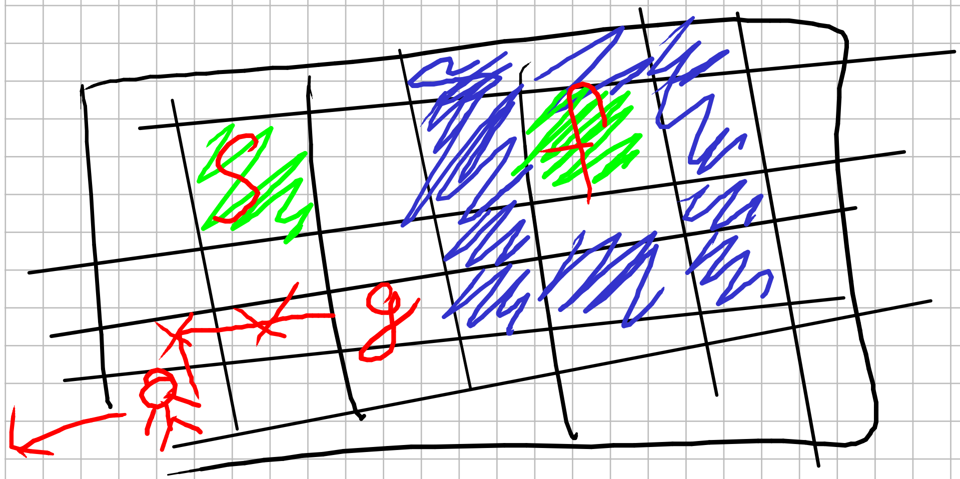


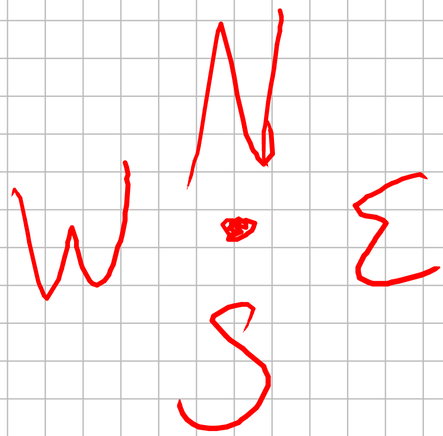


$n = 8$ turns

random enemy pos.

$$\text{find\_path}(\text{start}, \text{finish}) = [\text{start}, \ldots, \text{finish}]$$





$n = 5 \times 10 = 50$

$49 \cdot 48 \cdot 47 \cdot 46$

[N, ...

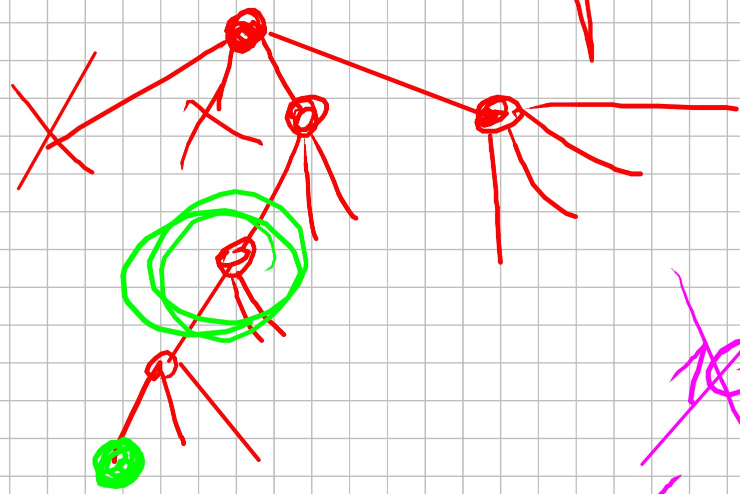
[W, ...

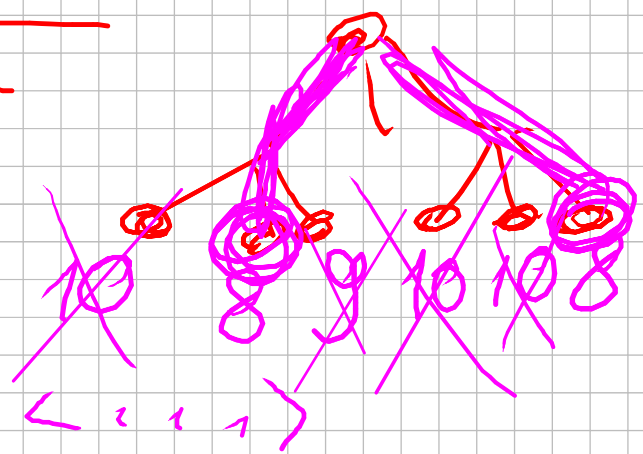
[S

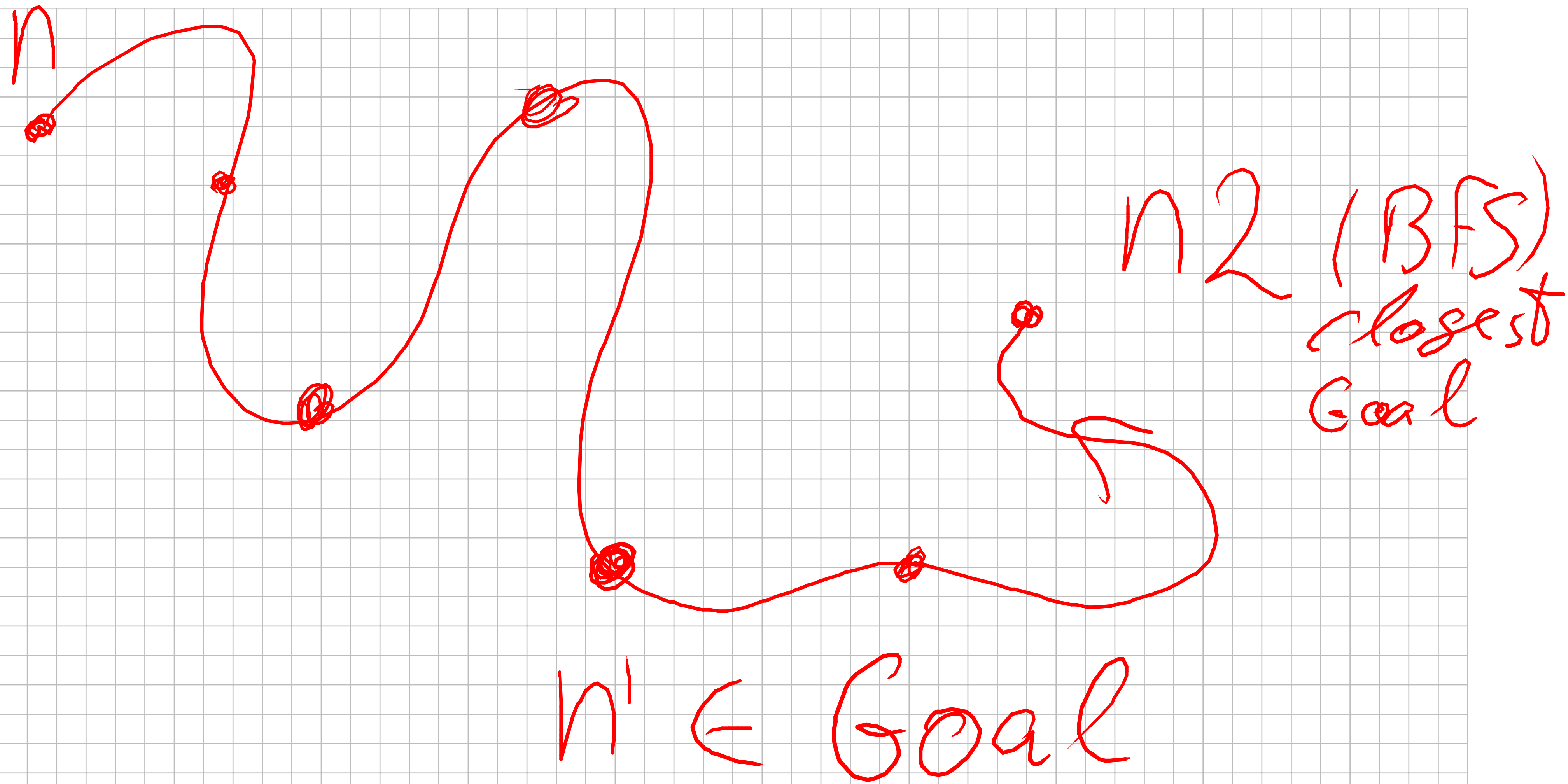
[E, ...

AlphaStar
AlphaZero

RL
planning tree






n
n2 (BFS)
closest
Goal
n' ∈ Goal

struct { node
cost
came_from }

PQ

HashMap best

continuity:

$n_1 \quad (x_1 \quad y_1)$

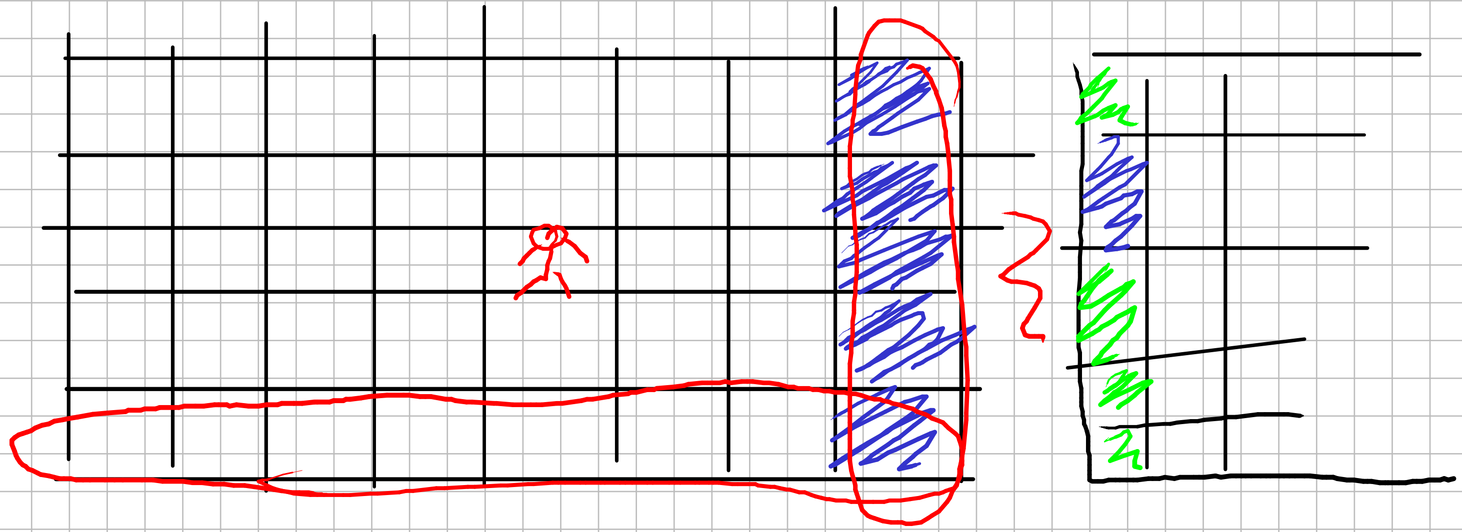
$n_2 \quad (x_2 \quad y_2$

$\Delta x = x_1 - x_2$

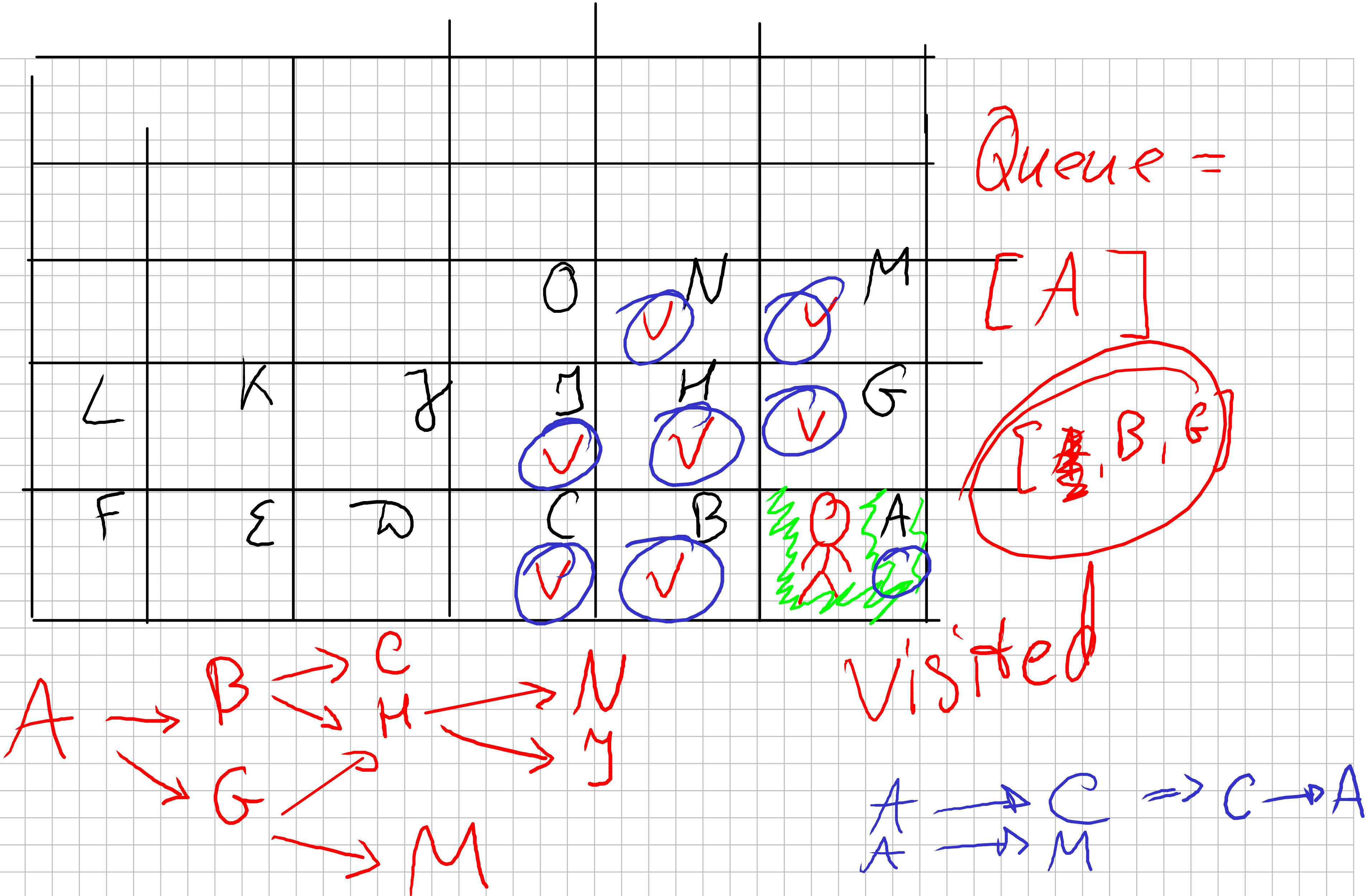
$\Delta y = y_1 - y_2$

$\Delta x^2 + \Delta y^2 = 1$

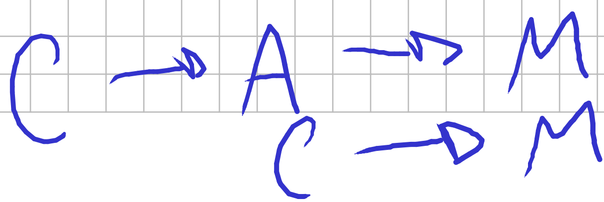
```
n2 = BFS
if n2 == Mountain:
        int old_size = Goals.size()
        for mountain_nb:
                if mountain_nb is Grass and in Goals
                Goals.remove(mountain_nb)
        if (olg_size == Goals.size())
                Goals.remove(n2)
                continue

path = continuous_path(n, n2)
tour.add(path)
n = n2
```



$\leq 49\%$ Water



Statement: such procedure will (sooner or later) list all nodes reachable from root node (A)



What if developer of the library (dependancy) decides to change some of it's functionality (maybe API) ?

it means: maybe some functions change their name, or return types, or even what they do

FullMapNode loses methods getX getY
node.getCoordinates().x()
deprecated

//System.out.println()

