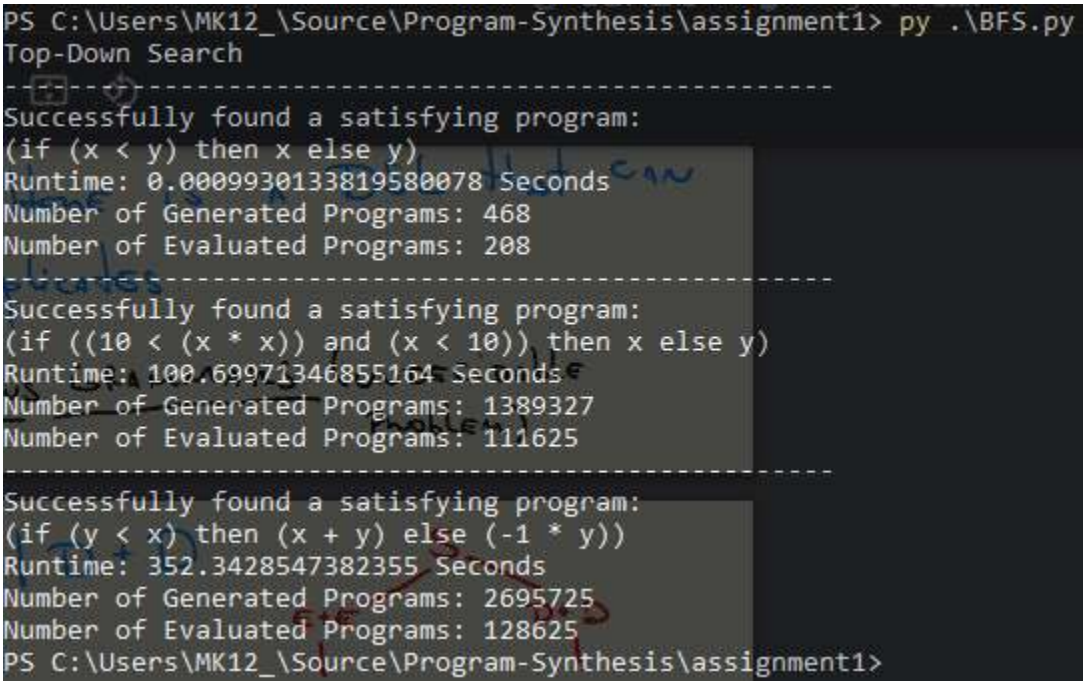
CMPUT 659 Assignment 1 Report

Zijie Tan

Ztan4@ualberta.ca

1. The codes for the BUS and BFS are in BUS.py and BFS.py, respectively.
2. BUS detects and eliminates the observational equivalent programs by executing them on the given input set.
3. The search of BUS is done by increased AST size
4. The constraints are implemented in grow() methods in BUS and children() methods in BFS.
    a. Plus, Times, Lt only accepts Num, Var, Times, Plus
    b. The condition of Ite only accepts And, Not, Lt, but the true and false cases of it accept everything
    c. And, Not only accepts Lt
5. The results of BUS and BFS on the given test cases are shown below:

```
PS C:\Users\MK12_\Source\Program-Synthesis\assignment1> py .\BUS.py
Bottom-Up Search
--------------------------------------------------
Successfully found a satisfying program:
(if (x < y) then x else y)
Runtime: 0.0 Seconds
Number of Generated Programs: 24
Number of Evaluated Programs: 21
Considering the trimming:
Number of Generated Programs: 84
Number of Evaluated Programs: 80
--------------------------------------------------
Successfully found a satisfying program:
(if ((x < 10) and (10 < (x * x))) then x else y)
Runtime: 97.5141613483429 Seconds
Number of Generated Programs: 1536
Number of Evaluated Programs: 1535
Considering the trimming:
Number of Generated Programs: 7389
Number of Evaluated Programs: 7386
--------------------------------------------------
Successfully found a satisfying program:
(if (x < y) then (-1 * y) else (x + y))
Runtime: 479.49767994880676 Seconds
Number of Generated Programs: 2607
Number of Evaluated Programs: 2032
Considering the trimming:
Number of Generated Programs: 14538
Number of Evaluated Programs: 14535
PS C:\Users\MK12_\Source\Program-Synthesis\assignment1>
```



And thus, from the above figures we obtain the following tables

Runtime (in Seconds)	Test Case 1	Test Case 2	Test Case 3
Bottom-Up Search	> 1e-10	97.5142	479.4977
Breadth-First Search	0.000993	100.6997	352.3429

generated programs	Test Case 1	Test Case 2	Test Case 3
Bottom-Up Search	84	7389	14538
Breadth-First Search	468	1389327	2695725

Evaluated programs	Test Case 1	Test Case 2	Test Case 3
Bottom-Up Search	80	7386	14535
Breadth-First Search	208	111625	128625

We can see from the above result that

a. In terms of runtime, the BUS algorithm performs better than BFS on test case 1 and 2, while BFS is significantly better on test case 3. The reason for this can be that the BUS algorithm is more efficient on smaller and better-structured search space, since its search process starts from small, simple program components and keeps eliminating candidate programs. In contrast, BFS quickly generates a large set of candidate programs, and it is generally more efficient on larger search space.
b. In terms of the number of generated and evaluated programs, BFS generates and evaluates much more programs than BUS during the search process. And in BFS, the difference between the number of generated programs and that of evaluated programs is quite large, while in BUS, such a difference is insignificant. The reasons for these differences can include:

i. BFS generates a lot of partial programs that cannot be evaluated.
ii. Hence, BFS cannot eliminate candidate programs, while BUS eliminates the observational equivalent programs by executing every encountered candidate programs.