



NPCA 部誌

2025

目次

第 1 章	電動キックボードのポート配置を最適化してみた	3
第 2 章	ポケポケの戦略をプログラミングで導く	8
第 3 章	日本語で書かれる PG 言語 なでしこ	14
第 4 章	MythicMobs でオリジナルモブを作ろう	18
第 5 章	Python を使った自作 SNS の開発	25
第 6 章	サイクリングの周回ルートを探索する	40

あいさつ

皆さんこんにちは。灘校パソコン研究部部長の Kohenyan と申します。本日は第 79 回灘校文化祭 “weave” にご来場いただきありがとうございます。灘校パソコン研究部では、競技プログラミング^{*1}や機械学習、Unity^{*2}などによるゲーム作成、CTF^{*3}やセキュリティ、サーバー構築など、情報系の様々な分野に日々取り組んでいます。一口にパソコンと言えども多様な「好き」を持った部員たちの集大成である展示と部誌を、ぜひ楽しんで行ってください！

この冊子について

部誌のコードやファイルは、灘校パソコン研究部の公式サイトにて公開しています。少しでもご興味を持っていただけたら、公式サイト内の「作品」ページをご覧ください。か、「NPCA 作品」で検索してください。

執筆者の表記について

執筆者名として、本名ではなくハンドルネームが記載されています。これは、部内では本名を呼ぶことが滅多になく、ほとんどの場合部内ではハンドルネームで互いを呼び合うという慣習によるものです。また、灘校の文化に従い、学年ではなく何回生かが記されています。2025 年 5 月現在、各回生は次の学年に相当します。

- 79 回生：高校 2 年生
- 80 回生：高校 1 年生
- 81 回生：中学 3 年生

^{*1} 与えられた課題を解決するプログラムをいかに素早く正確に記述するかを競うプログラミングのコンテストの総称

^{*2} ゲームを作るためのアプリケーション（ゲームエンジン）の一種

^{*3} Capture The Flag（旗取りゲーム）の略、情報セキュリティの技術を競う競技

第1章

電動キックボードのポート配置を最適化してみた

79 回生 Kohenyan

1.1 はじめに

こんにちは！パソコン研究部の部長をしている、高2のKohenyanと申します。部長になってみて、思った以上にやる事が多くて大変だなと感じる毎日です。

先に説明しておく、「情報科学の達人」というプロジェクトは、高校生や高専生が最先端の研究者と一緒に学びながら、自分たちで研究テーマに挑戦するというプログラムです。

今回はこの「情報科学の達人」という教育プロジェクトに参加して、その中で取り組んだ研究内容を記事にまとめました。このプロジェクトでは、東京大学の河瀬先生（特任准教授）がメンターとしてサポートしてくださいました。河瀬先生には、研究を進めるうえでたくさんアドバイスをいただき、本当に感謝しています。改めてお礼を申し上げます。

1.2 電動キックボードの研究をやろうと思ったきっかけは？

電動キックボードって少し前に結構話題になりましたよね。

それで、キックボードのポートってどれぐらいあるんだろう？ということで、学校の近く、三宮駅周辺のポートを調べてみました。

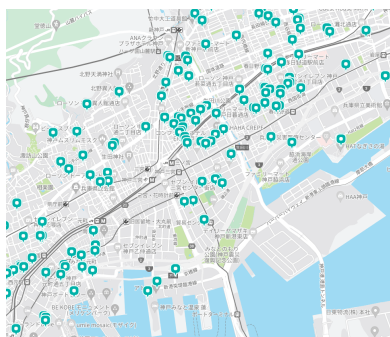


図 1.1: ポート配置の図

上の図を見て、ポートの配置偏ってるのでは？と思い、実際の需要とどれくらいずれているか、そして再配置を行うことでどのように改善するか調べることにしました。

1.3 前提知識

電動キックボードについて：電動キックボードはポートとポートの間を移動できる交通手段です。今回は LUUP という電動キックボードについて調べました。

線形計画問題とは：線形計画問題は、線形な目的関数を最大化または最小化し、線形な制約条件を満たす解を求める問題です。変数は連続値（実数）を取ることができ、特定の範囲内で（目的関数が最大化 or 最小化するような）最適な解を探します。

簡単な例) 目的関数: $-2x + 3y$ ←最小化

制約条件: $x + y \leq 5, x, y \geq 0$

このような問題を、ソルバー PuLP というものを使えば最小化される (x, y) を求めることができます。実際は、この x や y の変数が数十個、数百個もある問題を解きます。

1.4 実験の手法

神戸の三宮駅、神戸駅周辺をグリッド状に分割することで、プログラム上にリアルの情報を表現しました。また、ポート配置を最適化する問題を、のちのページのような線形計画問題として定式化し、ソルバー PuLP によって求解しました。

ポートを利用する人口は、特定の時間帯における人の分布を示す「人口流動マップ」（NTT ドコモ提供）を活用して推計しました。具体的には、平日 13 時の 500m グリッドごとの人口データに対し、電動キックボードを利用する割合（= 0.01 程度の定数）を乗じることで、各グリッドから駅へと移動する利用者数、つまり需要を算出しました。

（問題設定:2 つの駅があり、各利用者は最寄りの駅を利用します。キックボードを利用すると移動速度が徒歩の 2 倍となり、利用者全体の平均移動時間が最小となるように利用されるものとします。）

線形計画問題の定式化

注釈: パラメータ、関数、変数、目的関数、制約条件をのせますが、少し難易度高めなので、分からない場合は実験結果まで飛ばしてください。

パラメータ

- M : グリッド数
- pop_i : i 番目のグリッドの人口 ($0 \leq i < M$)
- $pos_{i,j}$: i 番目のグリッドの座標 ($i = M$ は駅 A, $i = M + 1$ は駅 B, $j = 0$ が x 座標, $j = 1$ が y 座標)
- $portprev_i$: i 番目のグリッドの現在のポート数 ($0 \leq i < M$)

関数

距離関数：

$$d(i, j) = \sqrt{(pos_{i,0} - pos_{j,0})^2 + (pos_{i,1} - pos_{j,1})^2}$$

変数

- $port_i$: i 番目のグリッドに設置するポート数 ($0 \leq i < M$)
- $v_{i,j}$: i 番目のグリッドが利用する移動手段の人数 ($0 \leq i < M$, $0 \leq j \leq M$, $j = M$ の場合は徒歩移動)

目的関数

$$\min \left\{ \begin{array}{l} \sum_{i=0}^M \sum_{j=0}^{M-1} \left(\underbrace{d(i, j) \times 2}_{\text{徒歩でグリッドからポート}} + \underbrace{\min(d(j, M), d(j, M+1))}_{\text{キックボードでポートから最寄駅}} \right) \times v_{i,j} \\ + \sum_{i=0}^{M-1} \left(\underbrace{\min(d(i, M), d(i, M+1)) \times 2}_{\text{徒歩でグリッドから最寄駅}} \right) \times v_{i,M} \end{array} \right\}$$

制約条件

1. ポート容量制約：

$$\sum_{j=0}^{M-1} v_{j,i} \leq port_i \quad (0 \leq i < M)$$

2. 人口分配制約：

$$\sum_{j=0}^M v_{i,j} = pop_i \quad (0 \leq i < M)$$

3. ポート増設上限 (X はポートを追加する数)：

$$\sum_{i=0}^{M-1} (port_i - port_{prev_i}) \leq X$$

4. ポート非減少制約 (再配置実験の時は除く)：

$$port_i \geq port_{prev_i} \quad (0 \leq i < M)$$

5. 非負制約：

$$v_{i,j} \geq 0, port_i \geq 0 \quad (0 \leq i < M, 0 \leq j \leq M)$$

1.5 行う実験

まず、実際のポートの配置と需要がどれぐらいマッチしているか調べるために、ポートの総数を変えずに (実際のポートの配置をすべて破棄した状態で) 再配置する実験を行いました。次に、ポートを追加したらど

れぐらい改善するのか確認するために、実際のポートの配置を維持したままポート数を増やしていく実験を行いました。

1.6 実験結果

1. ポートの総数を変えずに再配置した場合の改善効果: 平均移動時間の値が、再配置前では 6.9067 であったのに対して、再配置後は 4.5964 となり、再配置により 34% の改善が確認できました。
2. ポートを追加した場合の改善効果: 下図のように、ポートの追加に応じて平均移動時間が改善しました。

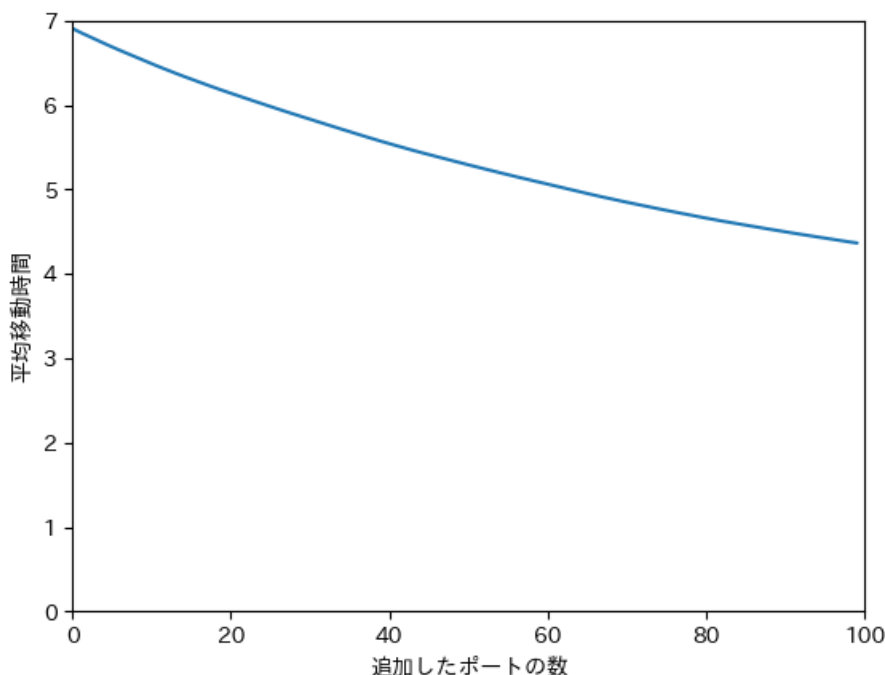


図 1.2: ポート追加による移動時間の変化

実験結果から分かること: しかし、元々のポートは 114 個であり、約 90 個の追加によって再配置と同等の改善効果が得られたことが分かります。従って、単純にポートを増設するよりも、適切に再配置の方が効率的な改善策であることが分かりました。

