HW12:ادریس لطف پوری 9717023152

Implementing PolyGraph for checking VSR

در پیاده سازی این روش به این صورت عمل شده است :

ابتدا با دریافت Schedule در قالب String به فرمت زیر:

"r2(A)r1(A)w1(C)r3(C)w1(B)r4(B)w3(A)r4(C)w2(D)r2(B)w4(A)w4(B)"

تمام read ها و write ها را مشخص میکنیم و آنه را در دو آرایه 3 بعدی به نام های read و write ذخیره میکنیم.

در سطر اول آرایه ها index عملگر در String ذخیره میشود.

در سطر دوم آرایه ها شماره Transaction ذخیره میشود.

در سطر سوم آرایه ها نیز نام دیتا ذخیره میشود.(A,B,…)

به کمک آرایه های بالا روابط بین Read ها و Write ها را پیدا میکنیم و آن را در یک ماتریس با نام PolyGraph ذخیره میکنیم. همچنین به کمک کلاس ساخته شده Graph آنرا به گراف تبدیل میکنیم.

بعد از آن قانون شماره 3 در رسم PolyGraph را اعمال میکنیم تا گراف کامل شود.

در این بین با اضافه کردن هر یال به کمک الگوریتم Graph Coloring آنرا برای وجود دور بررسی میکنیم و در صورت وجود دور به این نتیجه میرسیم که Schedule نمیتواند VSR باشد.

در آخر اگر در گراف دوری وجود نداشت به کمک متد TopologicalSort ترتیب Topologic آنرا مینویسیم.

در آخر نیز گراف را رسم و نمایش میدهیم.

#) در زمان اجرای برنامه با بستن پنجره مربوط به PolyGraph رسم شده برنامه پایان میابد.

تصویر ورودی برنامه:

```
string2="r2(A)r1(A)w1(C)r3(C)w1(B)r4(B)w3(A)r4(C)w2(D)r2(B)w4(A)w4(B)"
read = readIndexer(string2)
write = writeIndexer(string2)
print("----------------------------------------------------------------------")
print("This is the 'Read' operations in the Schedule:",read)
print()
print("----------------------------------------------------------------------")
print("This is the Writes Operation in the Schedule:",write)
print()
print("----------------------------------------------------------------------")
polygraph(read, write)

#GoodmanPole
```

تصویر خروجی برنامه:

```
----------------------------------------------------------------
This is the 'Read' operations in the Schedule: [[0, 5, 15, 25, 35, 45], [2, 1, 3, 4, 4, 2], ['A', 'A', 'C', 'B', 'C', 'B']]

----------------------------------------------------------------
This is the Writes Operation in the Schedule: [[10, 20, 30, 40, 50, 55], [1, 1, 3, 2, 4, 4], ['C', 'B', 'A', 'D', 'A', 'B']]

----------------------------------------------------------------
PolyGraph Matrix before applying rule 3:
 [[1, 1, 0, 0, 0], [0, 1, 1, 1, 1], [0, 0, 0, 0, 1], [0, 0, 0, 0, 0], [0, 0, 0, 0, 1]]
----------------------------------------------------------------
PolyGraph Matrix with its elements before applying rule 3:
 [[1, 1, 0, 0, 0], ['A', 'A', 0, 0, 0], [0, 1, 1, 1, 1], [0, 'B', 'C', 'BC', 'C'], [0, 0, 0, 0, 1], [0, 0, 0, 0, 'D'], [0, 0, 0, 0, 0], [0, 0, 0,
 0, 0], [0, 0, 0, 0, 1], [0, 0, 0, 0, 'BA']]

----------------------------------------------------------------
The PolyGraph which is saved in a matrix:
 [[1, 1, 0, 0, 0], [0, 1, 1, 1, 1], [0, 0, 1, 1, 1], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]]
----------------------------------------------------------------
The PolyGraph which is saved in a matrix with its data element:
 [[1, 1, 0, 0, 0], ['A', 'A', 0, 0, 0], [0, 1, 1, 1, 1], [0, 'B', 'C', 'BC', 'C'], [0, 0, 0, 0, 1], [0, 0, 0, 0, 'D'], [0, 0, 0, 0, 0], [0, 0, 0,
 0, 0], [0, 0, 0, 0, 1], [0, 0, 0, 0, 'BA']]

----------------------------------------------------------------
----------------------------------------------------------------
PolyGraph Info: Name:
Type: DiGraph
Number of nodes: 6
Number of edges: 11
Average in degree:   1.8333
Average out degree:   1.8333

----------------------------------------------------------------
This is the PolyGraph: defaultdict(<class 'list'>, {0: [2, 1], 1: [3, 4, 4, 2, 5, 3, 4, 4], 4: [5, 5, 1], 2: [5, 3, 4], 5: [], 3: [4]})
----------------------------------------------------------------
The Given Schedule is VSR and here is the Topological order for the Schedule:
[0, 2, 3, 4, 1, 5]
----------------------------------------------------------------
GoodmanPole

Process finished with exit code 0
```

PolyGraph رسم شده مربوط به Schedule وارد شده: