بسمه تعالی

دانشکده مهندسی کامپیوتر

پروژه:

تقریب سمبولیک تابع

به کمک برنامه نویسی ژنتیک

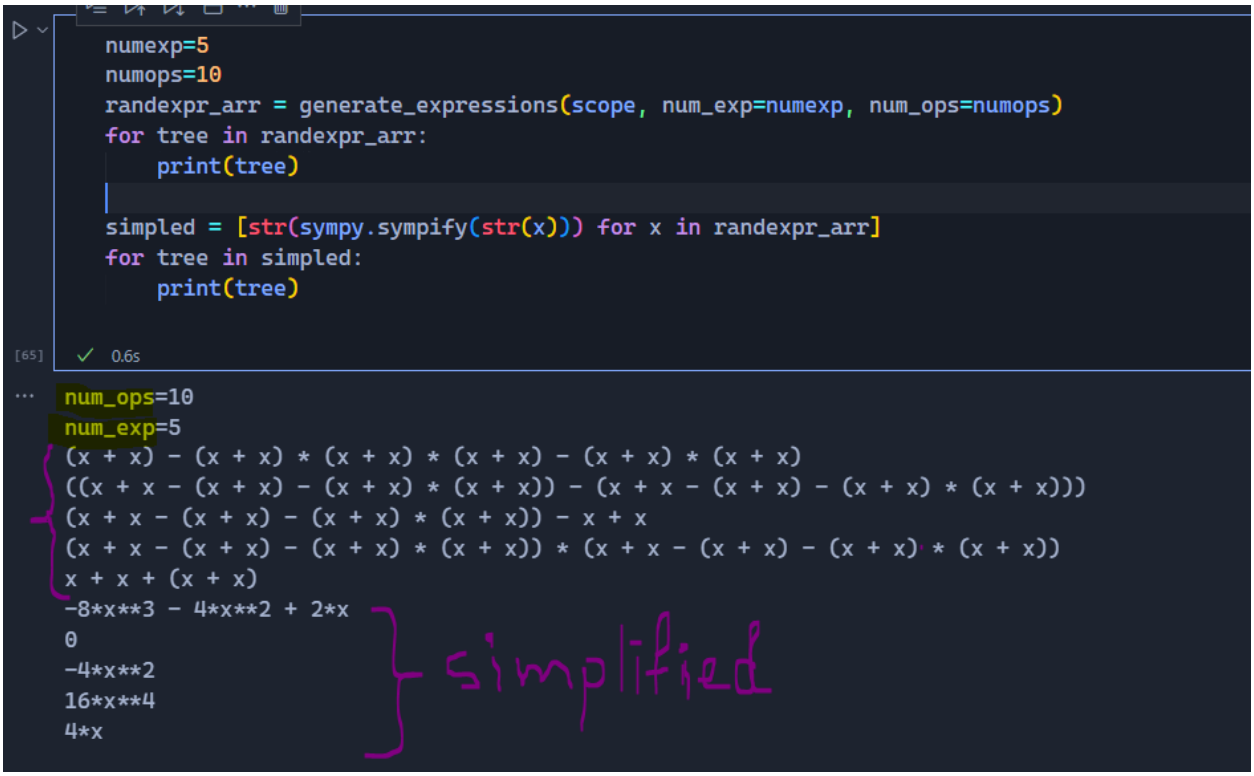
ارائه دهنده: آرین عبدالهی ثابت نژاد

شماره دانشجویی: ۹۹۵۲۱۴۴۲

مدرس: دکتر آرش عبدی

پاییز ۱۴۰۱

تابع تولید جمعیت اولیه میتواند دو پارامتر ورودی داشته باشد. اولی برای تعداد جمعیت خروجی اولیه و دومی تعداد عملگر های موجود به ازای هر عبارت.