1.7 まとめ

実験結果に書かれている通り、ポートを増設するよりも適切に再配置の方が効率的であるというのは、言い換えると、既存のポートが需要とかなり乖離された状態で配置されている、ということですね。今回は、ポート数が 100 個程度、駅も二つだけの神戸、三ノ宮駅周辺で実験しましたが、ポート数もっと多く、駅も密集している東京 23 区全体で実験してみたり、ポート配置による設置費用と、設置すると需要によってどれだけ利益が増えるか、も実験してみたいと思いました。

1.8 参考資料

- <https://note.com/shirotabistudy/n/n7fc4ba351eea> 「Python pulp メモ」

- <https://luup.sc/port-map/> 「LUUP ポート一覧」
- <https://mobakumap.jp/#34.688958,135.187581,14z> 「人口流動統計 - モバイル空間統計」

第2章

ポケポケの戦略をプログラミングで導く

79 回生 かつどん

2.1 はじめに

こんにちは。79 回生のかつどんです。好きなポケモンはラティアスです。僕の灘校文化祭も残り 2 回となくなってしまいました。去年はなんと準備期間から文化祭が終わるまで熱を出して寝ていました。最後の文化祭でそうならないことを祈るばかりです。

さて皆さん、Pokémon Trading Card Game Pocket 通称「ポケポケ」プレイされているでしょうか？ポケポケはポケモンカードゲームを原作としたスマホ向けのカードゲームアプリで、手軽にカードを集めたり対戦したりでき、去年の 10 月にリリースされて以来大人気のゲームになっています。この記事ではそんなポケポケの「対戦」について深く考えてみます。

皆さんポケポケをプレイしていてこんなことを思ったことはありませんか？

- 進化ポケモンを立てたいけどなかなか手札に来ない。
- 相手のカスミで運だけで負けた。
- 手札のカードどれから使ったら一番勝ちに近づくの？

やはりカードゲームなので当然確率が絡みますが、なかなか釈然としないことがあるかもしれません。今回は「進化ポケモンが立つ確率」に焦点を当てて確率をプログラミングで求め、戦略を考えてみます。この記事を読んでいるころにはランクマッチが開幕していると思うので、これを参考にプログラミングを駆使して爆勝ちしてみてください。

2.2 ポケポケをプレイしていない人へ

ここではポケポケの対戦機能について軽く説明します。既にプレイされている方は飛ばして大丈夫です。

ポケポケでは 20 枚のカードを使って 1 対 1 で対戦します。カードには大きく分けて「ポケモン」と「トレーナーズ」の 2 種類のカードがあります。対戦の最初に先攻・後攻を決めて山札を 5 枚引きます。それぞれの番のはじめに山札からカードを 1 枚引き、手札のカードを使って戦います。

ポケモンのカードは、実際にワザを使って戦うカードです。

ポケモンは「バトル場」と「ベンチ」という 2 種類の場所に出すことができ、自分のターンにバトル場にいるポケモンがワザを使って相手のポケモンに攻撃することができます。ワザを使うと自分の番が終わりま



す。ワザなどでダメージを受けて HP が 0 になるとそのポケモンはきぜつします。相手の番にバトル場のポケモンが相手に倒されたときはベンチからポケモンを出します。ポケモンは進化していないポケモンであるたねポケモンとその進化先のポケモンをデッキに入れておくことで進化させることもできます。ポケモンの中にも「ポケモン ex」という特別なポケモンがいて、相手のポケモン ex を倒すと 2 ポイント、そうでないポケモンを倒すと 1 ポイント獲得でき、3 ポイント取るとバトルに勝利となります。

トレーナーズのカードは戦うのではなく、自分のターン中に使用して山札を引いたりポケモンの HP を回復したり対戦をサポートしてくれるカードです。



トレーナーズのカードには「グッズ」「サポート」「どうぐ」の 3 種類があります。グッズは自分のターンに何回も使用できるカードです。サポートは自分のターンに 1 回しか使えません。どうぐは他とは違い、自分の場のポケモンに付けて効果を発揮します。

正直プレイしてみないとわからない部分が多いと思うのでこの記事を読んで興味を持ったら実際にプレイしてみてください。

2.3 今回使うデッキや手法など

今回は執筆時点で環境デッキの一角である「ダークライ ex+ ジバコイルデッキ」を用いて「最速で 2 進化ポケモンのジバコイルが立つ確率」を考えていきます。その番場に出したばかりのたねポケモンやその番進化させたばかりのポケモンを進化させることはできないため、2 進化が立つ最も早いターンは 3 ターン目です。前半では 1 つめのデッキ、後半では 2 つめのデッキについて考えます。どちらも海外大会の上位入賞デッキで強さはお墨付きです。



採用されているカードについて軽く説明します。

- コイル・レアコイル・ジバコイル 今回計算するテーマのポケモンです。自分で自分にエネルギーを付けられて優秀なアタッカーです。コイル→レアコイル→ジバコイルと進化します。
- ダークライ・ガルーラ どちらもジバコイルと相性が良いたねポケモンです。
- モンスターボール 山札からランダムにたねポケモンを1枚手札に加えるグッズです。
- 博士の研究 山札を2枚引けるサポートです。
- ポケモン通信 2つめのデッキにのみ採用されています。手札のポケモンを1枚選んで山札のランダムなポケモンと入れかえるグッズです。

- スピーダー 1 つめのデッキにのみ採用されています。使用したターンポケモンのにげるために必要なエネルギーが 1 減るグッズです。
- リーフ 使用したターンポケモンのにげるために必要なエネルギーが 2 減るサポートです。
- ナツメ・アカギ どちらも相手のベンチのポケモンをバトル場に引っ張り出すサポートです。
- ゴツゴツメット 持たせたポケモンがワザのダメージを受けたとき相手に 20 ダメージ与えるどうぐです。
- 大きなマント 持たせたポケモンの HP が 20 増えるどうぐです。

スピーダー以降のカードは今回調べる 2 進化が立つ確率には関係がないカードですが、どれも強力なカードです。

ところでみなさん、高校で確率を勉強するときどうやって求めていたでしょうか？全事象のうち〇〇が起きる場合が何通りあるから何分の何、という風に求めていたりしたと思うのですが、2 進化が立つ確率というのは全事象が多すぎて到底正確に求めることはできません。そこで、コンピュータの力を借りて「実際に何万回と試してみても何回成功したか」で確率を求めていきます。プログラミングの世界ではモンテカルロ法とか言われています。2,3 回試してみただけでは正確な確率は求められませんが、プログラミングを用いて大量に試行することでかなり正確に求めることができます。

2.4 実際にプログラムを書いて計算してみよう！

最初は 1 つめのポケモン通信が入ってないデッキで 3 ターン目にジバコイルが立つ確率を計算していきます。ちなみに今回、レッドカードなどのカードで相手からこちらの手札に干渉してくるカードについては考えないものとします。そんなもの考えてられないので。最初は順番を特に考えずにカードを使っていきます。具体的には全て博士の研究→モンスターボールの順番でプレイしています。後でカードを使う順番をよく考えると確率がどのように変わるかお見せしようと思います。

今回は僕が普段使い慣れている C++ を使って計算していきます。紙面上ではスペースを使いすぎてしまうためコードは紙版では割愛します。web 版では実際に使ったコードを載せているのでそちらも是非お読みください。ちなみに実験のために人に見せることとか考えずに適当に書き殴ってたら 400 行とかになってました。ひどすぎる。web に載せるコードはそういうきったねえコードじゃなくてきれいにする予定です。

```

56     if(hand['G'] >= 1){
57         for(int i = 0; i < 2; i++){
58             hand[deck[0].first]++;
59             deck.erase(deck.begin());
60         }
61         hand['G']--;
62     }
63     while(hand['F'] > 0){
64         bool flag = true;
65         for(auto x : deck){
66             if(x.first == 'A' || x.first == 'D' || x.first == 'E'){
67                 flag = false;
68                 break;
69             }
70         }
71         if(flag) break;
72         while(deck[0].first != 'A' && deck[0].first != 'D' && deck[0].first != 'E'){
73             random_device get_rand_dev;
74             mt19937 get_rand_mt(get_rand_dev());
75             shuffle(deck.begin(), deck.end(), get_rand_mt);
76         }
77         hand[deck[0].first]++;
78         deck.erase(deck.begin());
79         hand['F']--;

```

こんな感じです。ちなみにこれは博士の研究とモンスターボールを使用するときのコード。

実際にカードを引いたりポケモンを持ってきたりのシミュレーションを 100000 回行うコードを書いて実行してみると...36505 回成功、だいたい 36.5 %の確率で 3 ターン目にジバコイルが立つことがわかりました！だいたい 3 回に 1 回といったところです。みなさんがプレイする時の感覚とは合っているでしょうか？僕的には少し感覚より低いかも？という感じです。というわけでここからはポケモン通信を入れたり順番を変えたりして確率がどれくらいあがるか調べていきましょう！

2.5 ポケモン通信を入れて計算してみる

次はポケモン通信を入れたデッキで計算してみます。順番は深く考えずに博士→モンスターボール→ポケモン通信でプレイします。ちなみにポケモン通信は手札に 2 枚来たポケモンがあるときにだけ使用しています。さっきと同じように 100000 回シミュレーションしてみると...？

