基礎工学PBL （情報工学B）

3班レポート

班員　：　山久保 孝亮, 浅海 雄士, 畑 大輝, 佐藤 慶季, 川口 晴大朗

班員学籍番号　：　09B22084, 09B22001, 09B22068, 09B22039, 09B22025

1　プロダクト

ここでは開発した蛇の内容を記述していく。

1.1　蛇の戦略

私たちが考えた蛇の戦略の概要は以下の通りである。

1. 予め盤面全体を通る一筆書きのルートを決めておいて繰り返し移動し続ける。
2. 次のターンで安全な方向を調べ、消去法で次のターンの行動を決定する。
3. 基本的には外周を回るようにする。
4. Health をできるだけ使い果たしてから、food を食べるようにする。

以降の章でこれらの戦略の詳細、またなぜその戦略に行き着いたかについて述べる。なおこれら以外にも様々な戦略を考え、細かな工夫を様々行ったが、ここでは主なものだけを記述する。

1.2　戦略の詳細

1. 最初にルートを決めておいてその経路を繰り返し移動し続ける。
   盤面が 7 × 7 のマスのため, プログラムのコードに図 1 のように、48 マス分の一筆書きができるルートを作成した。特に餌の有無に関して, 動作を変えるということはほとんどなかった。

   この後, ルール変更（場面上に餌が常に 3 つしか出現しない）があり, この戦略だと、盤面上にある餌を全て食べてしまうことになり、すぐに蛇の長さが最大に達してしまうので没となった。

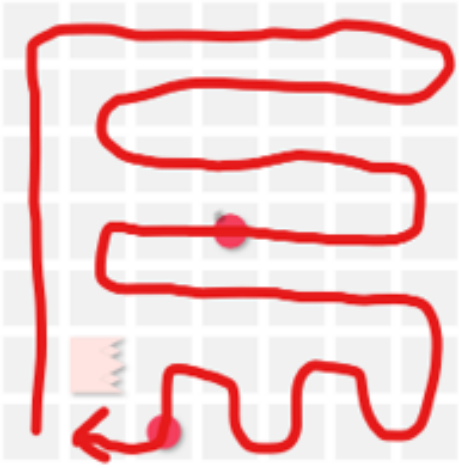

図 1: 蛇の通る経路

2. 次のターンで安全な方向を調べ、消去法で次のターンの行動を決定する。
   蛇が次のターンで力尽きる時、以下の図 2 の 2 パターンは容易に回避することができる。

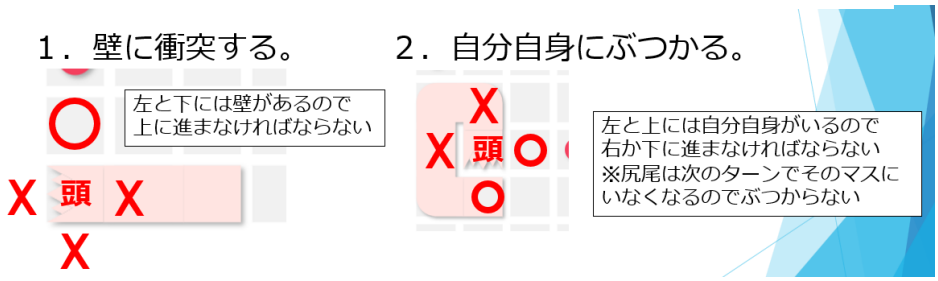

図 2: 次のターンで力尽きる 2 パターン

この 2 パターンで力尽きることは必ず避けなければならないので、他のどの戦略やアルゴリズムよりもこの結果を最優先した。安全な方向の選択肢が複数あるときは他の戦略・アルゴリズムによって絞り込むようにした。