Logger (verbose=True/False)

java -jar jarpath _ _ ____ NN

0 1 2 3

/build/libs/Example

ac

manual

-url=text -gameid=text

-debug=bool → Fake Network

→ Server

-verbose=bool

-strategy=int

args

main {

register

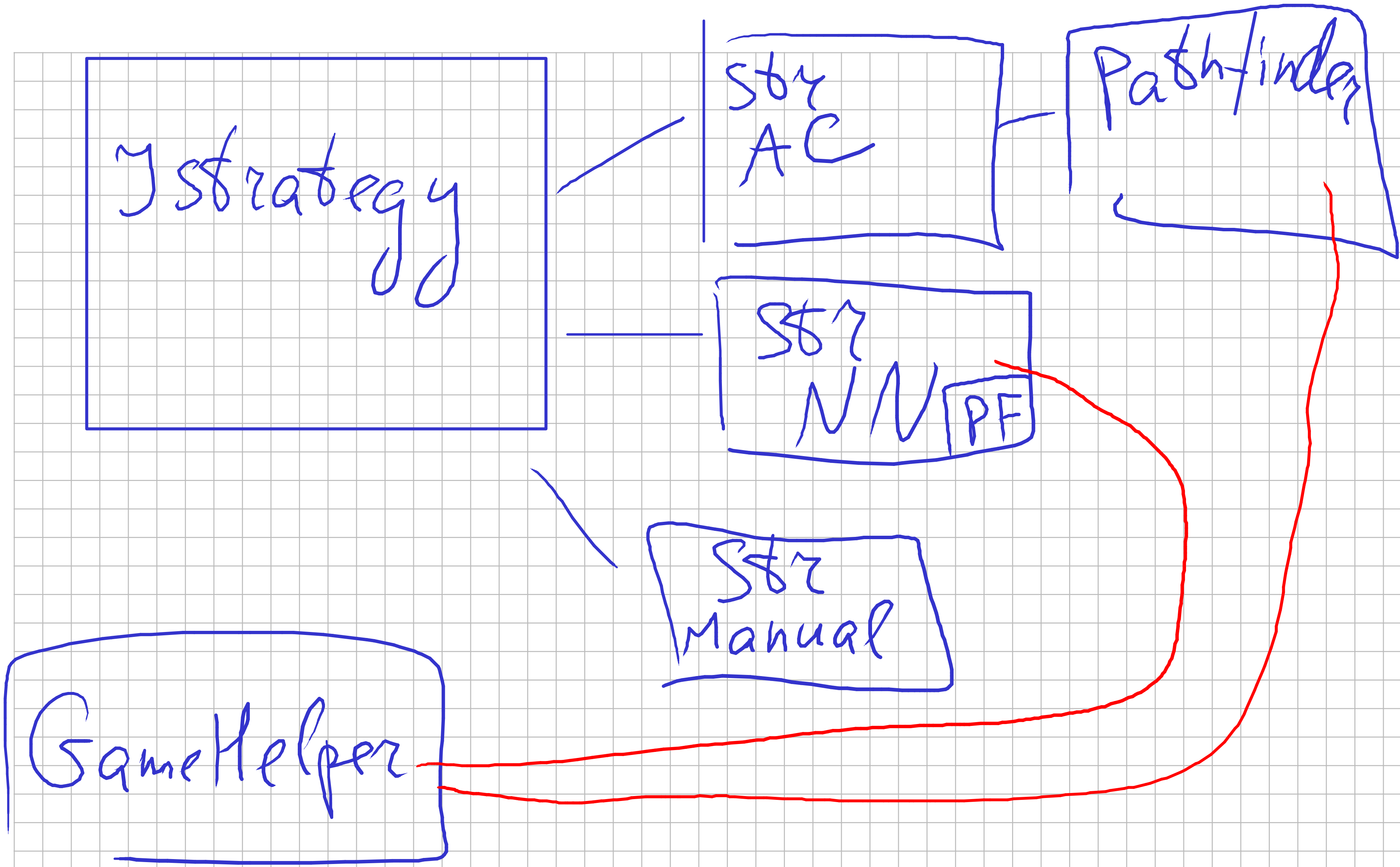
creat map

while true : move

}

```
main {
    args
    rendering.main()
}

rendering: clientmain
    render()
    event_handler {
        on button pressed:
            clientmain.calculatemove
            net.sendmove
```

```
network.setid ("Dima")
 n.sendmove (up)
network.setid ("Boris")
 n.sendmove (Down)
```


IStrategy
Str AC
Pathfinder
Str NN
PF
Str Manual
GameHelper

i8 binary repr

<< : i8 → i8

0	0	0	0	1	0	1	0
0	1	2	3	4	5	6	7

$10_{10} = 1010_2$

i8 = bool[8]