結果は 43004 回成功、約 43 %で 3 ターン目にジバコイルが立つという結果になりました。なんとカードを 1 枚変えるだけで 6.5 %も確率が上がるのです。ポケモン通信というカードの強さがお分かりいただけただけでしょうか？普段なんとなくデッキに入れているカードでもこうやって実際に大量に実験してみるとその強さが数値としてわかるのです。（ちなみにポケモン通信の代わりにデッキから抜けたスピーダーも強力なので 2 進化が立つ確率が上がったからといって一概にデッキが強くなったとは言えないことに注意してください(笑))

2.6 順番を考えて計算してみる

最後はカードを使う順番を変えてみます。こんな順番にします。

コイルがないとき



コイルがいるとき



以下の2点において変えています。

まずは博士の研究とモンスターボールの使用順です。1ターン目にまだコイルを持っていないときは博士→ボール、それ以外の場合はボール→博士→（博士でボールを引いたら）ボール という順番にします。なぜこうするのかというと、コイルが欲しい場合はコイルを引く確率を少しでも上げたいため、先に2枚引いてそこで他のたねポケモンを引いたら後でコイルを持ってこれる確率を上げるので先に博士を使う方がよく、逆にコイルを引いている場合は2進化が立つ確率を上げるためにたねポケモンはいらないため先にモンスターボールを使用してたねポケモンを山札から抜くことで欲しいカードを引ける確率を上げた方がいい、ということです。

2つ目はポケモン通信についてです。ここまでは手札にあったら番の終わりに必ず使用していましたが、そのターン欲しいポケモン（1ターン目であればコイル、2ターン目ならレアコイル、3ターン目ならジバコイル）を引いていないときに限り番の終わりに使用するように変えます。まだ引けていない進化先を引く確率を上げるためにはターンが過ぎてたくさん山札を引いてからポケモン通信を使った方が良いからです。

この2点について変更した100000回シミュレーションすると... 結果は44112回成功、約44.1%で3ターン目にジバコイルが立つという結果になりました。先ほどのような大きな変化はありませんがそれでも確率は上昇しました。ポケポケは運要素の強いゲームではありますが、こうしたデッキ構築やカードの使用順によって勝つ確率を上げることができるのです。

2.7 おわりに

いかがでしたでしょうか？プログラミングの便利さとポケポケの奥深さを理解していただけただけなら幸いです。

ポケポケには他にもいろいろ確率を計算してみたいことがあると思うので、そんなときにこの記事参考にシミュレーションして、どんな風にデッキを組めば安定性が高く強いデッキが組めるか、どうプレイすれば勝てるようになるか、いろんなデッキについて計算してみてください。もちろん、ポケポケ以外のことも計算してみてくださいね。ちなみに僕は紙のポケモンカードもやっているのですが、60枚のデッキでこれをしようとして挫折しました(笑)。

ここまでお読みいただきありがとうございました。引き続き灘校文化祭「weave」をお楽しみください！

第 3 章

日本語で書かれる PG 言語 なでしこ

81 回生 fujimon

3.1 はじめに

はじめまして、fujimon と申します。今回は日本語で書かれるプログラミング言語を見つけたのでどんなものか、そして構文がどんなものかなどを見ていこうと思います。

3.2 なでしことは

「なでしこ」は山本 峰章さんが日本語で誰でも気軽にプログラムできるようにしたいという思いでつくった言語です。特徴としては日本語で書かれている、が大きなものだと思います。ほかの言語 (c++,Python,javascript など) は大体英語で書かれます。プログラミング言語は世界中で使われるので英語で書かれるほうが良いです。ちなみにほかの日本語で書かれるプログラミング言語は「ひまわり」や「ドリトル」、「プロデル」など 7 種類以上あるみたいです。

3.3 「なでしこ」の構文など

それでは「なでしこ」の構文などを見ていきましょう。「なでしこ」の構文は公式 HP に載っていました。

→→→ <https://nadesi.com/>

まず、出力は ～～を表示。 とするとできるみたいです。

次に四則演算の仕方を見てみましょう。四則演算は例えば $3 + 5$ なら 3 に 5 を足して表示。

3に5を足して表示。

▶ 実行

クリア

保存

1行目 v3.7.3

8

とするとできるみたいです。

また、敬語の機能も実装されており、さっきのコードを敬語化して

3に5を足して表示する。

3に5を足して表示してください。

3に5を足して表示してください。お願いします。

3に5を足して表示する。
3に5を足して表示してください。
3に5を足して表示してください。お願いします。

▶ 実行

クリア

保存

1行目 v3.7.3

8
8
8

のようにしても $3 + 5$ を実行できます。

さらに拝啓、敬具とかいて手紙のようにすることもできるみたいです。ちなみに「敬具」には礼節ポイントを初期化（0にすること）して、「ください」、「お願いします」などの数をカウントし、礼節ポイントを表示させることもできるみたいです。謎使用ですね。

```

1  拝啓。
2  挨拶は「こんにちは」です。
3  挨拶を表示してください。お願いします。
4  礼節レベル取得して表示。
5

```

▶ 実行 クリア v3.7.3

こんにちは
2

上では1行目で礼節ポイントを0にし、3行目に「ください」、「お願いします」をつかっているので礼節ポイントが2になっています。

3.4 問題を解いてみる

それでは問題を解いてみましょう。

今回は AtCoder Beginner Contest 397 の A 問題を解きます。

https://atcoder.jp/contests/abc397/tasks/abc397_a

問題文

高橋君は自分の体温を計測し、 X °C であることがわかりました。

体温は次の 3 種類に分類されます。

- 38.0 °C 以上のとき「高熱」
- 37.5 °C 以上 38.0 °C 未満のとき「発熱」
- 37.5 °C 未満のとき「平熱」

高橋君の体温は何に分類されるのでしょうか？「出力」の項にしたがって整数で出力してください。

※高熱の時 1、発熱の時 2、平熱の時 3 を出力

これはただ単に X に応じて場合分けするだけです。

なでしこ (cnako3 3.4.20) ▼

```

1  入力を探ねてXに代入。
2  もし、Xが38.0以上ならば
3  |   「1」と表示。
4  違えばもし、Xが37.5以上ならば
5  |   「2」と表示。
6  違えば
7  |   「3」と表示。
8  ここまで。

```

解説をしておくと、1行目ではXという変数をつくり、ここに与えられた数字を代入しています。2, 3行目ではXが38.0以上の時の場合分けです。4, 5行目ではXが37.5以上38.0未満の時、5, 6行目ではXが37.5未満の時の場合分けです。ちなみに最後の行の「ここまで」はよくわかりません。ほかの人も書いていなかったらエラーがおきました。まあ普通にコードの最後を意味しているのでしょう。提出してACだと正解、WAだと不正解、CEだととにかくエラーが起きています。

A - Thermometer	fujimon0126 	なでしこ (cnako3 3.4.20)	100	219 Byte		116 ms	56180 KB	詳細
-----------------	---	----------------------	-----	----------	---	--------	----------	--------------------

ということで正解でした。

3.5 さいごに

初めて部誌を書くものでクオリティとか低いかもしれませんがご了承ください。
来年はバイナリコードとプログラミング言語について書きたいなと思います。

第 4 章

MythicMobs でオリジナルモブを作ろう

81 回生 ぽんだ

4.1 はじめに

皆さん、こんにちは。81 回生のぽんだです。今回は、Minecraft でオリジナルのモブ（生物）を作ります。よかったら最後まで読んでいってください。

前提条件

Minecraft の正規版を持っている/使える

Visual Studio Code(文中では VSCode と表記します) を使用できる

Java が PC にインストールされていて、バージョンが 17 または 21 である

環境

Windows 11

Minecraft 1.20.4

Java 21

PaperMC 1.20.4-499

4.2 サーバーの作成

※ここで立てるサーバーは自分および自分が繋いでいるネットワークに接続している人のみ入ることが出来るものになります。外部の人とのプレイはできません。

まず、<https://papermc.io/downloads/all> にアクセスします。左側には Minecraft の今までのバージョンが羅列されているので、1.20.4 を選択して #499 と書いてある行の Download を押しファイルをダウンロードしてください。

Legacy builds are not supported. Proceed at your own risk!			
Build	Changelog	Timestamp	Download
#499	9de3f75 Backport small fixes ba31f41 [ci skip] Clean up paperclip build-pr workflow	2024-10-31	Download
#497	d8d54d9 Only remove worldgen block entity on changed block (#10794)	2024-05-28	Download
#496	7ac24a1 Fix chunk data version check not running in chunk system 53d18b8 Adjust large packet handler to run when above protocol limit f4c7d37 [ci skip] Fix javadoc typo (#10445)	2024-04-25	Download
#493	a6b6ecd More Raid API (#7537)	2024-04-21	Download
#492	fc53ff5 Add Configuration for finding Structures outside World Border (#10437)	2024-04-21	Download
#491	c5f68ff Add CartographyItemEvent and get/setResult for CartographyInventory (#10396)	2024-04-21	Download
#490	a033033 Added chunk view API (#10398)	2024-04-21	Download
#489	3af1346 Allow setting player list name early	2024-04-20	Download
#488	988b814 Fix inventory desync with PlayerLeashEntityEvent (#10436)	2024-04-20	Download
#487	3b078f8 Add API for ticking fluids (#10435)	2024-04-20	Download

次に、任意の場所（自分がわかりやすい所が良いです デスクトップ等）に「paper1.20.4」など分かりやすい名前のフォルダを作成し、そのフォルダの中にダウンロードした paper-1.20.4-499.jar を入れてください。

そして、フォルダ内に新規テキストファイル（名前は後で変えるので何でもいいです）を作成し、そこに以下のコードを記述してください。

```
@ECHO OFF
```

```
java -Xms512M -Xmx4096M -jar paper-1.20.4-499.jar nogui
```

```
PAUSE
```