3. 基本的には外周を回るようにする。
上で述べた、安全な方向の選択肢が複数ある場合は基本的に以下の図 3 のように外周を回り続けるようにした。

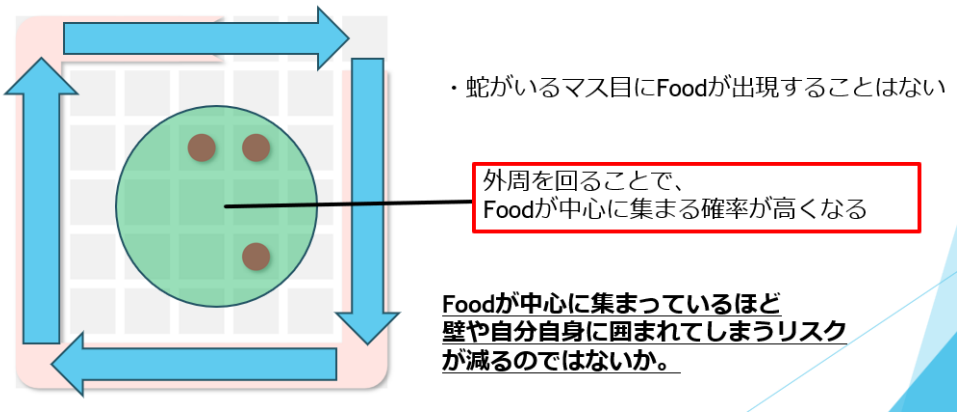

図 3:

battlesnake の仕様として、蛇が現在いるマスには餌が出現しないといったものがある。私達の班ではこの点に着目し、蛇の通る経路から餌の位置をある程度、制御できるのではないかと考えた。

餌を中心に集めることで、蛇の頭が壁と自分自身によって囲まれてしまうことが減るのではないかと考え、蛇が外周を回る事によって、餌が中心に集まる確率を高めることができるのではないかと考えた。

4. Health をできるだけ使い果たしてから、food を食べるようにする。
餌を取ることで蛇の長さが長くなる。また、今回のルールでは盤面の大きさは 7 × 7 と有限であり、また、蛇は長ければ長いほど蛇の体に衝突する確率が高まる。そのため、極力 health を使い果たしてから餌を食べるようにした。具体的な挙動としては、Health が十分あるときは、目の前に現れた food は避けるようにし、health が少なくなってから餌を取りに行くようにした。

1.3 戦略の利点と欠点

1. 予め盤面全体を通る一筆書きのルートを決めておいて繰り返し移動し続ける。
・メリット

ルール変更前において, とてもシンプルな方法でスコアを伸ばせるという点で, 良い戦略であった。
・デメリット
ルール変更（場面上に餌が常に 3 つしか出現しない）があり、この戦略だと、盤面上にある餌を全て食べてしまうことになり、すぐに蛇の長さが最大に達してしまう。

2. 次のターンで安全な方向を調べ、消去法で次のターンの行動を決定する。
・メリット
一ターン後に力尽きることを防ぐことができ、安全な方向が一つならそれ以上処理を行わなくて良い点。
・デメリット
これだけでは数ターン先まで予測することができない。（詳細については次の章で説明する。）

3. 基本的には外周を回るようにする。
・メリット
Food が中心に集まることで Food を取りに行きやすくなったり、自分自身と壁に囲まれる可能性が低くなる。進む向きが予め決まっており、処理時間が短く済む。また、health が少なくなるまで複雑な動作をする必要がないため、最序盤で力尽きることはなくなった。
・デメリット
蛇の長さが長くなってくると外周を回ることができず、決められたルートから外れざるを得なくなる点。

4. Health をできるだけ使い果たしてから、food を食べるようにする。
・メリット
餌を極力避けることから、蛇の長さがすぐには長くならないので、自分自身にぶつかりにくくなる。・デメリット
food が隅に集まってしまうと、取りに行くことが困難となる点。（詳細については次の章で説明する。）
餌を Health の値がなくなる直前まで避け続けることで、Health 不足で力尽きる危険性が高まった。

1.4 発生した問題点とその改善案

ここでは前の章で述べた問題点をどのように解決したか・もしくは改善できそうかについて記述する。

1. （戦略２）だけでは数ターン先まで予測することができない。
例えば以下の図 4 のようになったときに、蛇は次のターンで上下に進めることが (戦略２) によってわかる。しかし、上に進むと２ターン先に進めるますがなくなることがわかる。

