مار و پله

ستاره روشن

چکیده — در این گزارش به حل مسئله مار پله با استفاده از کتابخانه deap پرداختیم.

۱. مدل کردن مسئله

در ابتدا بازی مارپله و مکان نردبان‌ها را در یک آرایه مشخص می‌کنیم. در آرایه پازل مقادیر هر درایه مساوی است با مکانی بر روی صفحه مارپله برای مثال از ۰ پازل همان ۱ مارپله و هر درایه مساوی است با اینکسش مگر مکان مارها یا نردبان‌ها.

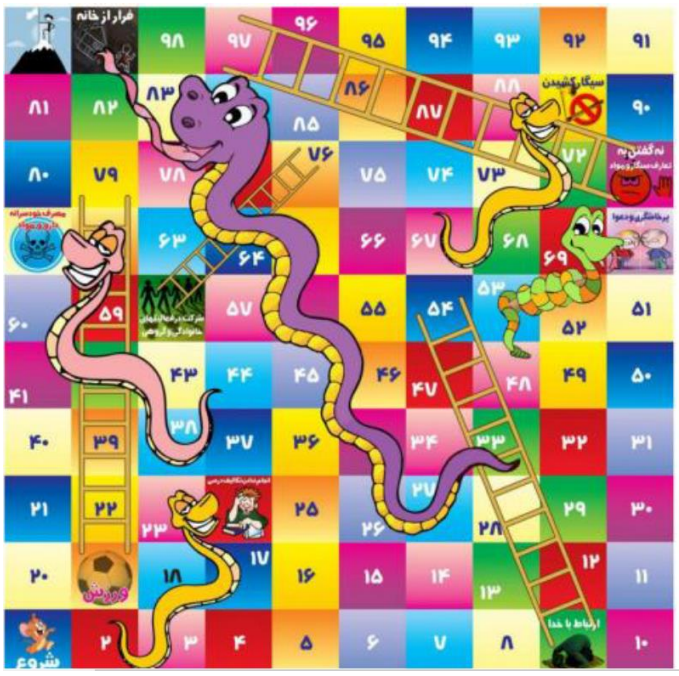
```
puzzle = np.arange(100)
```

```
puzzle [8] = 54
puzzle [23] = 1
puzzle [60] = 43
puzzle [18] = 61
puzzle [57] = 75
puzzle [98] = 32
puzzle [69] = 47
puzzle [70] = 96
puzzle [88] = 66
```

```
puzzle
```

```
rray([ 0,  1,  2,  3,  4,  5,  6,  7, 54,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 61, 19, 20, 21, 22,  1, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 75, 58, 59, 43, 61, 62, 63, 64, 65, 66, 67,
       68, 47, 96, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 66, 89, 90, 91, 92, 93, 94, 95, 96, 97, 32, 99])
```

شکل أ طراحی صفحه مارپله با توجه به شکل ب



شکل ب صفحه اصلی بازی مار پله.

در گام بعد هر کرومزوم بصورت زیر خواهد بود که شامل حداکثر ۱۶ ژن می‌باشد. مقدار تاس اول در آلل (allel) شماره ۰، تاس شماره دو در آلل شماره ۱ و به همین ترتیب.

تاس حرکت شانزده	تاس حرکت پانزده	...	تاس حرکت چهار	تاس حرکت سه	تاس حرکت دو	تاس حرکت اول

۲. تعریف عملگرها

در این قسمت ما به یک تعریف integer نیازمندیم که با آرایه numpy تعریف شده خط سوم در باکس بالایی شکل ت. همچنین fitness ماکزیمم است. در باکس پایینی در حال گرفتن هر عضو جمعیت هستیم با خط ۳ در خط ۲ محدوده هر متغیرها را تعیین میکنیم و نوع بازنمایی را. در خط ۵، اینکه جمعیت چگونه گرفته میشود است.

```
#minimize number of dice and maximize path
creator.create("Fitness", base.Fitness, weights = (1.0,))
creator.create("Individual", np.ndarray, fitness = creator.Fitness)
```

```
toolbox = base.Toolbox()
toolbox.register("attr_item", random.randint, 1,6)
toolbox.register("individual", tools.initRepeat, creator.Individual,
    toolbox.attr_item, 10)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

شکل ت. تعریف ارزیابی و جمعیت و اعضا

شل زیر تابع ارزیابی را نشان می‌دهد که برای هر individual یک بار بازی انجام می‌گردد. به عبارتی، مقادیر تاس‌ها را با توجه به تابع puzzle جمع کرده و اگر ۱۰۰ شد یا بزرگتر قبل از آنکه ۱۶ عدد باهم جمع شوند بقیه مقادیر را صفر کرده و مقدار قبل ۱۰۰ را بر می‌گرداند. در غیر اینطورت تا آخر آرایه می‌رود و بعد مقدار ارزیابی شده را برمی‌گرداند.