これを保存した後に、ファイル名と拡張子を「起動.bat」に変更してください。

起動.bat を起動するとコマンドプロンプトが起動すると思われるので、少し待ちます。

待つと、以下のような画面になるはずです。

```
Downloading mojang_1.20.4.jar
Applying patches
Starting org.bukkit.craftbukkit.Main
*** Warning, you've not updated in a while! ***
*** Please download a new build as per instructions from https://papermc.io/downloads/paper ***
System Info: Java 21 (Java HotSpot(TM) 64-Bit Server VM 21.0.1+12-LTS-29) Host: Windows 11 10.0 (amd64)
Loading libraries, please wait...
[08:49:55 WARN]: Failed to load eula.txt
[08:49:55 INFO]: You need to agree to the EULA in order to run the server. Go to eula.txt for more info.
続行するには何かキーを押してください . . . |
```

任意のキーを押してこのウィンドウを閉じ、先ほどのフォルダ内に追加された eula.txt を開いてください。この EULA というものは、End(er)-User License Agreement の略であり、ざっくり言うと「Minecraft やサーバー等のプログラムを使って利益を得ないでね」ということが記されています。(https://www.minecraft.net/ja-jp/terms/r3 MINECRAFT 利用規約および エンド ユーザー使用許諾契約書)

これを読んで理解し、同意してから次の手順に進んでください。

eula.txt 内のおそらく 4 行目に `eula=false` と書いてあるので、ここを `eula=true` と書き換えてください。

eula.txt を保存しもう一度起動.jar を起動します。しばらく待つと一番下に Timing Reset と表示されると思います。

表示されない場合は、Java の ver を確認する・起動.bat を確認する・eula.txt を確認するなどを試してみてください。

このサーバーに入りたい場合は、Minecraft を起動し、マルチプレイ→ダイレクト接続を押しアドレス欄に `localhost` と入力して接続すると入ることができます。

プラグインを導入するため、一度さっきのコマンドプロンプトで `stop` と実行してサーバーを閉じてください。これでサーバーの準備は完了です。

4.3 プラグインの導入

では、サーバーに MythicMobs を導入しましょう。

<https://mythiccraft.io/index.php?pages/official-mythicmobs-download/> にアクセスし、一番左の「Free version」をダウンロードします。次にサーバーの `plugins` フォルダにダウンロードした.jar ファイルを入れます。(執筆時 (2025/3/15) での最新 ver は 5.8.1 なので、実行する ver によっては少し変わる場合があります。) ここで一度サーバーを起動して、`plugins` フォルダ内に「MythicMobs」というフォルダができたことを確認してください。これで MythicMobs の導入は完了です。

```
└─ config
└─ data
└─ droptables
└─ items
└─ mobs
└─ packs
└─ randomspawns
└─ skills
└─ spawners
└─ stats.yml
```

MythicMobs フォルダ内はこのようになっているはず

4.4 オリジナルモブのコードを書く

MythicMobs フォルダ内の `mobs` フォルダを開き、`test.yml` というファイルを新しく作成します。このファイルを VSCode で開いてコードを書いていきます。(簡易テンプレートが最後にあるので是非使ってください)

ここでは以下の例を使用してコードの解説をしていきます。

コードの例

```
test:
  Type:  zombie
  Display:  '&d テスター君'
  Health:  100
  Armor:  0
  Damage:  10
  BossBar:
  Enabled:  true
  Title:  '&d テスター君'
  Range:  10
  Color:  fffffff
```

Style: SOLID

Options:

AlwaysShowName: true

Despawn: false

NoDamageTicks: 5

Equipment:

- diamond_helmet HEAD

- elytra CHEST

Skills:

- skill{s=testskill} @target ~onAttack

Drops:

- testdrops

Killmessages:

- '<target.name> はテスト中に事故に遭った'

一行ずつ説明していきます。

test: Mob の id を自由に決められます。これを使ってゲーム内で Mob を召喚できます。この場合 ID は **test** になります。

Type: zombie Minecraft 内でのバニラ Mob の id を記述してください。この場合ではゾンビ (zombie) を選択しています。

Display: '&d テスター君' 表示される名前の設定ができます。' ' 内に名前を記述してください。&d はカラーコード短縮形です。

(<https://x.gd/BgbU5> 詳しくはこちらを参照してください Minecraft Wiki-装飾コード)

Health:100 体力の設定です。この場合は 100 に設定しています。

Armor:0 その Mob が受けたダメージをどれくらい軽減するかの設定をします。

実際に受けたダメージ-Armor 値=受けたダメージとなります。今回は 0 なので 10 ダメージを受けた場合 10-0=10 でそのままの 10 ダメージを実際に食らうことになります。

Damage:10 その Mob が他に与えるダメージの設定です。攻撃力のことです。

BossBar: ボスバー（画面上部に表示される線の形のゲージのこと）の表示設定です。残りの体力を可視化できるので便利です。

Enabled: true true/false で有効/無効の切り替えができます。

Title: '&d テスター君' ボスバーの表示名を自由に決められます。名前と同じにするとわかりやすいです。

Range: 10 ボスバーをどの範囲に表示するかを決められます。Mob からの半径で指定します。

Color: fffffff ボスバーの色を決められます。カラーコードで記述してください。

Style: SOLID ボスバーのスタイルを決められます。大体 SOLID でいいです。

ボスバーが有効、表示名が「テスター君」となっていることが以下の画像のように確認できます。色もしっかり fffff=白になっています。

テスター君

Options: 各種オプションの設定です。

AlwaysShowName: true 常に名前を表示するオプションが設定できます。true だと Display で設定した名前が Mob の上に常に表示されるようになります。false だと Mob にカーソルを合わせた時のみ名前が Mob の上に表示されるようになります。

Despawn: false デスポーン=時間経過で Mob が消える機能の on/off を true/false で切り替えられます。

NoDamageTicks: 5 無敵時間（一度ダメージを受けてから次にダメージを受けられるようになるまでに時間のこと）の設定ができます。1 秒=20tick（1tick=0.05 秒）です。

Equipment: 装備の設定をすることができます。

- **diamond_helmet HEAD** HEAD,CHEST,LEGS,FEET,HAND,OFFHAND のどれかを指定することで、指定した場所にアイテムを持たせる/装備させられます HEAD: 頭 CHEST: 胴 LEGS: 脚 FEET: 足 HAND: 利き手 OFFHAND: 利き手と逆の手 です。利き手のデフォルトは右手です。

- **elytra CHEST**

この場合「ダイヤモンドのヘルメット」を「頭」に、「エリトラ」を「胴」に装備するように設定しているので、以下の画像のような見た目になります。



また、MythicMobs で作ったアイテムも装備させることが出来ます。（アイテムの作り方はスペースがないので割愛します）3D のテクスチャをつけたアイテムを頭に装備させてみました。（以下の画像）



Skills:

- skill{s=testskill} @target ~onAttack testskill という名前のスキルを~onAttack(攻撃したとき)に@target(ターゲット)に向かって使う という意味です。

Testskill には「攻撃した相手のチャット欄に preparing summon... と表示する」というスキルを設定しているため、以下の画像のようになります。(攻撃頻度が高いため、一秒に二回ほどの割合でこれがチャットに流れます。)



(攻撃されたときのチャット欄)

Drops:

- testdrops 別でドロップテーブルという Mob が何を報酬としてドロップするかを決めるものを作り、それを指定しています。

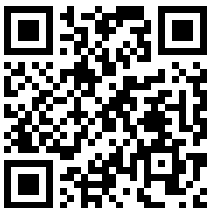
Killmessages: - '<target.name> はテスト中に事故に遭った' Mob がプレイヤーをキルした時のメッセージの設定です。<target.name>にはキルされたプレイヤーの名前が入ります。

この場合、名前を tuna334 だとすると、「tuna334 はテスト中に事故に遭った」と表示されます。

4.5 終わりに

このほかにも様々なオプションが存在しています。オプションを適切に設定することで自分だけのオリジナルモブが作れます。試しに作ってみた Mob(コードの例とは別のものです) との戦闘動画が下にありますので、よろしければご覧ください。

<https://youtu.be/Iot5pmpkppY>



参考にしたサイト

アジ鯖 Adminresources-MythicMobs

<https://adminresources.azisaba.net/plugins/MythicMobs/>

Mythic Mobs Wiki-home

<https://git.mythiccraft.io/mythiccraft/MythicMobs/-/wikis/home>

簡易テンプレート

ほぼ必須のものしか載せていませんので、参考にしたサイトなど見て自由に作ってください。

`[]`

`Type: []`

`Display: '[]'`

`Health: []`

`Armor: []`

`Damage: []`

第 5 章

Python を使った自作 SNS の開発

mikan0131

5.1 はじめに

こんにちは！ 81 回生の mikan0131 です。僕は競プロを中心に活動していますが、最近 Web アプリケーションに興味を持ったので、その一つの成果として、Python で TweetApp という SNS を作りました。最後まで読んでくれるとうれしいです。この Web 版では、TweetApp の細かい機能まで踏み込んでみようと思います。少々長い記事ですが、お付き合いください。

注：途中までは、紙版とかなり内容がかぶっているのですが、紙版を読んでくださった方は、「あ、ここはもう知ってるな」と思った部分を飛ばして読んでくださってもかまいません。

5.2 そもそも Web アプリケーションってなに？

僕の作ったアプリを紹介する前に、Web アプリケーションがただの Web サイトと何が違うのか説明します。

HP などのウェブサイトは、静的な Web サイトと呼ばれており、閲覧者が Web サーバーに対して送ったりクエスト（例えば「<https://npca.jp/>のページが見たい！」など）に対して一定のレスポンス（Web サイトを表示するための情報）しか送りません。

一方、Web アプリケーションなどの動的なウェブサイトは、閲覧者が送ったリクエストに対して、Web サーバー内のデータベースから、情報を参照して、一人ひとりに違うレスポンスを返しています。（図 1）