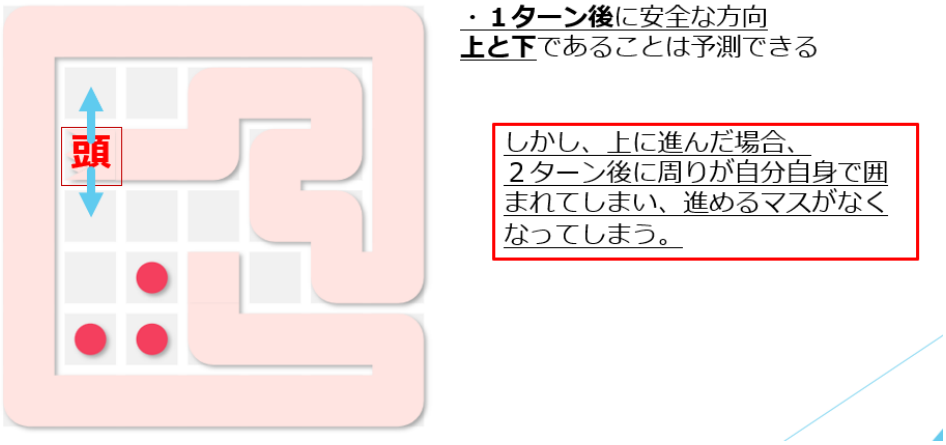

図 4: 数ターン先まで読む必要がある時

そこで、図 5 のように Flood fill 法を応用した順序付けによってこの問題を解決した。

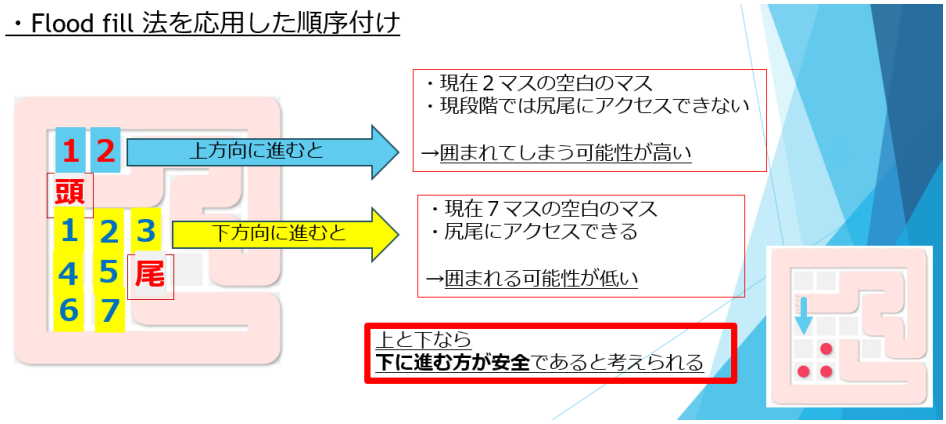

図 5: Flood fill 法を応用した方法による解決

2. food が隅に集まってしまうと、取りに行くことが困難となる点。
   例えば以下の図 6 のような餌の配置となってしまった場合、自分の体によって餌が囲まれてしまい、餌を取りに行くには数十ターンかかることが予想される。今現在の実装では、これが原因で餌不足によって力尽きることが多かった。

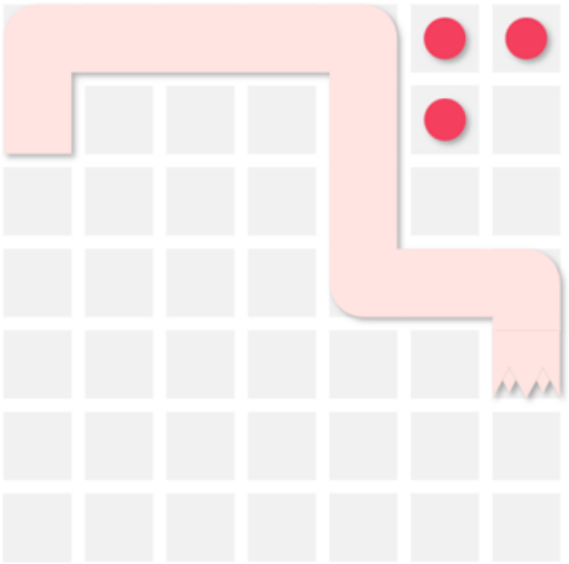

図 6: 餌が隅に集まってしまった場合

改善案として、Health が十分あるときは、外周を回るのではなく、盤面の中央あたりで自分の尻尾を追いかけるというものが挙がっていたが、時間の都合上、実装まで行えなかった。

2　プロセス

ここではグループワークの内容を記述していく。

2.1　作業分担

今回のグループワークを実施する上でのグループ内での作業内容と役割分担は以下のようになった。

- コード作成：佐藤, 畑
- レポート, プレゼン作成 (プロセス部分)：浅海, 川口, 山久保
- レポート, プレゼン作成 (プロダクト部分)：佐藤, 畑
- アルゴリズムの相談：全員
- デバッグ：山久保
- プレゼン発表：佐藤, 畑

