The Anadromi Project

Hancheng Wu, Cody Steinke, Owen McLeod, Kynan Ly
Department of Computing Science, University of Alberta

PURPOSE

- To build an engaging online resource for learning computer science algorithms
- To create a platform that elevates programming skills and algorithmic thinking

OVERVIEW

- The Anadromi Project is made with the balance of theory and practical application in mind
- Students can pick up skills and learn about algorithms through solving different problems
- The curriculum builds from simple iterative functions to complex recursive functions
  - Stage 1: Linear search, selection sort, bubble sort
    - Focuses on iterative algorithms for a proper foundation
  - Stage 2: Binary search, merge sort, quicksort
    - Progresses to recursive algorithms
  - Stage 3: Iterative Fibonacci, recursive Fibonacci, dynamic Fibonacci
    - Reviews iteration and recursion, then uses dynamic programming (memoization) to improve efficency

The Anadromi Project

UNIVERSITY OF ALBERTA
FACULTY OF SCIENCE
Department of Computing Science

METHODS

LESSONS

- Every algorithm featured in Anadromi is carefully chosen and designed to be built upon one another
- All Anadromi lessons are publicly hosted on GitHub
- The structure of each lesson:
  - Introduction
  - Theory
  - PseudoCode
  - Extra Resources
  - Coding Task
  - Key Words

CODING TASKS

- Each lesson will feature a coding task that uses the online integrated development environment (IDE) Cloud9 where the student can not only write but also test their own solutions to each algorithm
- Each coding task will have their own solution with comments explaining the program line by line

Below is an example of a student testing their code using the created testing enviroment

[P] /README.m | main.py | solution.py | test_main.py

```
1  # -*- coding: utf-8 -*-
2  import sys
3
4  sys.path.insert(0, '/home/ubuntu/workspace/merge_sort') #change this to correct folder name
5  sys.path.insert(0, '/home/ubuntu/workspace/solution')
6
7  from main import merge_sort #change to proper function name
8  from solution import solved_merge_sort #change to proper function name
9  from time import sleep
10 from random import sample
11
12 class aethetics(object):
13     BLUE = '\033[94m'
14     GREEN = '\033[92m'
15     WARNING = '\033[93m'
```

```
hanchengwu:~/workspace (master) $ make test

Running Tests...
---------------------------------------------------------------------

Test failed with param of [1, 3, 2]

[1, 3, 2] does not match the answer [1, 2, 3]
---------------------------------------------------------------------

Test passed with param of [1, 2, 3]

[1, 2, 3] matches the answer [1, 2, 3]
---------------------------------------------------------------------

Test failed with param of [3, 2, 1]
```

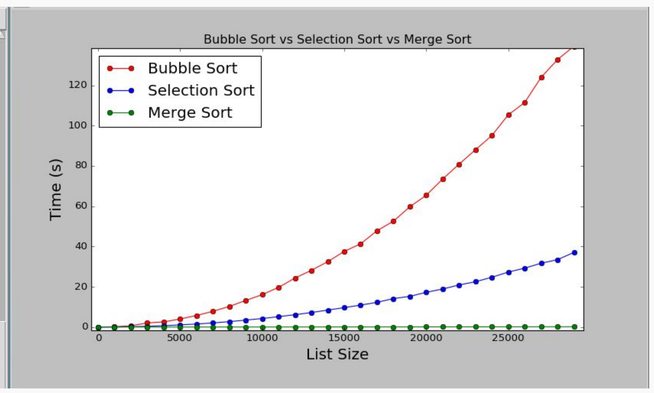
OTHER FEATURES

- Having the entire process online means minimal troubleshooting—every lesson will be a uniform experience for every learner
- PyLab is used for creating visualizations to help explain more abstract concepts, such as time complexity or algorithmic efficiency

Below is PyLab code for Anadromi and its respective visualization

File Edit Format Run Options Window Help

```
#sort graphs
time_bubble = []
time_selection = []
time_merge = []
time_quick= []

for i in ls_size:
    print(i/1000)
    ls = list(range(0, i))
    random.shuffle(ls)

    time_bubble.append(time_it_sort(bubble_sort, ls))
    time_selection.append(time_it_sort(selection_sort, ls))
    time_merge.append(time_it_sort(merge_sort, ls))
    #time_quick.append(time_it_quick(quicksort, ls, 0, len(ls) - 1))

plt.plot(ls_size, time_bubble, marker='o', color='r', label='Bubble Sort')
plt.plot(ls_size, time_selection, marker='o', color='b', label='Selection Sort')
plt.plot(ls_size, time_merge, marker='o', color='g', label='Merge Sort')
#plt.plot(ls_size, time_quick, marker='o', color='y', label='Quicksort')
#sort graphs end

plt.title('Bubble Sort vs Selection Sort vs Merge Sort')
plt.xlabel('List Size', fontsize=18)
plt.ylabel('Time (s)', fontsize=18)
plt.legend(loc = 'upper left', prop={'size': 18})
plt.show()

main()
```



RESULTS

- A project that introduces students to algorithms
- Resources created along the way: dev-cheat-sheet, journal, errors, lessons, testing-template, introduction to GitHub and C9
- Personal Results: learned more about algorithms, PyLab, makefiles, bash, and git

CONCLUSION

- Designing a curriculum for teaching and learning is a difficult but rewarding process
- The similarity in algorithms can enhance the learning experience
- Anadromi is a teaching tool for both formal classroom and informal self-taught learning

ACKNOWLEDGEMENTS

- Thank you to the classic textbook *Introduction to Algorithms* by Charles E. Leiserson, Clifford Stein, Ronald Rivest, and Thomas H. Cormen
- Thank you to my lab supervisors, Owen McLeod and Kynan Ly, and my supervisor, Cody Steinke for their support throughout my internship
- Thank you to the University of Alberta Computing Science Dept.'s High School Internship Program (HIP) and Computing Science Summer Camps for this opportunity