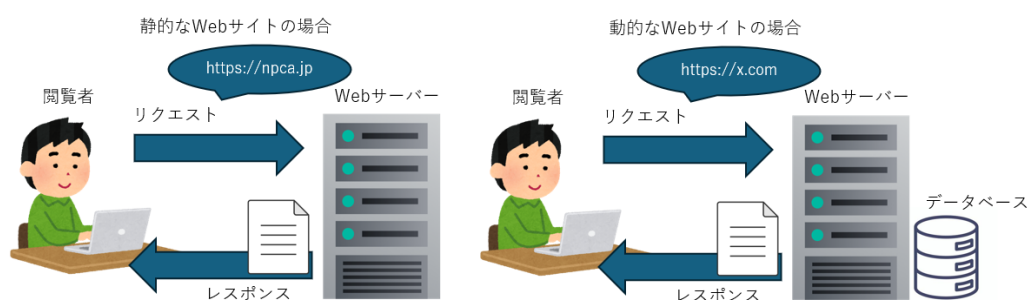


図 5.1: 静的な Web サイトと動的な Web サイトとの違い

このように、静的な Web サイトと動的な Web サイトではかなり違いがあります。今回僕が開発したのは、そのなかでも動的な Web サイトです。題名にもある通り、Python というプログラミング言語を使って、SNS のようなものを作りました。では、今から具体的な開発について紹介します。

5.3 事前準備

5.3.1 Python のモジュールをインストール

今回は Python の様々なモジュール（プログラミングでの、たくさんの道具が入った道具箱のようなもの）をインストールする必要があるので、以下のようなコードでインストールします。

ソースコード 5.1: install.bat

```
1 $ pip install flask flask_sqlalchemy SQLAlchemy PyMySQL
```

このコードで Python のモジュール Flask, Flask SQLAlchemy, SQLAlchemy, PyMySQL をインストールします。

- Flask
このモジュールはこのアプリの根幹になるモジュールです。Python によるページの表示・ページのルーティング（URL に対し表示するページを決めること）・ページ遷移などを担当します。
- PyMySQL
このモジュールは Python からデータベースにアクセスするためのモジュールです。このモジュールを使うことで、Python からデータベースのデータの取り出し、データの編集などを行うことができます。
- Flask SQLAlchemy
これは、SQLAlchemy という、データベースのデータを Python で扱える形にするモジュールの、Flask 専用のものです。このモジュールを使うことで、Python でデータベースのデータが扱いやすくなるだけでなく、SQL インジェクションなどのハッキングからサーバーを守ってくれるので、この Web 開発ではかなり重要になります。
- SQLAlchemy
Flask SQLAlchemy では関数の足りないところがあったので、これで補います。

5.3.2 ファイル構造

当たり前ですが、ファイル構造がちゃんとしてないと Web アプリケーションは機能しないので、Web アプリケーションのファイルを格納するディレクトリの中にファイル構造を設定します。これには決まりがあるので、とりあえず、下の画像のようにファイル構造を設定します。



図 5.2: ファイル構造

それぞれのディレクトリについて説明すると、

- app.py
この Python スクリプトでアプリ全体を統括します。
- static
このディレクトリは Web サイトの見た目となる HTML ファイルのオプションとなるファイルを格納します。
- templates
このディレクトリは、Web サイトの見た目である HTML ファイルを格納します。
- css・js・img
左から順に、CSS ファイル（サイトをデコレーションする）・JavaScript ファイル（サイトに動きを付ける）・画像ファイル（サイトに添付する画像）を格納します。

5.3.3 データベースの設定

「そもそも Web アプリケーションって何？」で説明した通り、Web アプリケーションにはデータベースが欠かせません。ということで、データベースを用意します。

今回は、MySQL というデータベースのシステムを Linux 系の OS である Ubuntu で動かすことにします。

さらに、データベースの編集を簡単にするために、phpMyAdmin というツールを用意することにします。

MySQL と phpMyAdmin の設定はやりたい人は各自やってみてください。—(筆者が覚えてない…)

この時、tweet_app というデータベースを作り、user・groups・posts という 3 つのテーブルを作り、その設定を次のようにします。

user テーブル

このテーブルでは、user に関する情報を管理します（一番大事）。

id では、ユーザーの ID（番号）、name では、ユーザーネーム、mail_address では、ユーザーのメールアドレス、password_hash_md5 ではユーザーのパスワードのハッシュ（のちに説明します）、groups では、ユーザーの所属するグループ、addresses では、ユーザーの友達のユーザー、request では、ユーザーに届いている友達申請を管理します。

カラム名	タイプ	autoincrement	デフォルト値
id	int	True	なし
name	varchar(200)	False	なし
mail_address	varchar(100)	False	なし
password_hash_md5	varchar(200)	False	なし
groups	json	False	なし
addresses	json	False	なし
request	json	False	なし

posts テーブル

このテーブルでは、ユーザーの投稿の情報を管理します。

from_address では、そのポストを投稿したユーザー、to_group では、そのポストが投稿されたグループ、to_address では、そのポストが送られたユーザー、content では、そのポストの内容、そして、created_at では、そのポストが投稿された日時を管理します。

カラム名	タイプ	autoincrement	デフォルト値
from_address	varchar(100)	False	なし
to_group	varchar(100)	False	NULL
to_address	varchar(100)	False	NULL
content	text	False	NULL
created_at	timestamp	False	CURRENT_TIMESTAMP

groups テーブル

このテーブルでは、グループ（掲示板）の情報を管理します。

id はグループの ID（番号）、name ではグループの名前、password_hash_md5 はグループに入るためのパスワードのハッシュ（のちに説明します）、about はそのグループについて表す文章、owner はそのグループの管理者権限を持つユーザーを管理します。

カラム名	タイプ	autoincrement	デフォルト値
id	int	True	なし
name	varchar(200)	False	なし
password_hash_md5	varchar(200)	False	なし
about	text	False	なし
owner	varchar(200)	False	なし

こんな表だけ見せられてもわからないと思うので、少し説明します。

項目について

- カラム名
それぞれのテーブルのデータの名前です。
- タイプ
データの型（整数・文字列・配列など）を表します。例えば、int なら整数、varchar(100) なら 100 文字までの文字列、json なら配列です。
- デフォルト値

各データのもともとの値を設定します。つまり、何もデータを書き込まなければ、デフォルト値になるということです。デフォルト値はほとんど設定してませんが、NULL なら、何も入っていない空っぽの値、CURRENT_TIMESTAMP なら、データが作られた時の時刻を表します。

これで、ひとまず事前準備は完了です。では、お待ちかねの、さっそく作っていく作業に入ります！

5.4 Web サイトの見た目を作り、表示する

5.4.1 HTML ファイルを作る

まず、事前準備の、ファイル構造の時点で気づいてる方もいるかもしれませんが、サイトの見た目となる HTML ファイルを、templates ディレクトリの中に作ります、例えば…

ソースコード 5.2: example.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset = 'ctf-8'>
5     <title>TweetApp</title>
6     <style> body {margin: 10px;} </style>
7   </head>
8   <body>
9     <h1>Hello TweetApp!</h1>
10  </body>
11 </html>
```

こんな感じですね。これを実際に Microsoft Edge なんかで表示すると、左の画像のようになります。しかし、この example.html を複雑にして、さらに static ディレクトリに配置した画像ファイルなどを添付し、CSS ファイルで HTML ファイルを飾り付けると、右のような画像になります。



図 5.3: example.html の場合



図 5.4: TweetApp の場合

お～、それっぽい Web サイトの画面ができてきました！ちなみに、僕はこの時点で機能なものにも作ってないのに 3 日ぐらい過ぎてました。

5.4.2 HTML ファイルを表示する

はい、この時点では、まだ HTML ファイルを作っただけです。これを表示する必要があります。なので、templates ファイルの中に example.html を作ったら、app.py に次のように記述します。

ソースコード 5.3: app.py

```
1 from flask import Flask, render_template
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def index():
7     return render_template('example.html')
```

このコードでは、1 行目で、事前準備でインストールした Python モジュールをインポート（プログラムで使えるようにすること）し、3 行目で、このアプリで Flask を使えるようにします。そして、5 行目で「/」という URL にアクセスしたときに行う処理だということを示してから、6 から 7 行目で example.html を表示できるようにします。

5 行目の () の部分を変えて同じようなプログラムを書けば、様々な URL に対して表示する HTML ファイルを指定することができます。

これで、TweetApp でサイトを表示することができました！次からは、さらに app.py や HTML ファイルに特殊な記述をして、様々な機能を実装していきます。

5.5 ログイン機能を作ってみる（データベース操作の練習）

ということで、さっそく TweetApp の根幹の機能であるログイン機能を作っていきましょう！事前準備の時に、user テーブルを作ったのを覚えていますか？あのテーブルを今から早速使っていきます。

5.5.1 データベースに接続する

ログインの情報の照合はもちろんデータベースを使って行うのですが、データベースに接続し、app.py からデータベースを動かさないことには、何も始まりません。なので、app.py にこう記述します。

ソースコード 5.4: app.py

```
1 from flask_sqlalchemy import SQLAlchemy
2
3 # データベースアクセス
4 user = 'root'
5 host = 'localhost'
6 database = 'tweet_app'
7 password = '****' #一応伏字
8
9 db_uri = f'mysql+pymysql://{user}:{password}@{host}/{database}?charset=utf8'
10 app.config['SQLALCHEMY_DATABASE_URI'] = db_uri
```

```

11 db = SQLAlchemy(app)
12
13
14 class User(db.Model):
15     __tablename__ = 'user'
16     id = db.Column(db.Integer, primary_key = True, autoincrement = True)
17     name = db.Column(db.Text())
18     mail_address = db.Column(db.Text())
19     password_hash_md5 = db.Column(db.Text())
20     groups = db.Column(db.JSON())
21     addresses = db.Column(db.JSON())
22     request = db.Column(db.JSON())

```

このようにすることで、データベースの情報を Python のオブジェクトという型の情報として扱うことができます。要するに、Python の中でとても使いやすくなります！例えば、mikan0131 というユーザーの情報が欲しいな、と思えば、`mikan0131 = User.query.filter(User.name=='mikan0131').first()` と書けば、変数 mikan0131 の中に mikan0131 のパスワードのハッシュ（のちに説明します）も、所属グループも友達の情報もすべて入ります。こうして、Python の中でデータベースの情報を扱うことができるようになるのです。

5.5.2 ログイン情報の照合

では、データベースに接続したところで、ログイン機能の本格的な実装に移ります。まずは、Web サイトに入力した情報を、app.py が受け取る必要があります。このとき、HTML に form という入力情報を送る要素を追加します。なので、ログイン画面の HTML ファイルは次のように記述します。

ソースコード 5.5: login.html