```
def evaluation (individual):
    result = 0.0
    for i in range (10):
        result += individual[i]
        if result >=100:
            for j in range (i+1,10):
                individual [j] = 0
            return result - individual[i],
        if puzzle [int(result)] != result:
            result = puzzle[int(result)]

    return result,
```

شکل ث تابع ارزیابی

عملگرهای crossover و جهش نیز به ترتیب uniform و swap تعریف شده‌اند. در قسمت شکل ج از tournament برای انتخاب استفاده شده است و با سایز ۱۰.

```python
toolbox.register("evaluate", evaluation)
toolbox.register("mate", crossover)
toolbox.register("mutate", mutate)
toolbox.register("select", tools.selTournament, tournsize = 10)
```

شکل ج. استفاده از toolbox برای ارزیابی، crossover، جهش و انتخاب

در قسمت crossover و جهش می‌توان از ماژول‌های آماده نیز استفاده کرد اما خود ما اینجا تعریف کردیم در

```python
def crossover (ind1, ind2):
    #uniform crossover
    temp = ind1
    for i in range (10):
        p = random.randint(0,1)
        if p == 1 :
            ind1[i] = ind2[i]
            ind2[i] = temp[i]
    return ind1, ind2
```

```python
def mutate (individual):
    #Swap mutation
    i = random.randint(0,9)
    j = random.randint(0,9)

    temp = individual[i]
    individual[i] = individual[j]
    individual[j] = temp
    return individual
```

شکل ح. باکس بالایی عملگر crossover و باکس پایینی عملگر جهش.

۳. اجرا

این قسمت شکل خ خط دوم تعداد جمعیت در هر generation تعیین می‌گردد. در خط ۹ دیگر کار تکاملی آغاز شده و درصورتی خاتمه می‌یابد که هم به ۲۰۰۰ نسل برسد و هم به ماکزیمم فیتنس که برابر ۱۰۰ است برسد. سپس عملگرهای crossover و mutation با احتمالات به ترتیب ۰.۵ و ۰.۸ اعمال خواهد شد و در آهر خط ۳۰ کامل جمعیت offspring جایگزین جمعیت قبلی خواهد شد. خط ٤٢ تعیین برنده می‌کند.

تمامی پارامترهای مورد استفاده به اختصار در جدول أ آورده شده است.

جدول أ. پارامترهای مورد استفاده

Representation	Integer
recombination	Uniform
Recombination probability	۰.۵
mutation	swap
Mutation probability	۰.۸
Parent selection	Tournament (k = ۱۰)
Survival selection	generational
Population size	۳۰۰
Termination condition	۲۰۰۰ generation and best fitness = ۹۹

```python
def main ():
    pop = toolbox.population(n = 300)
    fitnesses = list (map (toolbox.evaluate, pop))
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
    CXPB, MUTPB = 0.5, 0.8
    fits = [ind.fitness.values[0] for ind in pop]
    g = 0
    while max(fits) < 100 and g < 2000:
        g = g + 1
        print ('--Generation %i---' %g)
        offspring = toolbox.select(pop, len(pop))
        #toolbox.clone() method ensure that we don't use a reference to the i
        offspring = list(map(toolbox.clone, offspring))
                # Apply crossover and mutation on the offspring
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < CXPB:
                toolbox.mate(child1, child2)
                del child1.fitness.values
                del child2.fitness.values

        for mutant in offspring:
            if random.random() < MUTPB:
                toolbox.mutate(mutant)
                del mutant.fitness.values
        invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
        fitnesses = map(toolbox.evaluate, invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit
        pop[:] = offspring
        fits = [ind.fitness.values[0] for ind in pop]

        length = len(pop)
        mean = sum(fits) / length
        sum2 = sum(x*x for x in fits)
        std = abs(sum2 / length - mean**2)**0.5

        print("  Min %s" % min(fits))
        print("  Max %s" % max(fits))
        print("  Avg %s" % mean)
        print("  Std %s" % std)
    winner = tools.selBest(pop, k = 1)
    print (winner)
```

شکل خ. بدنه اصلی کد

یک نمونه از کرومزوم برنده در زیر آمده است:

Winner ۱: ۳, ۳, ۳, ۳, ۳, ۳, ۳, ۳, ۳, ۳
Winner ۲: ۲, ۲, ۲, ۲, ۲, ٦, ٦, ۲, ۱, ۲