کلاسِ *اکسپرشن تری* به صورت تقریبا کامل نوشته شده تا تمامی عملگر ها مستقل از اینکه داخل الگوریتم کار کنند یا خیر، بتوانند از رشته، تجزیه شده و به درستی داخل درخت قرار بگیرند. به همین منظور میتواند ورودی به constructor لیستی از اپراتور ها را داد و مشخص کرد که هر عملگر چپ به راست parseشود یا راست به چپ. همچنین دیکشنری از اولویت های عملگر ها و اینکه دو عملونده یا تک عملونده بودن هر کدام از عملگر ها را میتوان به آن پاس داد. همچنین میتوان با پاس دادن ریشه درخت به تابع print() نحوه پارس شدن درخت را در کنسول مشاهده کرد.(برای دیباگ)

```
op = {'+', '-', '*', '/', '^',
      'sin', 'cos'}
op_info = {'+': (2, 1), '-': (2, 1),
           '*': (2, 2), '/': (2, 2),
           '^': (2, 3),
           'sin': (1, 4), 'cos': (1, 4)}
assotiation = {'+': 'LR', '-': 'LR',
               '*': 'LR', '/': 'LR',
               '^': 'RL',
               'sin': 'RL', 'cos': 'RL'}
varchar = {'x'}
```

```
randexpr_exp = []
randexpr_tree = []
for index, expression in enumerate(randexpr_arr):
    test = Expression(expression=expression, operators=op, operators_info=op_info,
                      operators_associativity=assotiation, variables=varchar)
    randexpr_exp.append(test)
    randexpr_tree.append(test.tree())
    print(f"{index}:", randexpr_exp[index], "& ", expression)
    print(str(sympy.sympify(str(expTree2str(randexpr_tree[index])))))
    print(randexpr_tree[index])
```

parsed input

[61] ✓ 0.1s

```
0: x-x*x-x+x+x-x & x - x * x - x + x + x - x
-x**2 + 3*x
```

simplified

```
    _____
   /     \
  x     __-__
       /     \
      *       +__
     / \     /   \
    x   x   x     +__
                 /   \
                x     -
                     / \
                    x   x
```

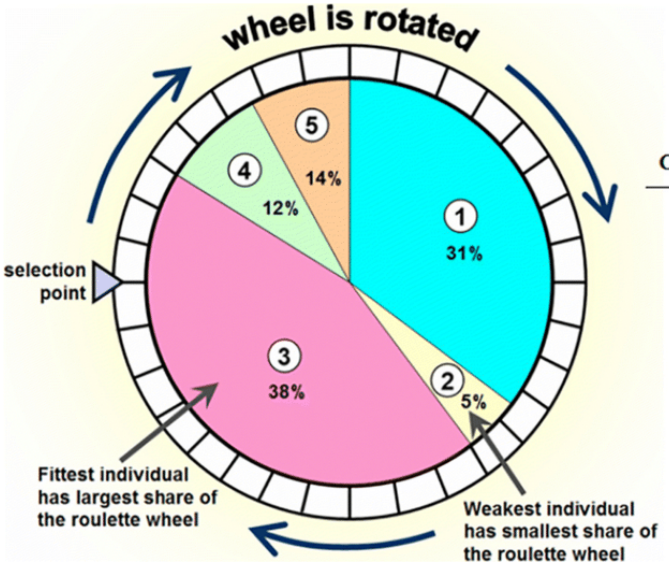
تابع شایستگی برای هر درخت با استفاده از تابع MSE یا میانگینِ مربعِ خطا طبق فرمول زیر محاسبه میشود:

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

برای محاسبه $\hat{y}$ مقادیر اولیه x در قالب یک آرایه نامپای با رنج [-5,5] به تابع بازگشتی evaluate_np() داده میشود تا $f_{predicted}(x)$ برای هر درخت از جمعیت محاسبه شود.

> اعداد بزرگتر یا کوچکتر از رنج [-5,5] تست نشده اند چرا که بخاطر overflow در تایپ های نامپای در عبارات تواندار مشکل در برنامه پیش می‌آمد.

تابع انتخاب درخت با استفاده ازroulette wheel selection نوشته شده است تا احتمال انتخاب درختانی با ذات و تابع شایستگی نزدیکتر به تابع black box بیشتر باشد و جمعیت بعد از crossover تکامل بهتری داشته باشند.



