プログラミング演習I第12回レポート

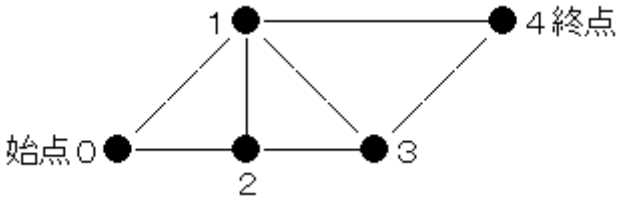
学籍番号：2364902

名前：キム　ギュソク

(1) 課題番号と課題内容

A. 基本課題２

i. ５つの地点０～４を結ぶ道路網が図のようにある。始点を０とし終点を４とした時、０から４に至る経路をすべて求める。
ただし、同じ地点を２度通ってはいけない。



(2) フローチャートまたは疑似言語によるアルゴリズムの記述

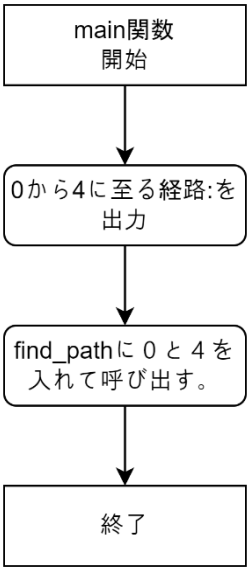


図 1。main関数フローチャート

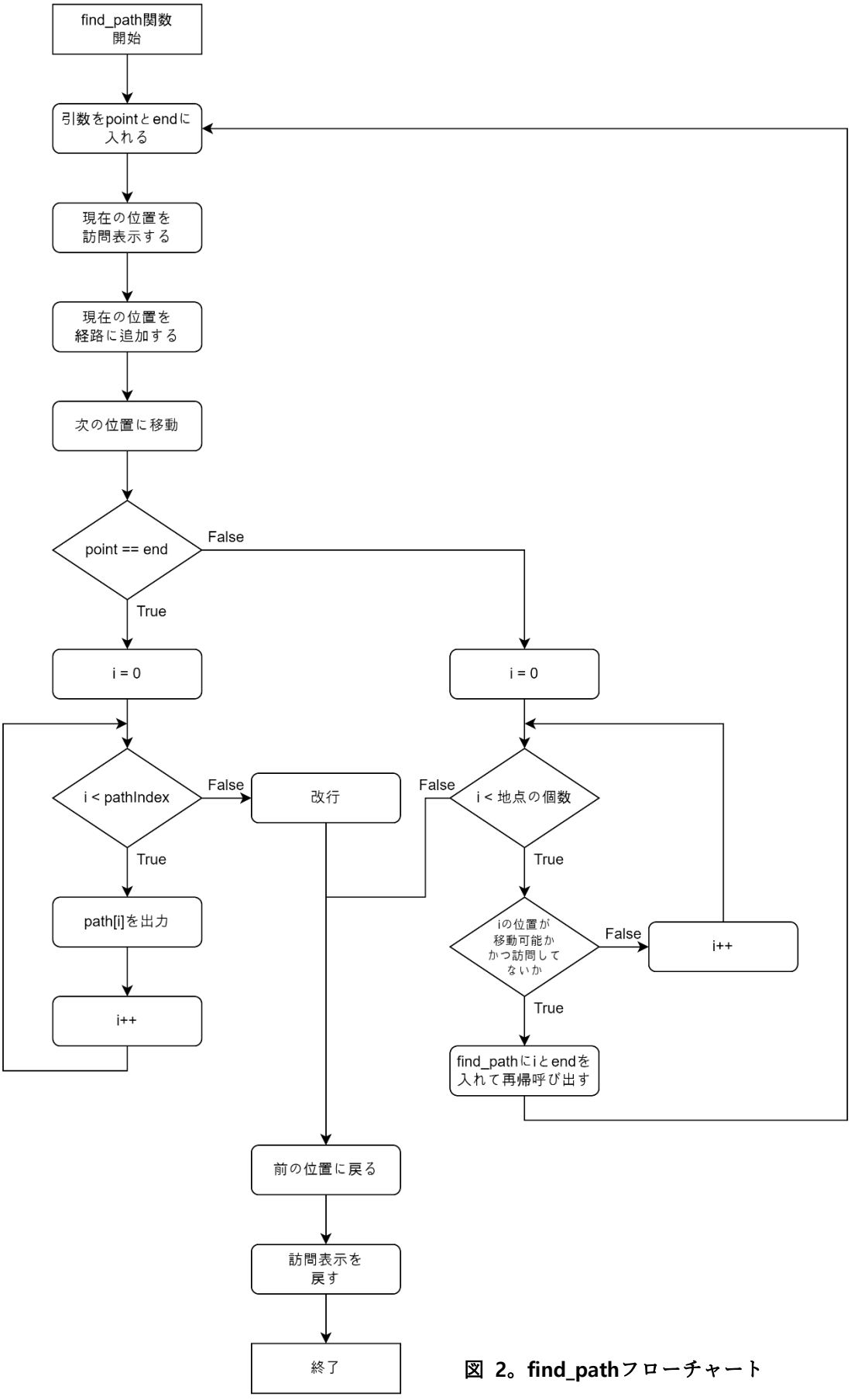
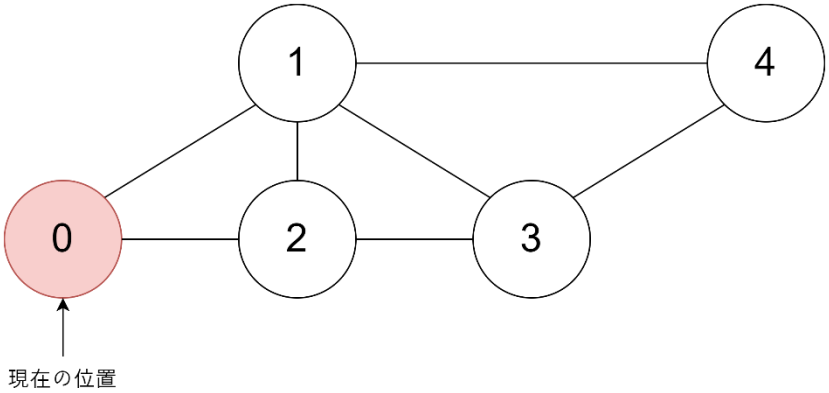