以下でこの役割分担になった経緯を記述する。

1. 班員間のプログラミングの知識に偏りがあったためにコード作成班 2 人とそれ以外のレポート, プレゼン作成班 3 人に分かれた。
2. レポート, プレゼン作成はメンバー全員で行うように取り決めた。この理由としては, レポートとプレゼンは内容がほとんど同じであるため同時進行させることが可能と判断したためである。
3. また, レポートのプロダクトの部分はコードのアルゴリズムなどの知識が必要であるということからコード作成班が主に取り組むことにした。
4. 2。2 の議論の工夫で詳細は述べるが, アルゴリズムの相談は班員全員で行うことにした。
5. 蛇を実際に動かしてどのように死んでしまうのか, 比較的早い段階で死亡してしまう例はどのようなものなのかを知りたいが, 蛇を実際に動かすのに時間がかかってしまうというコード作成班からの声があったのでレポート, プレゼン作成班のメンバーで 10 回程度蛇を動かし, 死亡した原因とスコアを記述し共有することとした。
6. プレゼンの発表者 2 人は質疑応答の時間があるということからコード作成班が行うことにした。

続いて, 役割分担をした結果から考えた改善点を記述する。

1. コード作成班とレポート, プレゼン作成班の作業量の差が大きくなってしまった。原因としては, コード作成班は毎週の授業のために授業時間外に活動するのに対し, レポートやプレゼンはグループ活動の内容以外はコード作成班がある程度進めてからでないと取り組めなかったことが考えられる。
2. コード作成班は次回授業に向けてコードを作成するが, コードを書こうとしたときにまだデバッグができていないことが多かった。これはコード作成班とデバッグ班との間でどのタイミングでデバッグを終わらせておくべきかというコミュニケーションが取れていなかったことが原因であると考えられる。

具体的な改善策としては以下のようなものが挙げられる。以下の段落ごとの数字は上述した改善点に対応する数字である

1. コードが苦手な人も少しでも多くコードを書き、仕事の分散をする。
2. コード作成班とデバッグ班の都合を考えてあらかじめ何曜日までにデバッグをしておくかを話し合っておく。また, デバッグする人数を増やすことで予定が合わないという確率を減らす。

2.2 議論の工夫

今回私たちが議論をするうえで工夫した点は以下の通りである。

1. ホワイトボードを使って班員全員にアルゴリズムなどをわかりやすく共有し, 理解の度合いが全員でそろえられるようにした。
2. 全体でアルゴリズムについて意見を出し合う際、ホワイトボードや、パソコンで内容を洗い出し、全員が視覚的に内容を把握できるようにした。また, 班員全員の意見が反映されるように議長が一人一人に意見を振り、より密度と内容が濃い議論を行うことを意識した。
3. アルゴリズムの説明, プログラムのコードを作成していくうえでの知識は全員でズームに入り, パソコン上で同じ画面を視聴し, 全員で理解の度合いを高めた。
4. 班員間でプログラミングの知識に偏りがあったので授業で相談する際は具体的なコードでの議論でなく, 戦略を中心に議論を行った。

続いて、議論を行う上でさらに改善できた部分を以下に示す。

1. 班員の中でコードに関する理解度に差があり、議論の際にコードを書くのが苦手な班員に、コードの説明をする時間が発生し、議論が長引いてしまった。
2. 毎授業の議論で誰が議長の担当なのかわからなくなり、議論がスムーズに進んでいないことがあった。
3. この班は遅刻者、欠席者が多かったため、全体で話し合いを行い、議論を深める時間がすくなかった。

具体的な改善策としては以下のようなものが挙げられる。

1. 事前にコードを共有して班員が授業時に一度コードを理解したり, わからない点を各自メモしておいてコード作成班に説明してもらう時間を削減する。
2. 日誌に当日の議長や書記だけでなく, 次回授業時の議長や書記の名前も書いておく。

2.3 活用したツール

- battkesnake のローカル環境の構築
  各自でプログラム作成や、デバッグを行えるようにした。これにより、時間のかかるデバッグ作業を分担することが可能になり、また、各々でいつでも蛇の動きが確認できるので、改善案を考えやすくなった。
- CC
  周りの班の蛇の動きを参考にして、既存の戦略と融合させることで、よりよい戦略を生み出すことができた。
- GitHub
  プログラムコードの共有だけでなく、バグが発生した際にその原因の特定にも使用した。
- Overleaf(レポート作成) レポートは Overleaf をとおして、班員全員に共有され、各々が担当の箇所を中心に分担して作成した。
- PowerPoint
  スライドは PowerPoint をとおして、班員全員に共有され、各々が担当の箇所を中心に分担して作成した。