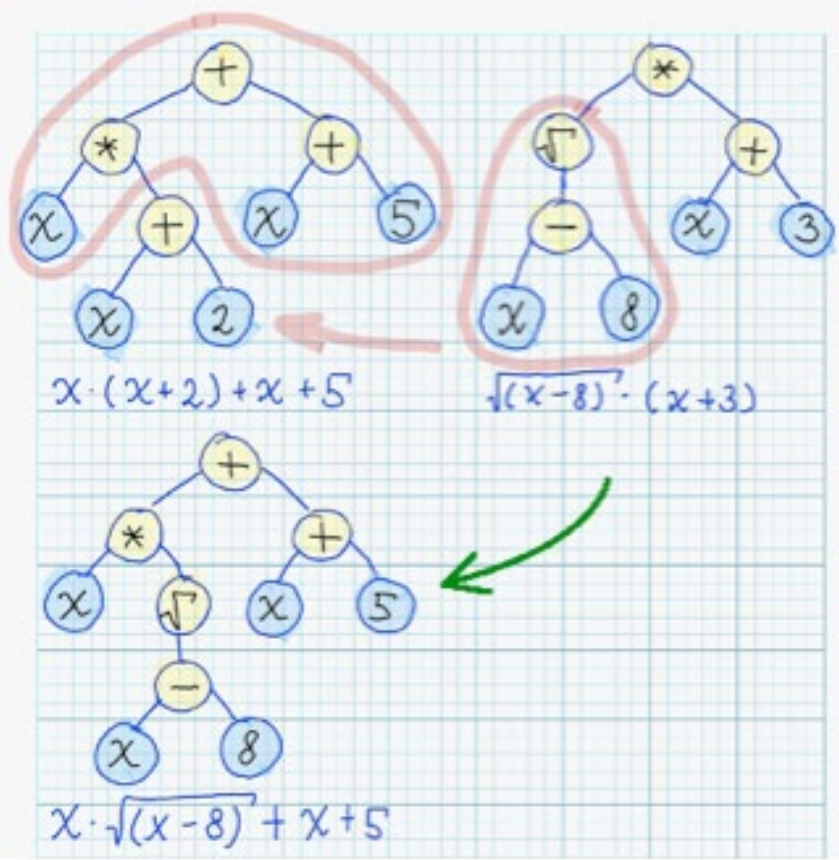
Chromosome	Fitness value
C1	31%
C2	5%
C3	38%
C4	12%
C5	14%

برای پیاده سازی این بخش ابتدا توجه به این شده که مساله minimizing میباشد پس نمودار فینتس ماکسیموم گرفته شده و سپس ماسکسیموم با قرینه تابع فیتنس نسبت به محور ایکس جمع زده شده است.

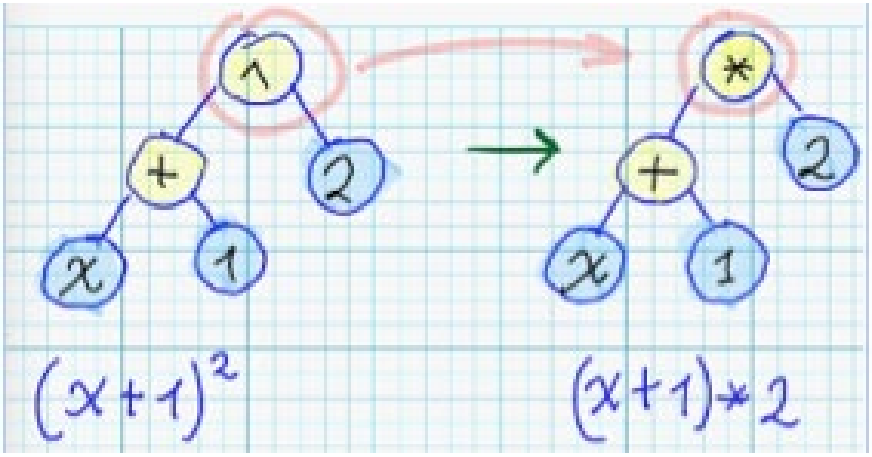
سپس با استفاده از تابع numpy.cumsum() ارایه‌ای از بخش‌های دایره درست میکنیم سپس رندوم میزنیم تا بفهمیم در کدام بخش از دایره قرار گرفته‌ایم.