図 2。find_pathフローチャート

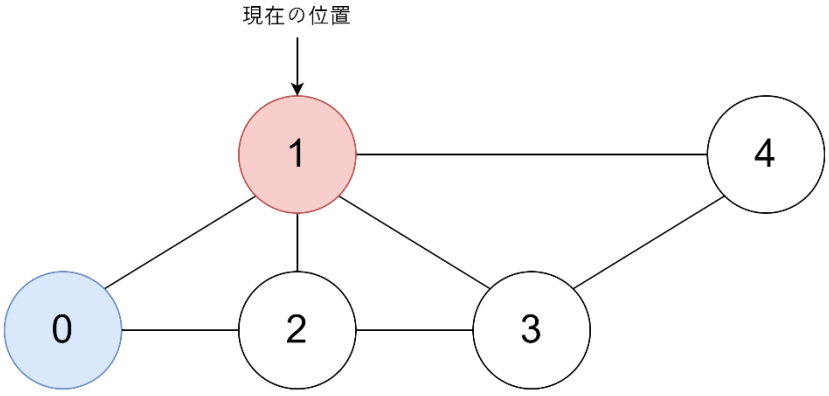
(3) アルゴリズムが正しいことの説明

A. 正当性

i. このアルゴリズムは現在の位置から次に移動できる場所を探して移動できるときに移動して現在の位置を移動した場所に移ることで0から4までの経路を探索している。そして、0から探索して4になるときに終わるようにしている。今回の課題では5個の地点が存在するため全てが繋がっていると仮定すると現在の位置から移動できる地点は四つ存在するので四つの地点を全ての地点に移動できるか確認して移動するため全ての経路を探索することができると思う。



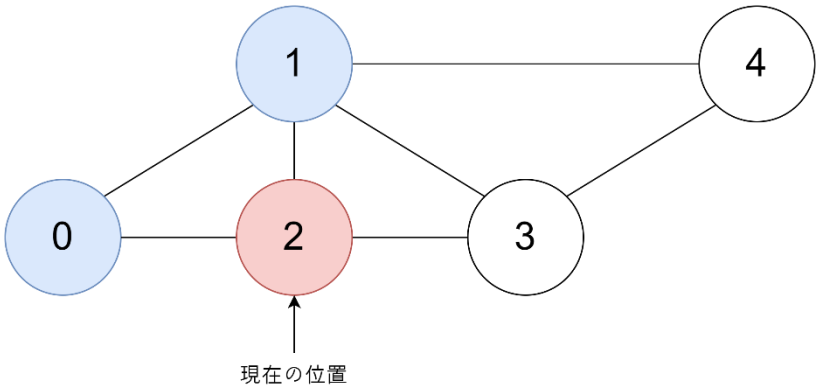
このような状態で0からスタートするので現在の位置は0である。そして、0から1～4全ての地点に行けるか確認する。1から確認するため1に行けることを確認して1に現在の位置を移動する。そして、経路配列に現在の位置を入れる。