```

1 <head>
2     <link rel = 'stylesheet' href = '/static/css/login.css'>
3     <script src = '/static/js/login.js'></script>
4 </head>
5 <body>
6     <div class = 'login-form'>
7         <h1>Please Login</h1>
8         <form action = '/login' method = 'post'> <!-- ←これがフォーム -->
9             <input class = 'username-form form' placeholder = 'username' type = '
                text' name = 'user_name'>
10             <p></p>
11             <input class = 'password-form form' placeholder = 'password' type = '
                password' name = 'password'>
12             <p></p>
13             <button class = 'enter-button' type = 'submit'>ENTER</button>
14         </form>
15     </div>

```

```
16     <a class = 'register-link' href = '/register'>Don't you have an account?  
    Register here.</a>  
17 </body>
```

このようにすることで、下の画像のようになり、app.py に入力した情報を送ることができます。

Please Login

NPCA

.....

ENTER

図 5.5: ログイン画面

これで、app.py に情報を送ることができたので、次は、app.py でデータベースと照合します。今回は、user テーブルと照合します。user テーブルにはユーザーの情報が格納してあります。ここで、(やっと) ハッシュについて説明します。ハッシュとは、ある文字列を違う文字列に変換するための関数であり、変換後の文字列から元の文字列を復元するのは不可能とされています。それゆえ、ハッシュはパソコンのパスワードを管理するのに使われています。なので、user テーブルの password_hash_md5 には、ユーザーのパスワードのハッシュを格納し、入力情報もハッシュ化して、app.py で、ハッシュ化した文字列が一致するかどうか判定します。では、app.py でデータベースの情報を取り出し、入力情報をハッシュ化して、ログインできるか照合してみましょう！

ソースコード 5.6: app.py

```
1 # ログインフォーム  
2 @app.route('/login', methods = ['POST', 'GET'])  
3 def login_form():  
4     if request.method == 'POST': # ポスト通信に設定  
5         # 送られてきたフォームから情報を抽出  
6         user_name = request.form['user_name']  
7         password = request.form['password']  
8         # 入力されたパスワードをハッシュ化  
9         enc_password = bytes(password, encoding = 'utf-8')  
10        password_hash = hashlib.md5(enc_password).hexdigest()  
11        # 入力情報のハッシュとデータベースから引き出したパスワードのハッシュを比較  
12        if (check_login(user_name, password_hash)):  
13            # セッションのユーザーネームを編集
```

```

14     session['username'] = user_name
15     session['login'] = 'ok'
16     return redirect(url_for('index'))
17     # 一致しなかった場合、ログインフォームにもう一回リダイレクト
18 else:
19     session['login'] = 'falied'
20     return redirect(url_for('login_form'))
21 login = True
22 if (session['login'] == 'falied'):
23     login = False
24 print(login)
25 return render_template('login.html', login = login)

```

このコードで、入力したパスワードのハッシュと、データベースに記録してあるパスワードのハッシュが一致すれば、セッション（コンピューターのブラウザに記録される情報）に自分のユーザー名が記録されます。ただし、セッションはこのアプリが管理しているパスワードによって暗号化されているので、簡単にセッションに情報を書き込むことはできません。このようにして、ログイン機能を実装しました。

5.6 グループに投稿し、投稿を編集・削除できるようにする

このアプリを実用化するために、グループに投稿し、それを保存・削除・編集できるような機能を実装します。

5.6.1 どうやって投稿を表示する？

今回は、前述のログイン機能と違って、グループの投稿を更新して、投稿されたポストによって違う Web サイトを表示しなければなりません。つまり、ただの HTML ファイルでは投稿を表示できない！

そこで、「Web サイトの見た目を作り、表示する」で、Web サイトを表示するために、`render_template` という関数を使ったのを覚えているでしょうか？

あの関数で、引数に“example.html”というファイル名を入れて、その HTML ファイルを表示することができました。しかし、実はこの関数、`render_template("index.html", username="mikan0131")` のようにすることで、HTML ファイルに情報を送ることができるのです。例えば、このように記述すれば、index.html という HTML ファイルで username という変数が使えるようになります。

ソースコード 5.7: layout.html

```

1 <!DOCTYPE html>
2 <html lang = "ja">
3     <head>
4         <meta charset = "utf-8">
5         <title>TweetApp</title>
6         <link rel = 'stylesheet' href = '/static/css/layout.css'>
7         <link rel = 'icon' href = '{{ url_for('static', filename = 'favicon.ico'
            '))}}'>

```

```

8      <!--animations' script-->
9      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
        </script>
10     <script src = '/static/js/layout.js'></script>
11     <!--icons' css-->
12     <link href="https://use.fontawesome.com/releases/v6.7.2/css/all.css" rel="
        stylesheet">
13     {% block static %}
14     <!-- ほかのファイルで、スクリプトをここに入れられる -->
15     {% endblock %}
16 </head>
17 <body>
18     <header>
19         <img id = 'icon' class = 'header-logo' src = '/static/favicon.ico'
            width = 60 height = 60>
20         <p class = 'header-logo'><a href = '/'>TweetApp</a></p>
21         <form class = 'search' action = '/search' method = 'get'>
22             <i class="search-icon fa-solid fa-magnifying-glass"></i>
23             <input class = 'search-input' name = 'word'>
24         </form>
25         <ul id = 'header-holder' class = 'header-holder'>
26             {% if session['login'] == 'ok'%} <!-- ログインしているか判定 -->
27                 <li id = 'username'>{{ session['username'] }}</li>
28                 <li class = 'list'><a href = '/profile/{{ session['username']
                    }}'>プロフィール
                    </a></li>
29                 <li class = 'list'><a href = '/logout'>ログアウト</a></li>
30             {% endif %}
31             {% if session['login'] != 'ok' %}
32                 <li id = 'username'>? ? ? ?</li>
33                 <li class = 'list'><a href = '/login'>ログイン</a></li>
34                 <li class = 'list'><a href = '/register'>新規登録</a></li>
35             {% endif %}
36         </ul>
37     </header>
38     {% block content %}
39     <!-- ほかのファイルで、スクリプトをここに入れられる -->
40     {% endblock %}
41 </body>
42 </html>

```

これが、TweetApp のホームの上部を構成する HTML ファイルなのですが、皆さんは、このコードに `{{}}` や `{% %}` という記号があることに気づいたでしょうか？このような括弧の中では、Python の条件分岐や、変数を使うことができます。ひとつずつ説明すると、

- `{{ 変数 }}`

このようにすると、「変数」を HTML に表示することができます。

例えば、`render_template("layout.html", username="mikan0131")` とすると、`layout.html` で `{{username}}` とすれば、`layout.html` で `mikan0131` と表示できます。

- `{% if ~~~~ %} . . . . {% endif %}`

このようにすることで、Python の条件分岐である if 文を使うことができます。

- `{% block ??? %} . . . . {% endblock %}`

このようにすることで、ほかの HTML ファイルで、この HTML ファイルをテンプレートとして使うことができます。

こんな感じです。このようにして、HTML の中でも、プログラムを書いて。表示する内容を変えることができます。この機能を利用することで、グループの投稿を Web サイトに表示することができるのです。

5.6.2 グループを機能させるためのページの編成

ここからは、少しページの繊維が複雑になるので、どこからどのページに遷移するのか整理しておきます。

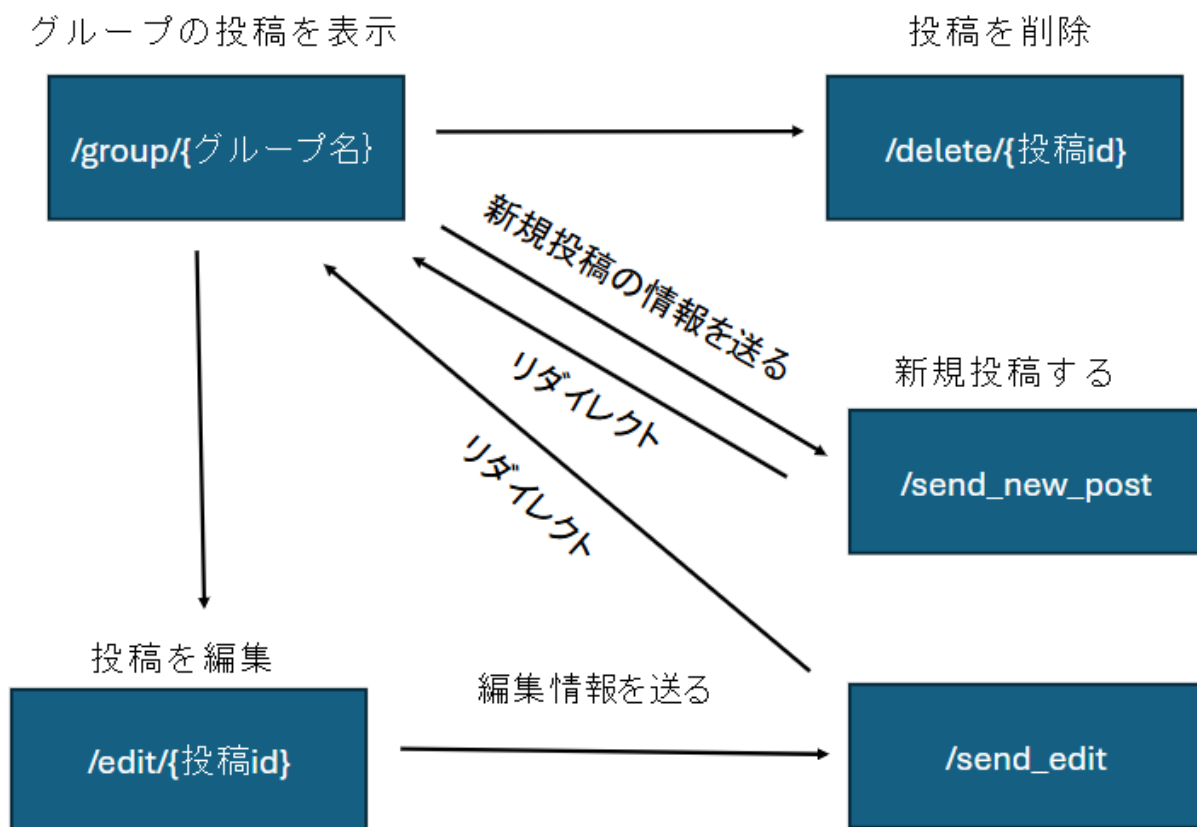


図 5.6: ページ遷移図

ページの遷移図ができたところで、機能の実装に移ります。

5.6.3 グループに投稿できるようにする

ログイン機能を作ったときに、HTML ファイルに form という要素を埋め込んだのを覚えているでしょうか？この要素を、グループの投稿表示画面に埋め込み、新規投稿フォームとします。このフォームを入力すると、/send_new_post に送信されるわけですが、app.py の /send_new_post のコードは次のようになります。

ソースコード 5.8: app.py

