提出日：2024/6/21

プログラミング演習　第 10 回演習レポート

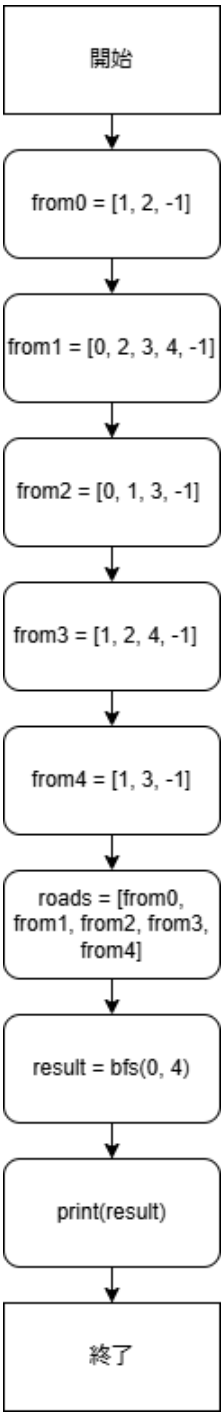
担当教員：杉本　千佳先生
所属：理工学部　数物・電子情報系学科
電子情報システム EP
学年・クラス：２年　Fe1
学籍番号：2364092
氏名：熊田　真歩

（１）課題番号：基本課題 4

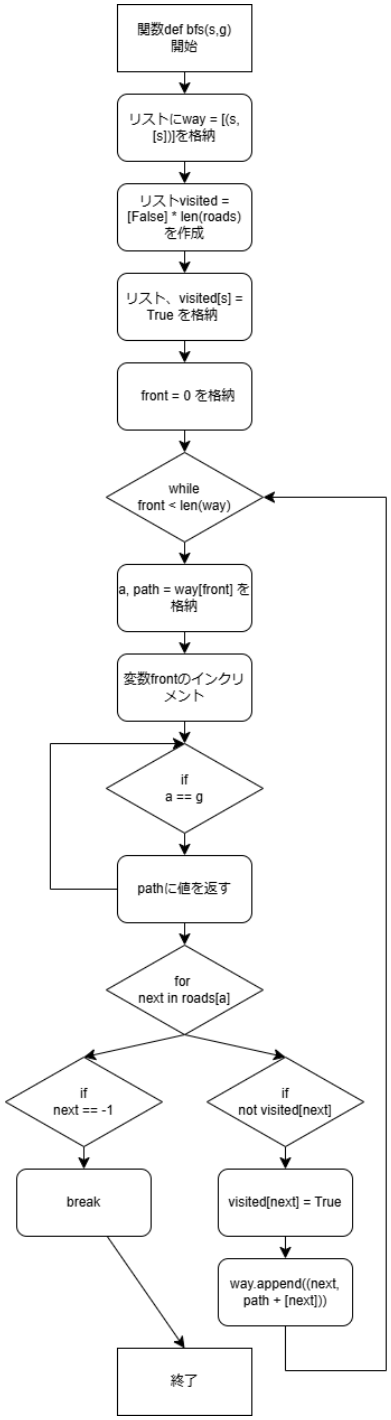
課題名：経路探索の解を求めるプログラムの改良

（２）プログラムのフローチャート

・メインフのプログラム



・bfs 関数



（３）アルゴリズムが「正しいこと」である証明

本プログラムでは、地点 0 から地点 4 まで到達するために考えられる道のりの中で最短距離を出力するというものである。ここでアルゴリズムが正しいこと次の 2 つの項目に分けて説明する。はじめにすべての入力に対してプログラムは終了するという停止性に対して、本プログラムではプログラムを実行してから入力することはなく、プログラムは停止するため、停止性はいかなる場合にも実現されていることが分かる。次にすべての入力に対して「正しい答え」を出力することについて。上でも述べた通り、本プログラムでは入力する項目はなく、常に道の最短距離である、0.1.4 と正しい答えを出力する。すなわちプログラムが出力する答えは常に所望のものである。これらより、本プログラムで採用したアルゴリズムは正しい。

（４）ソース・プログラムの説明

```
from0 = [1, 2, -1]       #0 地点から出ている道の先の地点
from1 = [0, 2, 3, 4, -1]#1 地点から出ている道の先の地点
from2 = [0, 1, 3, -1]    #2 地点から出ている道の先の地点
from3 = [1, 2, 4, -1]    #3 地点から出ている道の先の地点
from4 = [1, 3, -1]       #4 地点から出ている道の先の地点
roads = [from0, from1, from2, from3, from4] #道順を記録するファイル
def bfs(s,g):  #最短ルートを見つける関数
    way = [(s, [s])]  #(現在の地点, 経路)を変数 way に代入
    visited = [False] * len(roads)  #訪問済みの地点を記録するリスト
    visited[s] = True                #訪問済みの地点を記録
    front = 0                       #先頭位置を示す変数 front
    while front < len(way):          #way の長さ分
        a, path = way[front]         #2 変数に道リストを代入
        front += 1                   #front を 1 つづらす
        if a == g:                   #目的地に到達したら
            return path              #最短経路を返す
        for next in roads[a]:        #次の地点を探索
            if next == -1:           #ある地点からの候補の道がなくなったら
                break                #ループ終了
            if not visited[next]:    #まだ通ったことのない道の時
                visited[next] = True#道をリストに記録する
                way.append((next, path + [next]))  #最短距離リストに道を格納
```

```
result = bfs(0, 4)   #0→4 の最短ルート関数を呼び出し
print(result)        #最短距離の結果出力
```

（５）考察

本プログラムでは基本課題 3 で作成した道のりを出力するプログラムを改良し、考えられる道のりの中で最短距離であるものを出力するというプログラムである。ここで工夫点として見つかった経路より短い経路が見つかる可能性がある場合のみ調べるように効率化を行った。こうすることで、プログラムのメモリや時間が短縮でき、効率の良いプログラムとなる。

また、今回は多くのリストを活用することによって、最短距離を求めることを実現した。ここで今回は出力結果もリストを用いて出力した。また、通った道に関しては True を格納することで再び同じ道を通ることなくスタートからゴールまで通ることを実現した。最後に関数を呼び出し、結果を出力した。

（６）感想

基本課題 4 は基本課題 3 を改善するだけで、簡単だと思ったが作ってみたら少し難しかった。見かけには寄らないなと痛感した。また、見つかった経路より短い経路が見つかる可能性のみ調べ、効率化を図るという工夫が難しかった。