経路配列

0	1		

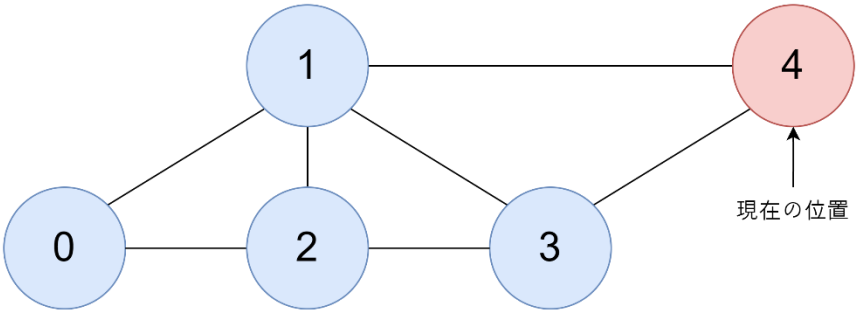
その後、１の位置から0,2,3,4の全ての位置に移動できるか確認する。この時に既に通った青の位置にはいかないようにする。すると、次の現在の位置になる位置は2であるため２に移動する。



経路配列

0	1	2		

この過程を続けると



経路配列

0	1	2	3	4

このように０から４に移動する経路を見つけることができる。現在の計算と最後の位置である４が一致するとバックトラッキングを始めて他の経路を探索することで全ての経路を探索することができる。よって、このアルゴリズムは正当な結果を出力するアルゴリズムであると考えられる。

B. 停止性

i. このアルゴリズムでループする場所は経路を出力するためのループと全ての地点を探索するためのループと全ての経路を探すための再

帰関数存在する。

1. 経路出力ループと地点探索ループ
    A. この二つのループは決められている回数に合わせてループするので回数を正しく設定すると無限ループに入ることはなく正しく停止する。経路を出力するループは全ての経路を出力するとループを脱出するために経路の長さを予め変数として設定することで正しい回数に停止する。また、地点探索ループの場合地点の個数分ループすることで無限ループに入ることなく正しく停止する。
2. 再帰関数ループ
    A. find_path関数の中では関数の中で自分の関数を呼び出しているため特定の停止条件がない場合無限ループに入る可能性を含めている。よって、呼び出された際に現在の位置が目的の位置であるかを確認することで現在の位置が目的の位置にあるときに自分の関数を呼ばないことで正しく停止すると考えられる。

(4) ソース・プログラムの説明

A. ソースコードの説明

i. 地点の個数をNUMで#defineを利用して定義する。その後、経路の図をデータ化するために２次元配列を宣言して各地点の番号をインデックスの番号にしていける地点のインデックスに１を入れていけない地点に０を入れることで経路図を２次元配列に入れた。その後、訪問したか否かを確認するcheak配列を０に初期化して宣言する。そして、経路を入れる経路配列pathを宣言する。そして、経路のインデックス変数を０で初期化して宣言する。その後、find_path関数のプロトタイプ宣言する。次に、main関数では0から4に至る経路:を出力してfind_path関数にスタート地点０と目的地点の４を入れて呼び出す。その後、return０して終了する。find_path関数ではスタート地点をpointという変数に入れて、目的地点をendに入れて現在の位置であるcheak[point]に訪問している意味の１を入れる。その後、経路配列であるpath[pathIndex]に現在の位置を入れて経路を追

加する。そして、次の経路インデックスに移動する。再帰関数の終了条件である現在位置と目的地点が一致するif(point == end)条件の時に経路を出力するために繰り返し文を利用してpath配列を出力する。もし、終了条件を満たさなかった場合全ての地点に訪問しているか確認して行けるか確認して再帰呼び出しを行う。そして、経路を見つけた場合や詰まったときに戻るためのバックトラッキングpathIndex—とcheak[point] = 0を書く。

(5)考察

A. このアルゴリズムの場合全ての経路を探索するときに詰まるまで又は目的地に到達するまで進めて戻ってきて他の経路を探す手段を選んでいる。この方法は全ての経路を完璧に探索することができる。しかし、時間的に効率的には言えない。今回の場合は五つの地点の経路を探すためで全ての経路を探すのにあまり時間がかからない。しかし、地点が多くなり経路がより複雑になる場合は一つの経路をずっと進んで詰まったときや目的地に到達したときに他の経路を探すために戻るときに多くの時間が必要である。そして、戻る過程でまた全ての場合を処理するため多くの処理時間が必要になる。よって、より効率的に全ての経路を探索するためには並列的に探索を行うことが一番効率的だと思われる。スタート地点で行ける地点全てに同時に探索を行うことである。もし、バックトラッキングアルゴリズムを利用して全ての経路を探索するとn個の地点が存在すると仮定すると、O(n!)の時間複雑度になると考えられる。しかし、並列的に探索を行うとO(n)の時間複雑度になると考えられる。n!とnの差は大きいと考えられる。よって、多くの地点や経路がある場合は並列的に探索を行うことがより効率的である。

(6)感想

A. 経路の図を２次元配列に入れる方法を考えることで日常生活の中で色んな情報をデータ化してアルゴリズムにより分析できることに気づいた。また、より効率的なアルゴリズムを考えることで既知のアルゴリズムをより効率的なアルゴリズムに発展できるか考えるようになった。