py3\_sample.pyに関するメモ　byレーザー

　一応読み終わった．簡単にだけは理解できたと思いたい...

　コースの全格子点にコストを定義して，できるだけコストが低いルートを選んでいくようなプログラム．基本的に自分や障害物に近いほどコストは高く．また，コースの両端も少しコストが高くなるため，障害物やコースアウトを避けつつより遠くを目指せるような気がする．また，速度が増せば増すほどコストも重くなるので，スピードを出しすぎたりしないようになってるように思われる．なお，サンプル内では3手先まで探索している模様．

　おそらくではあるが，最初に視覚情報が与えられた時や，スピードが出てる時のコスト設定はうまくいかないと思う．コストの設定の仕方が視界の最奥のコストを設定した後，手前に向かって障害物がない場所だけコストを加算していくという方式をとっているため，障害物がなくても障害物がある場所よりコストが重くなることがあると思う．そのため，自分から壁に突っ込むことは割とありそう．

　あと，これも相手のことは何も考えていない．

　障害物がある部分のコストは一定っぽいから，そのコストだけ避けるようなプログラム追加したらより強くなるような気がする．

　動かしてないから何とも言えないけど割と強いような気はする．少なくともランダムよりは強いと思う．

　以下に出てきた関数を軽くまとめる．

　まとめるの下手すぎてちょっとわかりにくいと思う．ごめんなさい．

　あと，結局この手法の名前がわからないので題名変えてません．

　py3\_map.py についてはこっちに軽くまとめました

<https://docs.google.com/document/d/1_Uv9wNcjtlO1XFm7hSDiAWisRSb9D0oMBzsikHYJtHY/edit>

マップのコストの計算方法を間違ってました

上に書いてあることは全てスルーしてください（py3\_map.py については多分あってる）

消すのもあれ何で一応残しておきますが...

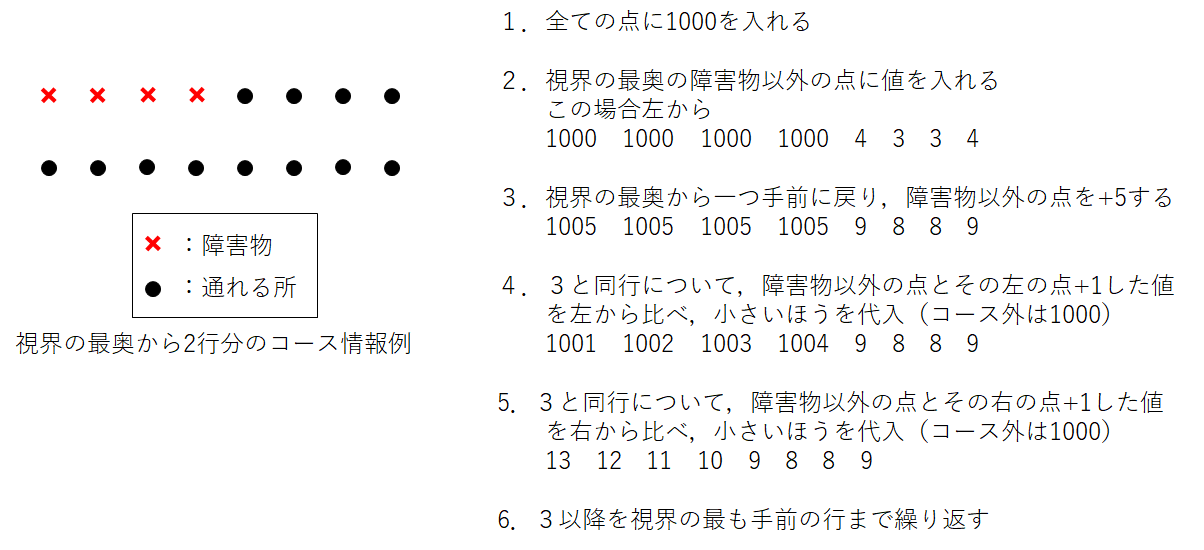
下に正しいコストの入れ方を載せます．図は視界の最奥から2行だけとりだしたものです

当然のことながらこの前集まったときはゴールできなかった理由を予想したけど全部間違いです

おそらくは python が重くて考慮時間が無くなったものと思われる

深さを１減らして動かしてみたら全コースでゴールできたから多分それで間違いない

ややこしいことをしてしまって申し訳ない



関数まとめ

1.make\_cost\_map

　マップの点ごとのコストを定義

　基本的に自分や障害物に近いほどコストは高くなる．また，コースの両端も少しコストが高くなる

2.readline

　入力して標準エラー出力に出力

　ランダムと全く同じ

3.next\_state

　次ステップの状態を出力

　ランダムと全く同じ

4.evaluate

　コストを (速度の和) \* 5 大きくしてる

　ブレーキ的な役割...?

5.search

　探索部．定めた深さまで探索して，最も良い値を返す

　探索中に視界外にでたらその場で探索を打ち切る

6.main

　a)考慮時間，制限ステップ数，コース幅，コース長，視界の広さ，入力して0を出力

　b)現在のステップ数，残り考慮時間を入力

　c）自分と相手の状態を入力

　d）視界内のコース情報を入力

　e)視界内のコース情報より点ごとのコストを決定

　f)入力できる加速度の組み合わせのうち，一番評価値の高いものを選択

　g)どの加速度の組み合わせを選択しても停滞する場合，加速度の範囲を（-現在の速度，現在の速度 + 1）まで増や　　し，その際の評価が高い加速度の符号より，入力する加速度を決定

　h) g)の加速度の範囲でもなお停滞する場合， (現在の速度) \* (-1) を加速度とする

　i）決定した加速度を指定された形に変換して出力

　j) b）以下をひたすら繰り返す