> علاوه بر سلکشن خاص منظوره، 20٪ از جمعیت نسل قبل که بهترین شایستگی را داشتند دوباره به جمعیت نسل بعد اضافه میشوند تا تکامل نسل به نسل تضمین شود.

تابع CrossOver به این صورت کار میکند که با استفاده از تابع انتخاب دو ورودی به آن داده می‌شود. سپس این تابع با حرکت inorder روی تابع یک لیست از node ایجاد میکند. سپس به صورت رندوم یکی از این نود ها که برگ نیستند انتخاب میکنیم. روی درخت دوم هم همینکار را میکنیم سپس دیکشنری درونی دو آبجکت را عوض میکنیم. انگار که از پدر جای این دو را در هر دو درخت عوض کرده ایم.



تابع جهش اینگونه عمل میکند که یکی از عملگر های درخت را به صورت شانسی پیدا کرده و آن را به صورت شانسی با یک عمگر دیگر جا به جا میکند.



محدودیت های برنامه:

۱ - علمیات های محدود شده:

اکسپرشن تری میتواند عملیات های سینوس و توان و تقسیم را پارس و حتی محاسبه کند اما در الگوریتم ژنتیک صرفا به علت کمبود وقت **عبارات چند جمله ای** دیباگ شده و از کارکرد آنها اطمینان حاصل شده است. همچنین تابع تولید جمعیت اولیه در مورد تولید عبارات دارای تقسیم مشکل دارد و عباراتی مانند x/x را تولید میکند که در محاسبه سمپل x=0 برنامه دچار مشکل تقسیم به صفر میشود. عباراتی که دارای توان هستند نیز گاها دچارoverflow میشوند و موجب تولید mse منفی میشوند.

به همین علت عملگر های تست شده به جمع، تفریق و ضرب محدود شده است. هرچند که وجود چند عمل ضرب پی در پی مشابه توان و وجود چند جمع پی در پی مشابه وجود ضریب در پشت عبارات است.

۲ - تعداد ورودی محدود شده:

الگوریتم ژنتیک مشکلی با توابع چند ورودی ندارد مشکل در تابع محاسبه است. تابع محاسبه مقدار برای درخت برای پشتیبانی از چند ورودی بودن نیاز به تغییرات کوچکی دارد که با توجه به کمبود وقت هنوز پیاده‌سازی نشده است.

نمونه ورودی و خروجی و نمودار ها:

$$x^2 + 2x$$

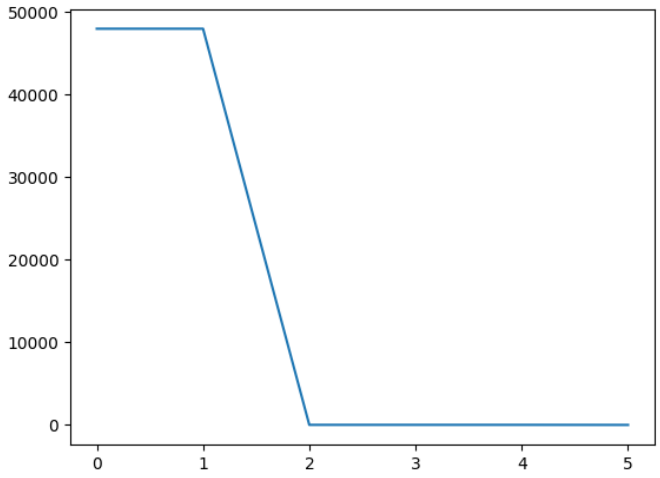
با وجود وابستگی به جمعیت اولیه، این تابع نهایتا بین 2 تا 8 نسل پیدا میشود. نمونه خروجی آن در فایل *Doc/MyDoc/sample1.txt* وجود دارد.