```
1 @app.route('/send_new_post', methods = ['POST'])
2 def send_new_post():
3     content = request.form['content'] # 情報抽出
4     new_post = Posts() # 新しい投稿データを作成
5     new_post.content = content # 新規投稿の内容を代入
6     new_post.from_address = session['username'] # 自分のユーザー名を新規投稿の送り主にす
        る
7     new_post.to_group = request.form['group'] # どのグループに贈るか指定
8     db.session.add(new_post) # データベースを同期
9     db.session.commit()
10    # グループの投稿表示ページにリダイレクト
11    return redirect(url_for('show_group', group = new_post.to_group))
```

このようにして、

グループで新規投稿を入力 → /send_new_post にリダイレクト → app.py で posts テーブルを編集 → 投稿表示ページにリダイレクトという流れができました。実は、このコードでは、new_post という変数に posts テーブルの新しいデータを作っていて、そのデータに値を代入することで、Python を使っていながら、データベースを編集することができるのです。ここでも、SQLAlchemy の力を実感できると思います。

5.6.4 グループの投稿を編集・削除できるようにする

今回は、投稿ページに編集ボタン・削除ボタンを作り、それを押すと、編集リンク・削除リンクに贈られるという設定にします。あと、他人の投稿を削除しようとする輩を防ぐため、セッションに保存されているユーザー名に応じて、編集できるかどうか判定します。-(ちゃんとセキュリティ対策する自分えらい) まずは、ポストの編集を実装します。

ソースコード 5.9: app.py

```
1 @app.route('/edit/<int:id>', methods = ['GET'])
2 def edit(id):
3     post = Posts.query.get(id) #特定の番号の投稿を取得
4     if (post.from_address == session['username']): # 投稿した本人かどうか判断
5         default = post.content
6         return render_template('edit.html', id = id, default = default)
7     else:
8         return "<h1>You can't edit this post. Please login first.</h1>"
```

```

9
10 @app.route('/send_edit/<int:id>', methods = ['POST'])
11 def send_edit(id):
12     content = request.form['content'] # フォームに入力された内容を取得
13     editted_post = Posts.query.get(id) # 特定の番号の投稿を取得
14     editted_post.content = content # 投稿の情報を編集
15     # 送り主と送り先を編集
16     group = editted_post.to_group
17     user = editted_post.from_address
18     if (user != session['username']): # 投稿した本人かどうか判断
19         return "<h1>You can't edit post" + str(id) + ". Please login first.</h1>"
20     db.session.commit()
21     return redirect(url_for('show_group', group = group))

```

上のコードは/edit/{投稿id}にアクセスしたときのソースコード、下のコードは/send_editにアクセスしたときのソースコードです。/edit/{投稿id}にアクセスすると、投稿を編集するフォームが表示されます。その情報を/send_editに送り、そこでデータベースを操作します。フォームを入力したら、どこかのURLに飛ぶ必要があるのですが、投稿ページにリダイレクトするだけの/send_editというURLを用意し、ページを表示することなく、データベースを編集します。(下図) 実際、編集してみると、編集するフォームからすぐに投稿ページに移っているように見えます。

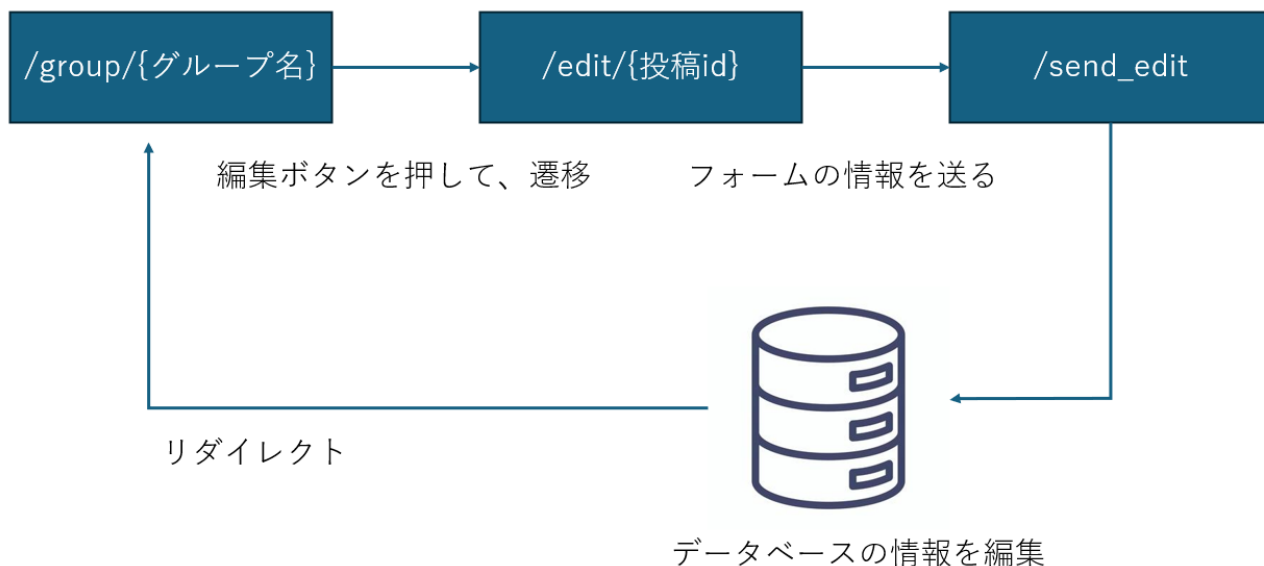


図 5.7: 投稿の編集のイメージ

s 次に、ポストの削除を実装します。

ソースコード 5.10: app.py

