گزارش پروژه عقب‌گرد

درس طراحی الگوریتم

مسئله «ماشین آتش‌نشانی»

مجتبی فیاضی – 9728973

توضیحات کلی

هدف این مسئله پیدا کردن تمام راه‌های ممکن از ابتدای مسیر به محل وقوع آتش است. برای این منظور از الگوریتم dfs با کمی تغییر برای بکارگیری تکنیک عقب‌گرد استفاده شده است. روال کار الگوریتم به این صورت است که با شروع از گره اول، هر یک از گره‌های همسایه آن گره چک می‌شود، اگر امیدبخش بود (در اینجا یعنی قبلاً ویزیت نشده) همین کار به صورت بازگشتی برای آن گره تکرار می‌شود. پس از رسیدن به مقصد محتوای آرایه visited به عنوان مسیر در خروجی چاپ می‌شود.

```
def dfs(self, v, visited):
    visited.append(v)

    if v == self.goal:
        self.paths.append(visited)
        FireTruck.printPath(visited)
        visited.pop()
        return

    for neighbour in self.streetGraph[v]:
        if self.isPromising(neighbour, visited):
            self.dfs(neighbour, visited)

    visited.pop()

def findAllPaths(self, start, goal):
    visited = []
    visited.append(start)
    self.goal = goal
    for neighbour in self.streetGraph[start]:
        self.dfs(neighbour, visited)
```

تابع findAllPaths را با دانستن گره مقصد و مبدأ صدا می‌زنیم. یک نمونه خروجی برنامه بالا:

```
CASE 1:
1 2 3 4 6
1 2 3 5 6
1 2 4 3 5 6
1 2 4 6
1 3 4 6
1 3 5 6
1 3 2 4 6
There are 7 routes from the firestation to streetcorner 6
CASE 2:
1 5 2 3 4
1 5 7 8 9 6 4
1 6 4
1 6 9 8 7 5 2 3 4
1 3 2 5 7 8 9 6 4
1 3 4
1 8 7 5 2 3 4
1 8 9 6 4
There are 8 routes from the firestation to streetcorner 4
```

پیچیدگی زمانی

برای تخمین پیچیدگی زمانی الگوریتم، از روش مونته کارلو استفاده شده است. بدین منظور برای یک نمونه از مسئله، تابع estimate ده هزار بار فراخوانی شد و میانگین اینها به عنوان شاخص پیچیدگی گزارش شده است.

```python
def estimate(self, start):
    v = start
    visited = [v]
    numnodes, m, mprod = 1, 1, 1
    while m != 0:
        t = len(self.streetGraph[v])
        mprod = mprod * m
        numnodes = numnodes + mprod*t
        promisings = []
        for neighbor in self.streetGraph[v]:
            if self.isPromising(neighbor, visited):
                promisings.append(neighbor)
        m = len(promisings)
        if m != 0:
            v = promisings[int(random.random() *
                               len(promisings))]
            visited.append(v)

    return numnodes
```

نمونه خروجی تابع بالا برای دو نمونه مختلف از مسئله:

```
CASE 1:
1 2 3 4 6
1 2 3 5 6
1 2 4 3 5 6
1 2 4 6
1 3 4 6
1 3 5 6
1 3 2 4 6
estimation: 76.8608
There are 7 routes from the firestation to streetcorner 6
CASE 2:
1 5 2 3 4
1 5 7 8 9 6 4
1 6 4
1 6 9 8 7 5 2 3 4
1 3 2 5 7 8 9 6 4
1 3 4
1 8 7 5 2 3 4
1 8 9 6 4
estimation: 153.0
There are 8 routes from the firestation to streetcorner 4
```