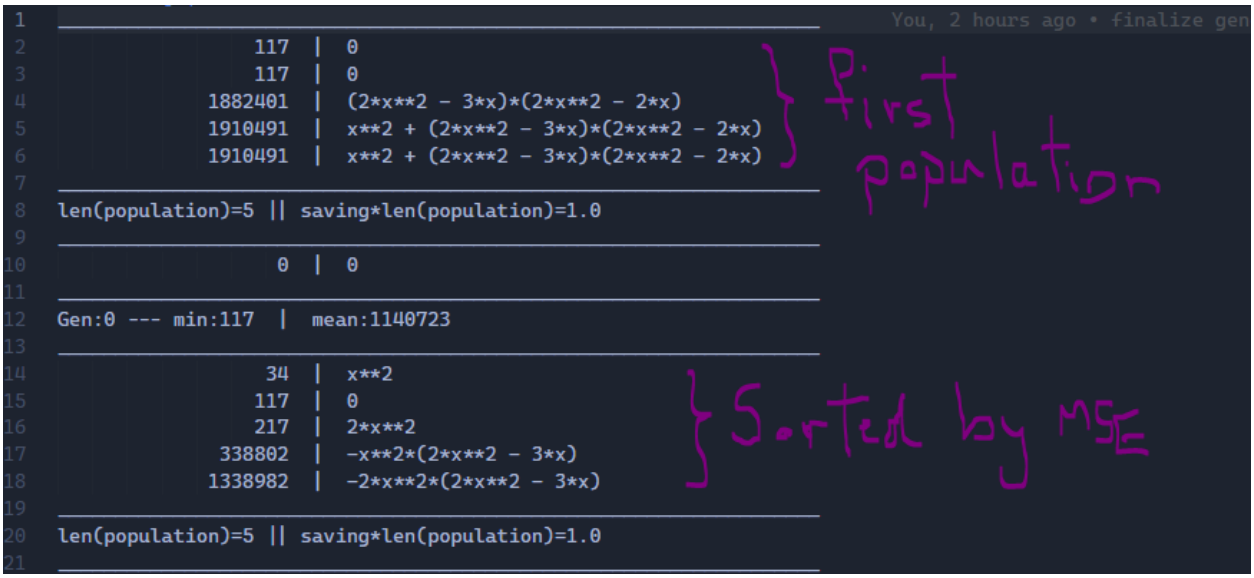
در ابتدای فایل جمعیت اولیه آمده است. سپس ساده شده ی درخت جمعیت اولیه پرینت شده و ام اس ای آن ها بصورت سورت شده در کنارشان آمده است. سپس محاسبه شده که 20 درصد نسل قبل چند عضو میشود و بهترین ها پرینت شده و به نسل بعد اضافه میشنود. سپس گزارشی از نسل داده شده و دوباره کل این روند تا رسیدن به جواب تکرار میشود.

> باید توجه داشت شاید ساده شده‌ی درخت‌ها مشابه هم باشند اما باید توجه که همانطور که گفته شد میتوانند ترکیبی از جمع و تفریق ها در درخت باشند در فرم های مختلف و نباید اینگونه پنداشته شود اگر یکسانند پس چرا حذف نمیشوند.

تنظیمات و نمودار *mse* به نسل:

با جمعیت اولیه 5 نفر و تعداد اپرواتور 10 به ازای هر عبارت




You, 2 hours ago • finalize gen
117 | 0
117 | 0
1882401 | (2*x**2 - 3*x)*(2*x**2 - 2*x)
1910491 | x**2 + (2*x**2 - 3*x)*(2*x**2 - 2*x)
1910491 | x**2 + (2*x**2 - 3*x)*(2*x**2 - 2*x)
First population
len(population)=5 || saving*len(population)=1.0
0 | 0
Gen:0 --- min:117 | mean:1140723
34 | x**2
117 | 0
217 | 2*x**2
338802 | -x**2*(2*x**2 - 3*x)
1338982 | -2*x**2*(2*x**2 - 3*x)
Sorted by MSE
len(population)=5 || saving*len(population)=1.0

$$x^5 + 3x^4 + 2x$$

این عبارت وابستگی بسیاری به جمعیت اولیه داشته و پس از چندبار ران کردن (تا جمعیت اولیه از اختلاف نژاد خوبی برخوردار شود) میتوان عبارت را بدست آورد. نمونه خروجی آن در فایل *Doc/MyDoc/sample2.txt* وجود دارد.