```

1 @app.route('/delete/<int:id>')

```

```

2 def delete_post(id):
3     delete_post = Posts.query.get(id) #削除する投稿のデータを取得
4     db.session.delete(delete_post) #削除するための操作を行う（ここではまだデータベースに反
        映されない）
5     group = delete_post.to_group
6     user = delete_post.from_address
7     if (user != session['username']): # 操作しているユーザーが投稿を削除できるか判断
8         return "<h1>You can't delete this post. Please login first.</h1>"
9     db.session.commit() #データベースを同期（ここでやっとデータベースに反映される）
10    return redirect(url_for('show_group', group = group))

```

これも、原理は編集するときと同じです。しかし、今回は投稿の内容などの情報を入力する必要がないので、投稿ページから直接データベースを操作するためだけの URL、`/delete/{投稿id}` に飛び、データベースを操作します。（下図）この時も、編集するときと同じく、削除ボタンを押してもページが遷移しているようには見えません。

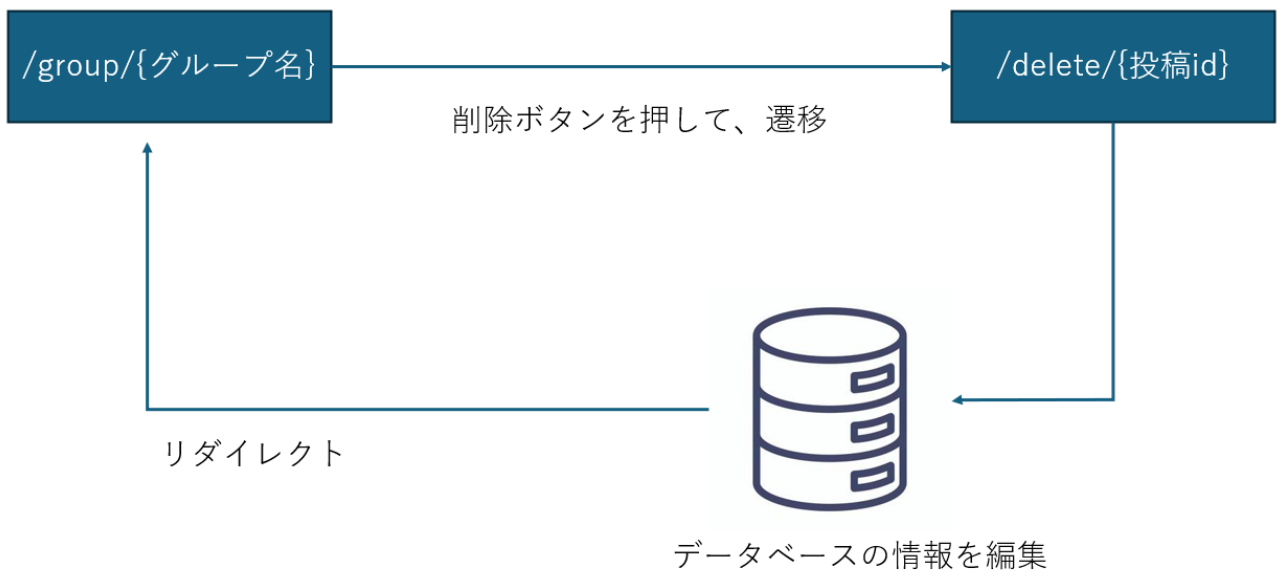


図 5.8: 投稿の削除のイメージ

投稿の編集・削除ともに、データベースを操作する「だけ」の URL に飛ばして、そこでデータベースを操作することで実装することができました。

これで、グループの投稿・編集・削除の実装をすることができました。Flask と SQLAlchemy を組み合わせることで、このような複雑な操作もできるようになります。

これで、SNS とまではいかなくとも、掲示板のような機能がほぼ完成したといってよいでしょう。これに、個人チャットなどの機能を追加していけば、SNS の機能は完成します。皆さんは、X (旧 Twitter) や LINE などが作っているような SNS を作るのは難しそう…というイメージがあったかもしれませんが、僕のような個人でも、すこし知識をかじっただけで、簡易的なものはできてしまうのです。

さて、ここまでいかがだったでしょうか？これをきっかけに、Web 開発に興味を持つ人がいればうれしい

です。

そして、そのようなことに興味を持てば、ぜひ、プログラミングの勉強をして、やってみてください。
最後に、この長い記事を読んでくださった方々、本当にありがとうございました！

5.7 おまけ

今回作った「TweetApp」のソースコードは、下の QR コードまたは URL のページにすべて載っています。（さらに、実はこれ、環境さえ整えれば、機能します！）

また、この部誌に書ききれなかった「TweetApp」の製作記は、少しずつ下の URL のブログに載せていこうと思うので、そちらも、よろしくお願いします！（このブログでは、本当は個人チャット機能なども実装する予定です！）

ソースコード: <https://github.com/mikan0131/TweetApp>



ブログ: <https://qiita.com/mikan0131/items/143d7a096f4c3827e6c0>

第 6 章

サイクリングの周回ルートを探索する

79 回生 number_cat

6.1 はじめに

こんにちは、79 回生の number_cat です。いつのまにか高 2 になってしまいました。

僕はときどきサイクリングをしているのですが、サイクリングルートを自動で作成できたら便利だと思ったので、自分で作ってみることにしました。

今回作成するルートは、特定の目的地を設定するのではなく、一定距離を走行して出発地点に戻る「周回ルート」です。特に、なるべく良いルートを生成することを目指します。

6.2 周回ルートの良さを評価する

良いルートとはどのようなものでしょうか。まず、以下の 2 つの基準を満たしていることが望ましいと考えます。

1. 指定距離に近い（例：30km のルートを作る場合、なるべく 30km に近いこと）
2. 道の重複が少ない（同じ道を何度も通るのは面白みに欠けるため、できるだけ異なる道を通ること）

さらに、今回は次の基準も考慮します。

3. 路上条件が快適である（なるべく坂が少なく、広い道を通ること）

快適な路上条件の定義には議論の余地がありますが、本研究では、坂の少なさや道路の広さを指標とします。そして、この基準がどの程度満たされているかを評価し、単純な手法と比較してどれだけ快適なルートを選択できているかを検証します。

これらの基準を数値化するため、それぞれ「距離ペナルティ」「重複ペナルティ」「非快適度」を算出し、3 つの合計を「総ルートコスト」と定義します。この値が小さいほど、より良いルートであると判断します。

6.3 実装環境

実際にルートを作成するプログラムを実装しました。本研究では神戸市内を対象としています。

道路データは OpenStreetMap から取得し、勾配を求めるための標高データは国土地理院のデータを使用しました。

データの取得には Anaconda 環境で OSMnx と GeoPandas を用いました。具体的には、OpenStreetMap の道路データを OSMnx で取得し、標高データとして 国土地理院の基盤地図情報数値標高モデル (10m メッシュデータ) を使用しました。この標高データは 基盤地図情報ビューア を用いて Shape 形式に変換し、それを GeoPandas に読み込んで処理しました。

取得したデータを一度テキストファイルに出力し、そのあと C++ で読み込んで経路探索アルゴリズムを実行しました。

6.4 手法

周回ルートは最大 40km 程度なので、出発地点から半径 20km 以内の道路を考えれば十分です。この範囲の交差点 (ノード) 数と道路 (エッジ) 数は約 10 万ほどあり、効率的なアルゴリズムが必要になります。計算量は $O(E \log V)$ (E は道路数、 V は交差点数) に抑えました。

また、簡単のため、出発地点は交差点の一つとします。

6.4.1 ルートの作成方法

まず、2 つの経由点を選び、出発地点とそれぞれを結ぶことで周回ルートを作成します。各点間のルートは、坂のきつさや道の広さを考慮した「非快適度」に道の長さを重みづけしたものをコストとして扱い、その最小コストルートをダイクストラ法で求めます。ただし、経由点の選び方によって距離が大きく変わるため、さまざまな候補を試し、最も総ルートコストが小さいものを採用します。

6.4.2 計算量の削減

ここで問題になるのは計算量です。単純に複数個 (n 個) の経由点の組み合わせについて最小コストルートを求めると、そのたびに $O(E \log V)$ 時間かかるので、全体で $O(nE \log V)$ になってしまいます。

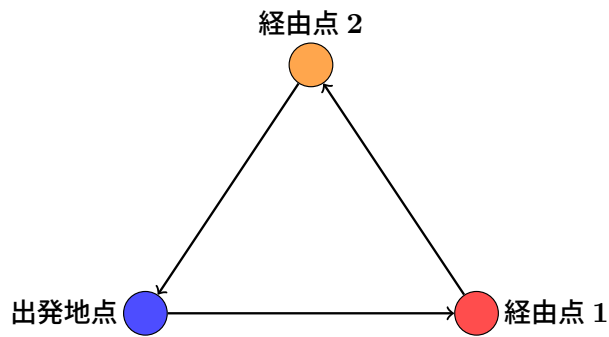
そこで、1 つ目の経由点 (経由点 1) を固定し、2 つ目の経由点 (経由点 2) を動かすことにします。こうすると、ダイクストラ法を 2 回使うだけで、

- 経由点 1 → 各交差点 の最小コストルート
- 各交差点 → 出発地点 の最小コストルート

がそれぞれ求まり、経由点 1 を固定したときに各交差点を経由点 2 とする周回ルートを $O(E \log V)$ で算出できます。最終的に、試したルートの中で総ルートコストが最小のものを出力します。

補足: 経由点 1 は、出発地点からのルート長が 30~40% で、非快適度の平均が低い点とします。極端な形状のルートの生成を防ぐためです。

実装の詳細は https://github.com/numbercat314/cycling_route_search に載せる予定です。



6.5 経路例



図 6.1



図 6.2

6.6 検証

路上条件を考慮できているのか確認するために、距離と重複のみを考慮したナイーブな手法と総ルートコストを比較します。

神戸市内を対象として、ランダムに 100 の出発地点を選びます。さらに、5 種類の距離を指定して、総ルートコストの平均値を算出しました。結果を以下の表に示します。

$X_{[\text{km}]}$	今回	ナイーブ	比率
3	0.348	0.453	0.77
5	0.305	0.393	0.78
10	0.195	0.292	0.67
20	0.149	0.256	0.58
40	0.145	0.240	0.60

表 6.1: 提案手法と比較手法の結果

今回の手法では、ナイーブな手法と比べて総ルートコストを 20～40% 削減できています。つまり、路上条件をしっかりと考慮できていることが分かります。

6.7 おわりに

本研究では、サイクリングの周回ルートを、路上条件を考慮しながら探索できるプログラムを実装しました。その結果、ナイーブな手法よりも良いルートを生成できることが確認できました。

今回は「なるべく坂が少なく、広い道を通る」ことを快適な路上条件としましたが、「なるべく坂が多く、狭い道を通る」といった設定に変更しても、同じアルゴリズムでルートを作成できます。ただし、「少し坂がほしい」といったルート全体のバランスを考慮しないと最適化できない条件には対応できません。

こうしたルート全体のバランスを考慮した最適化を行う手法として、焼きなまし法などの確率的最適化手法が考えられます。焼きなまし法を用いることで、局所最適解に陥ることを防ぎつつ、総ルートコストがより小さい解を探索できるように思います。どうして思いつかなかったのでしょうか……。焼きなまし法のための初期解として利用することくらいはできそうですが。

また、現状はシミュレーションのみなので、実際にこのルートを走ってみて、体感的な評価を検証するのも面白そうです。

以上、サイクリングの周回ルート探索でした！

6.8 参考文献

- c60evaporator. ”【Python で GIS】GeoPandas まとめ”. Qiita. 2021-07-04.
<https://qiita.com/c60evaporator/items/ac6a6d66a20520f129e6>
- NAVITIME_Tech. ”自転車向け「周遊ルート機能」の開発秘話”. note. 2024-02-28.
https://note.com/navitime_tech/n/n9b060bc0dd82
- ritogk. ”地図からこだわりの道を抽出する方法”. Zenn. 2023-12-25.
<https://zenn.dev/homing/articles/f9a314841c737d>
- Captain-K. ”OpenStreetMap と Python で地域道路データを解析: OSMnx と GeoPandas の使い方”. Hangout Laboratory.
<https://hanlabo.co.jp/memorandum/3110>
- moshi. ”【Python】OSMnx で車両走行道路の最短経路探索&可視化をしてみる”. Qiita. 2021-04-12.
<https://qiita.com/moshi/items/e383491664d028cd2166>
- WisteriaHill. ”OSMnx を使って道路ネットワークを取得してみる”. FRONT. 2018-11-02.
<https://wisteriahill.sakura.ne.jp/CMS/WordPress/2018/11/02/osmnx-graph-network>
- kntoshiya. ”NetworkX で地理院ベクトルタイルの道路データを使ってネットワーク解析”. Qiita. 2023-12-16.
<https://qiita.com/kntoshiya/items/77a8964fd36ef57e4425>