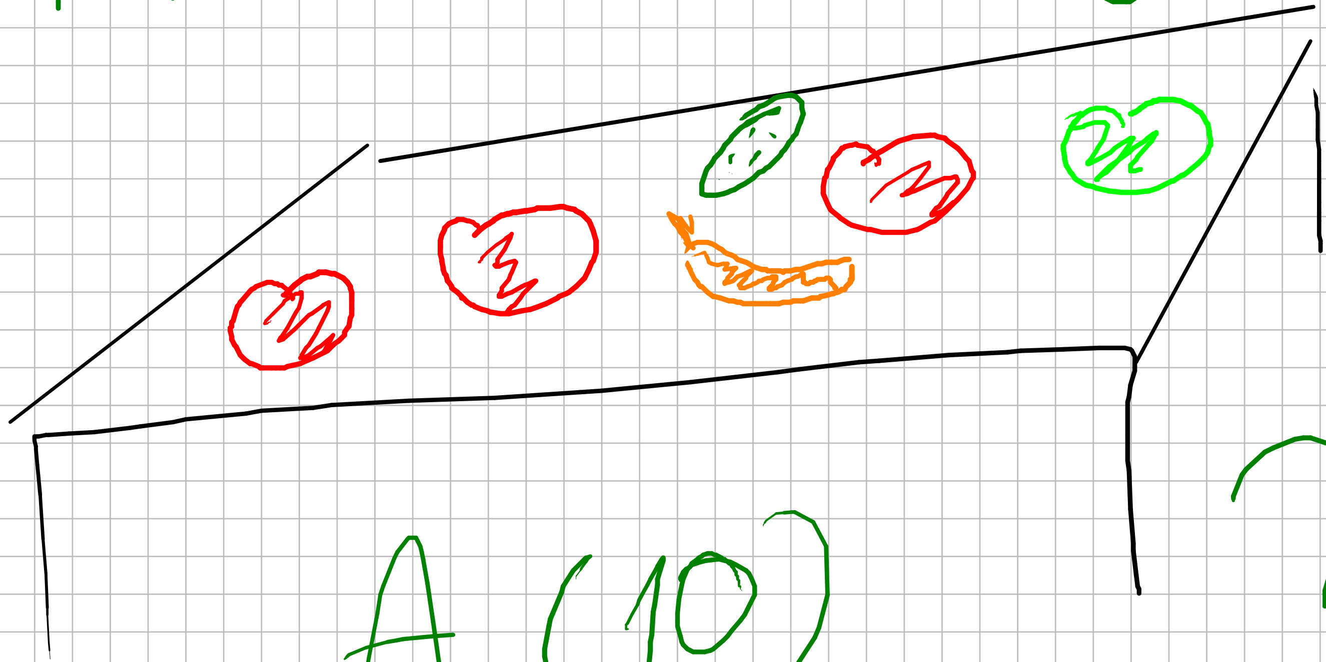
0	0	0	1	0	1	0	0

idx $j \to j-1$

$10100_2 = 20_{10}$

$101_2 = 5_{10}$

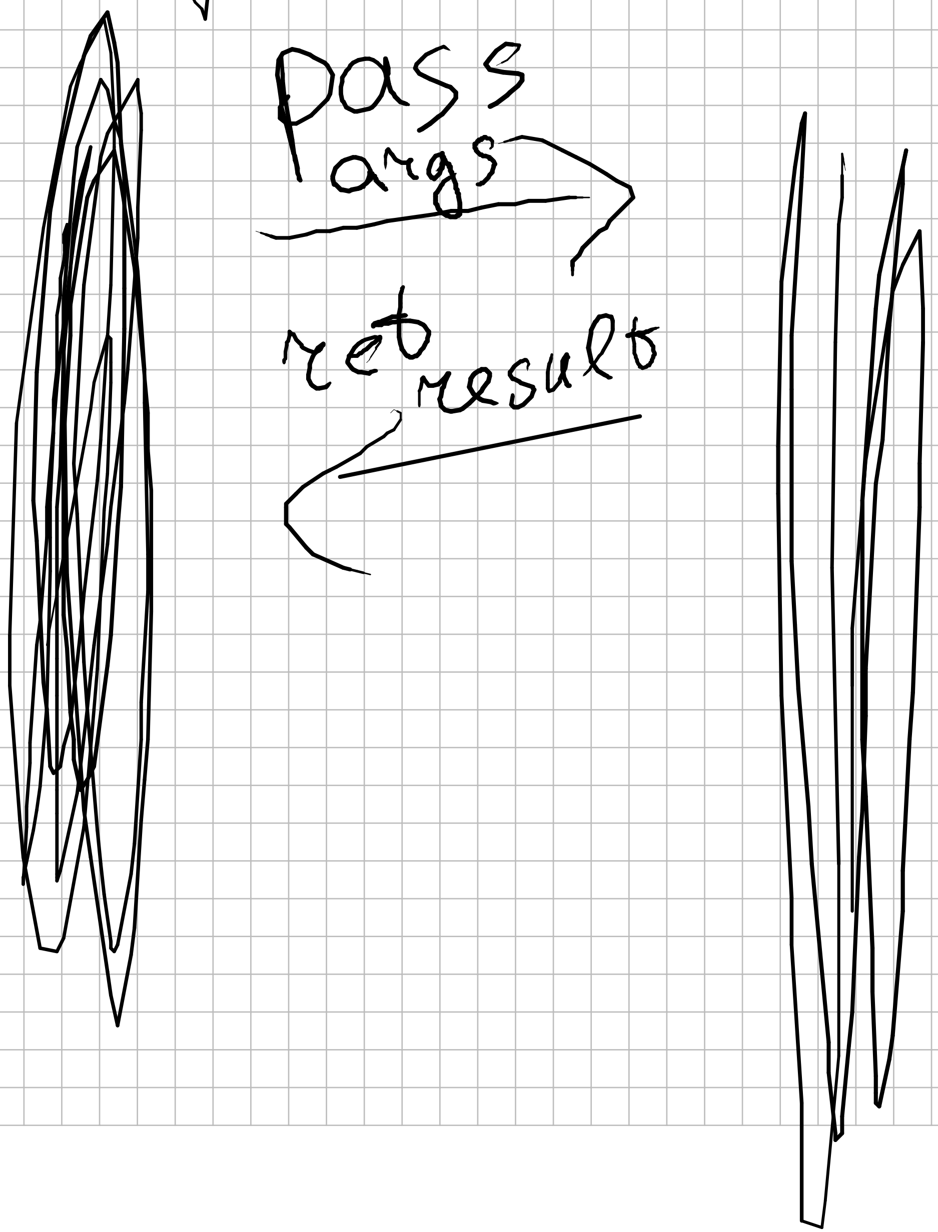
diversity



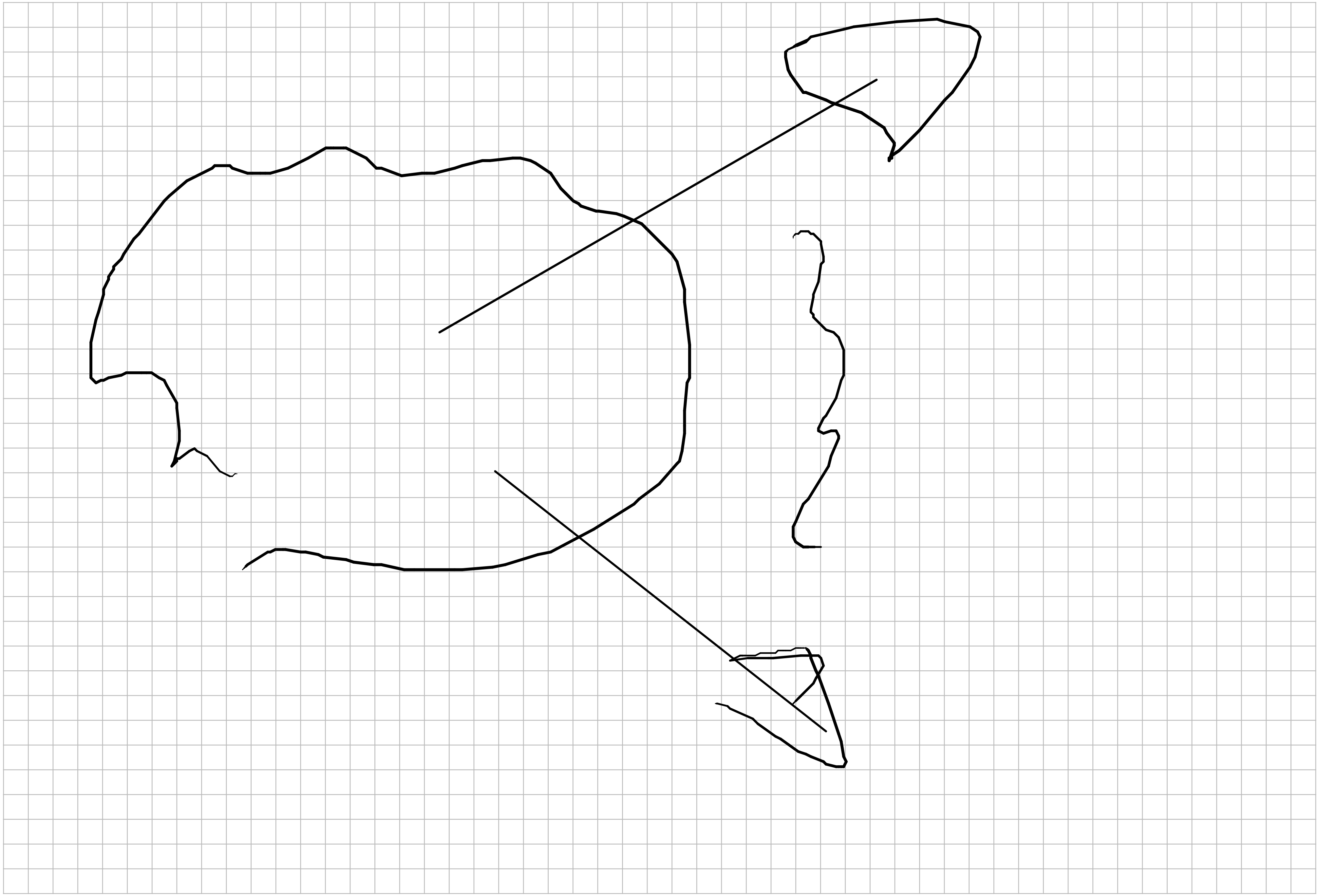
$D(10) +$

$A(10)$
$+ B(10) + C(30) + B(20) \cdot 5$

lifetime



comp



BFS nearest