این فایل به خاطر *true* بودن فلگ دیباگ لاگ کمتری پرینت کرده است و تنها میتوان جمعیت نهایی را دید و پرینت کردن 300 عبارت جمعیت به ازای هر نسل معقول نیست.

در ابتدای این فایل جمعیت اولیه پرینت شده است. سپس هر نسل اطلاعاتش پرینت شده و بهترین عضو آن نسل هم پرینت شده است. در آخر جمعیت آخرین نسل و *mse* آن ها پرینت شده. سپس درخت عبارت نهایی و ساده شده عبارت حاصل از درخت آمده که معادل عبارت خواسته شده است.


```
-x**3*(-x**2 + x) - 4*x
-x**4 - x**3*(-x**2 + x) + x**2 - 5*x
-x**4 + x**3*(x**2 + 3*x) + x**2 + x
x**3*(-x**2 + x) - x**2 + 7*x
x**2 + 3*x
x**3*(-x**2 + x) - x**3 + 7*x
x**2 + x
x**3*(-x**2 + x) - x**2 + 6*x
2*x**3*(-x**2 + x) - x**3 - x**2 + 13*x
x**3*(-x**2 + x) - 2*x**3 + x**2 + 5*x
x**4 - x**3 - 4*x


Gen:0 --- min:47994  |  mean:1125897
-x**4 + x**3*(x**2 + 3*x) + x**2 + x
len(population)=50 || saving*len(population)=10.0
Gen:1 --- min:47994  |  mean:1181909
-x**4 + x**3*(x**2 + 3*x) + x**2 + x
len(population)=50 || saving*len(population)=10.0
Gen:2 --- min:8  |  mean:20965173
x**3*(x**2 + 3*x) + 3*x
len(population)=50 || saving*len(population)=10.0
Gen:3 --- min:8  |  mean:173919057
x**3*(x**2 + 3*x) + 3*x
len(population)=50 || saving*len(population)=10.0
Gen:4 --- min:8  |  mean:134147555
x**3*(x**2 + 3*x) + 3*x
_____________________________________________________
                0  |  x**3*(x**2 + 3*x) + 2*x
                8  |  x**3*(x**2 + 3*x) + 3*x
                8  |  x**3*(x**2 + 3*x) + 3*x
                8  |  x**3*(x**2 + 3*x) + 3*x
                8  |  x**3*(x**2 + 3*x) + x
               34  |  x**3*(x**2 + 3*x)
               76  |  x**3*(x**2 + 3*x) - x
               76  |  x**3*(x**2 + 3*x) + 5*x
              116  |  x**3*(x**2 + 3*x) + x**2 + 3*x
              116  |  x**3*(x**2 + 3*x) - x**2 + x
              134  |  x**3*(x**2 + 3*x) - x**2 - x
              134  |  x**3*(x**2 + 3*x) - x**2 - x
              166  |  x**3*(x**2 + 3*x) + x**2 + x
              459  |  x**3*(x**2 + 3*x) - 2*x**2 - x
```

```
           585421  |  2*x**2 + x*(-x**2 + 4*x) + x
           638871  |  x**4
          2039289  |  -x**3*(x**2 + 3*x) + x
        149102917  |  4*x**2 + x*(x**2 + 3*x)*(-x**4 + 3*x**2 - x) + 4*x
        681739126  |  -x**3*(x**4 + x) - 2*x**2 - x
     15002456239?8  |  x**3*(x**2 + 3*x) + x**3*(x**3*(x*(-x**2 + 5*x) + x) +
_____________________________________________________
Gen:final --- min:0  |  mean:30021787858
x**3*(x**2 + 3*x) + 2*x
```