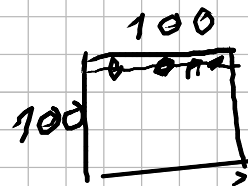
Контейнеры: set array

hash

$i \to i$ 96%

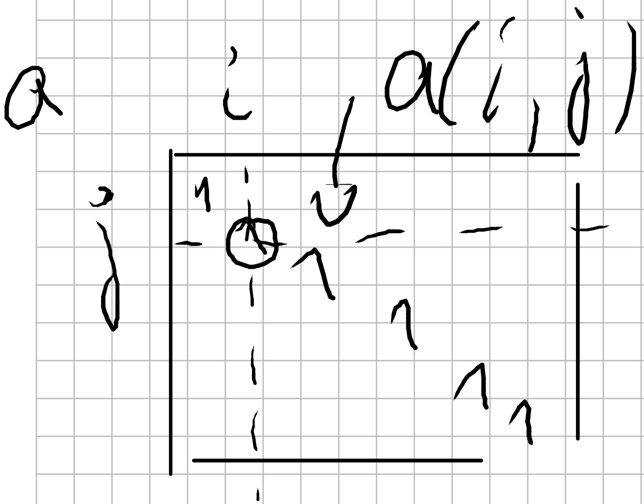
BFS $\longrightarrow$ Graph

$N = 100$



$i = 0$

$\sim 4/100$



$a \equiv$ adjacency matrix

$i \to j$

$a(i,j) = 1 \Leftrightarrow$

$a(j,i) \stackrel{?}{=} a(i,j)$?

$j \to i$

$i \not\to j$ 0

$a(i,i) \equiv 1$

$a(i,j) = \mathrm{cost}(i,j)$

m = Map< Point , List< Point > >

(x, y) -

{(-1, 0) , , , }

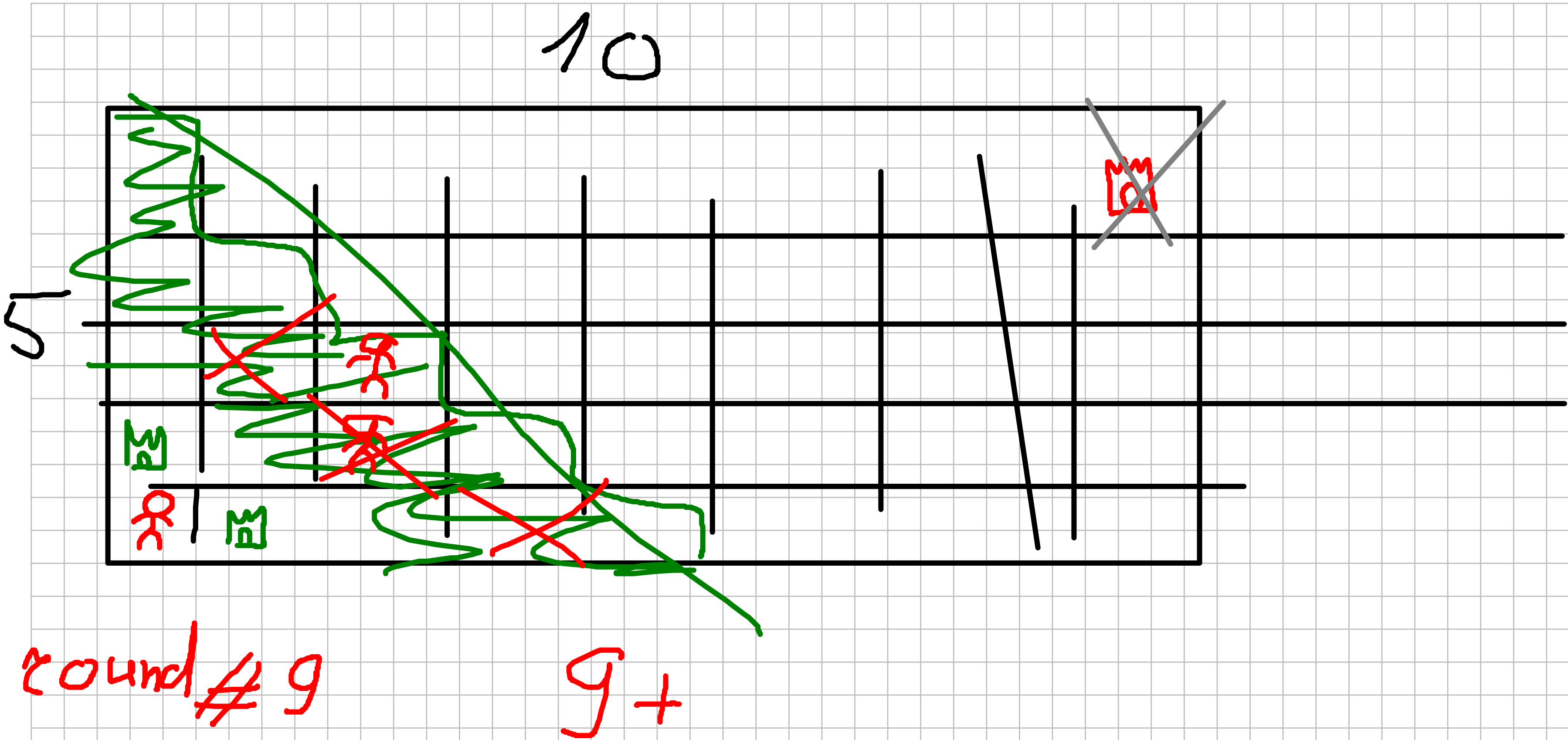
neighbours

root ⟶ queue

n = q.get()

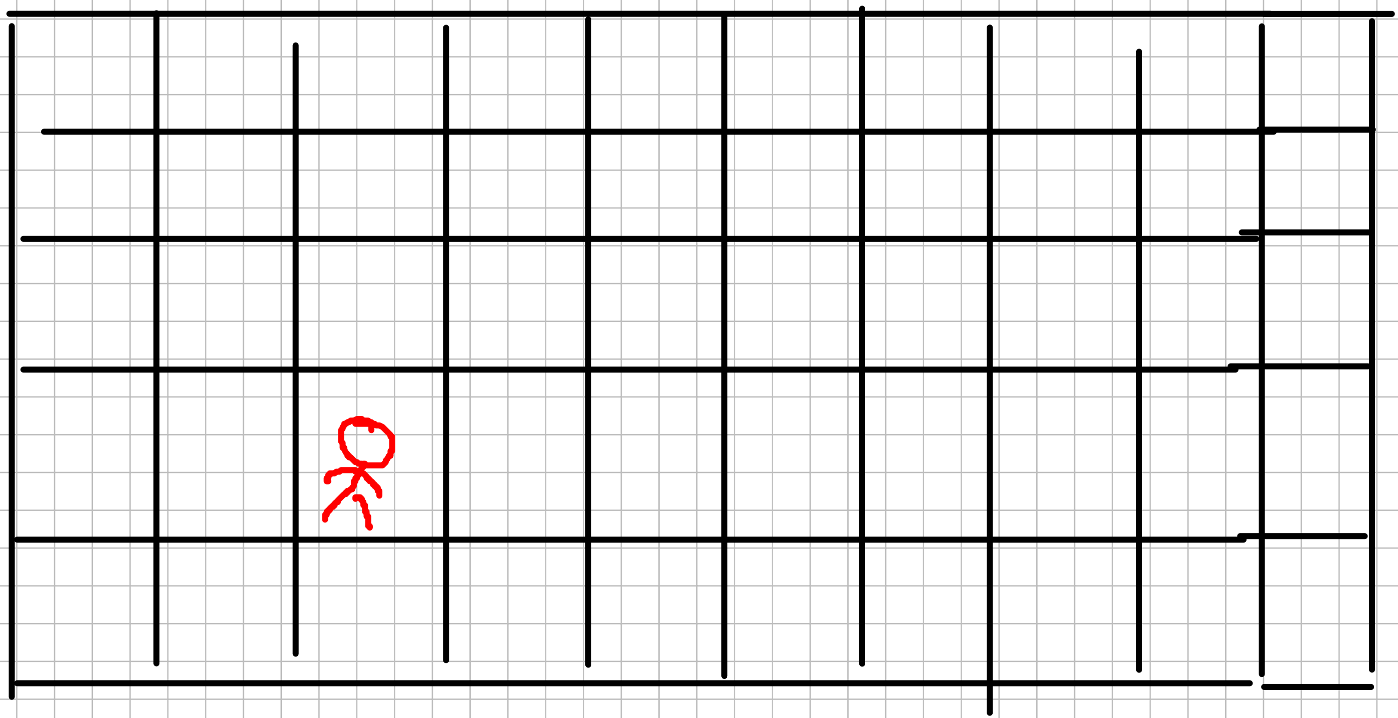
for nb in m(n):

q.put(nb)



we are looking at all map nodes that are no futher away
from pPos than distance=8

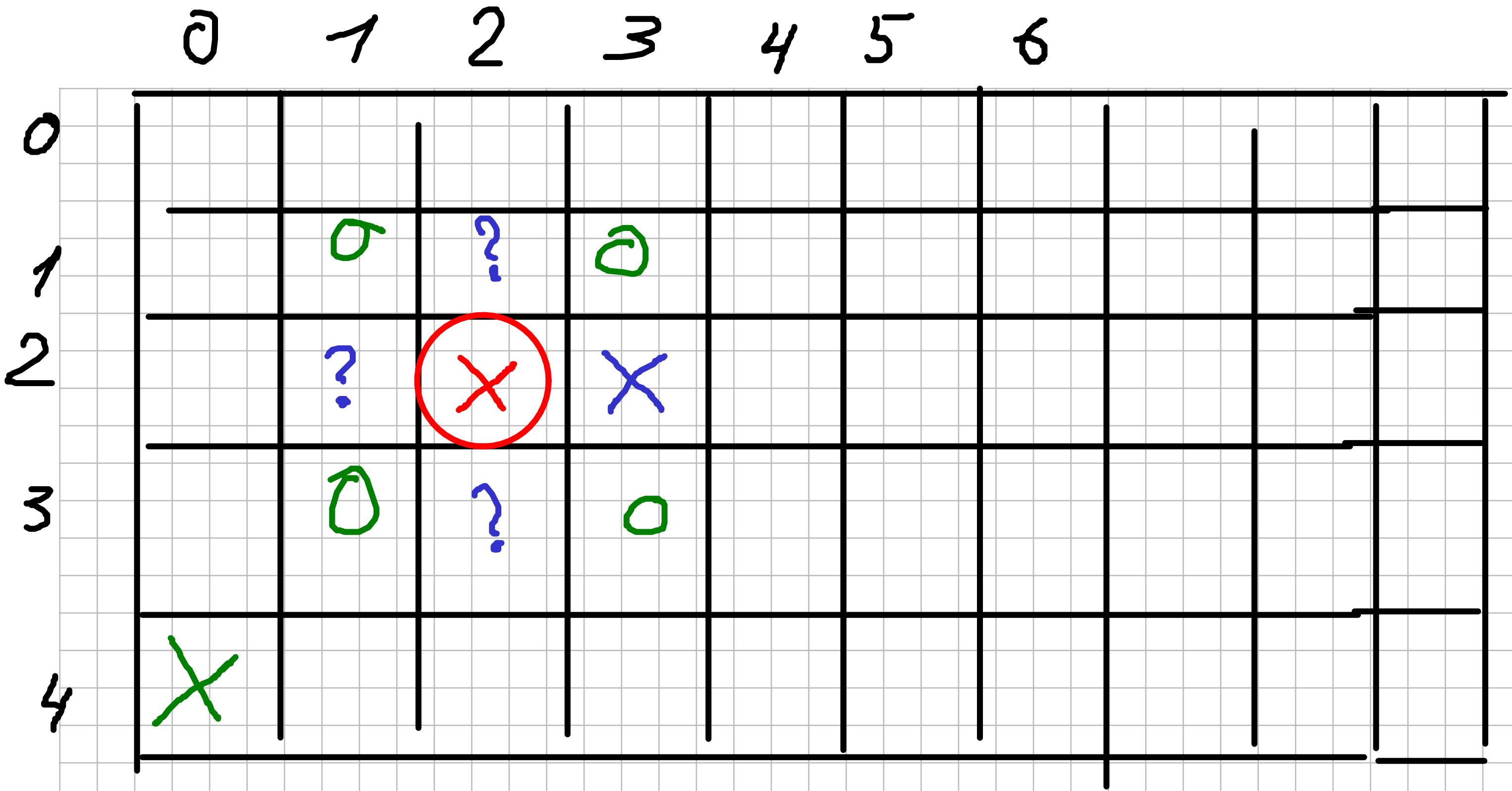
What is the reason behind randomizing enemy pPos during first N=8 rounds ?



pPosEnemy:
1: (2,3)
2: (2,3)
3: (2,4)
4: (2,4)
5: (1,4)
...

Q: where does enemy player start at the beginning of the game ?
A: at the location of their own castle !

Lets assume that server always sends us correct location of players.
Then pPosEnemy(t=1) gives coordinates of enemy castle.



zed - bl

2 2

3 2

$dx = -1$

$dy = 0$

≤ 2

0 4

$dx = 2$

$dy = -2$

zed - gz

1 0

0 -1

0 1

-1 0

$dx^2 + dy^2$

$= 2$

$\leq 2 \iff$

$= 1 +$

$= 2 \times$