

Report Project 1

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.

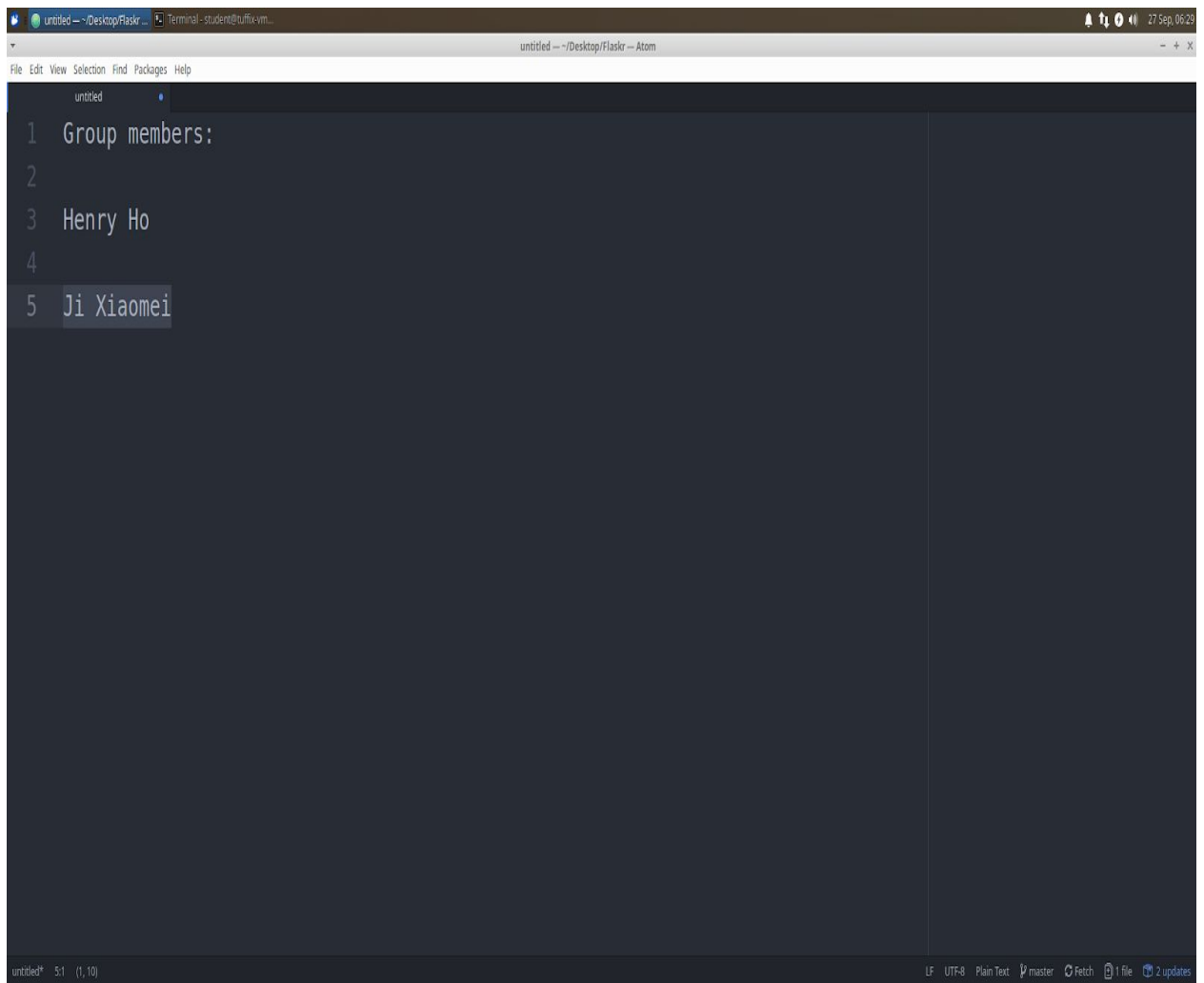
Report Project 1: implementing algorithms

CPSC 335 - Algorithm Engineering

Fall 2018

Team members: Henry Ho (vho23@csu.fullerton.edu)
 Ji Xiaomei (xiaomeiji@csu.fullerton.edu)

2. A full-screen screenshot, inside Tuffix, showing the Atom editor, with your group member names inside Atom. One way to make your names appear in Atom is to simply open your README.md.



The screenshot shows the Atom text editor interface. The main editor area displays the following text:

```
1 Group members:
2
3 Henry Ho
4
5 Ji Xiaomei
```

The status bar at the bottom indicates the file is named 'untitled*' and is at line 5, column 10. The bottom right corner shows the file encoding as 'UTF-8', the text mode as 'Plain Text', and the git status as 'master' with '1 file' and '2 updates'.

3. Two pseudocode listings, for the two algorithms.

The left-to-right algorithm:

```
def left_to_right (before):
    todo = before
    numberOfSwap = 0

    if before is not sorted and before is alternating:
        while todo is not sorted:
            for disk x in todo:
                if x is dark and the next disk x+1 is light:
                    swap x and x+1
                    numberOfSwap ++

    return sorted_disk( todo, numberOfSwap)
```

The lawnmower algorithm:

```
def lawnmower (before):
    todo = before
    numberOfSwap = 0

    if before is not sorted and before is alternating:
        while todo is not sorted:
            for x in todo from leftmost one to the right :
                if x is dark and the next disk x+1 is light:
                    swap x and x+1
                    numberOfSwap++

            for x in todo from the one before the rightmost to left:
                if x is light and the next disk x-1 is dark:
                    swap x and x-1
                    numberOfSwap++

    return sorted_disk(todo, numberOfSwap)
```

4. A brief proof argument for the time complexity of your two algorithms.

```
sorted_disks sort_left_to_right(const disk_state& before) {

    auto sorted(before);      2n
    int numSwap = 0; //number of swap time 1

    if (!before.is_sorted()) && (before.is_alternating())){ 2n + 2n
        //keep swapping if it is not sorted yet.
        while (!sorted.is_sorted()){
            for (size_t i = 0; i < before.total_count() - 1; i++){ 1
                //if the one before and the one after it is like DL then we swap. if not keep going.
                if ((sorted.get(i) == DISK_DARK) && (sorted.get(i + 1) == DISK_LIGHT)){ 2
                    sorted.swap(i); 1
                    numSwap++; 1
                }
            }
        }
    }

    return sorted_disks(sorted, numSwap); 2n
}

// Algorithm that sorts disks using the lawnmower algorithm.
sorted_disks sort_lawnmower(const disk_state& before) {

    auto sorted(before); 2n
    int numSwap = 0; //number of swap time 1

    if (!before.is_sorted()) && (before.is_alternating())){ 2n + 2n
        //keep swapping if it is sorted yet.
        while (!sorted.is_sorted()){
            size_t i = 0; 1
            for (; i < before.total_count() - 1; i++){ 1
                //if the one before and the one after it is like DL then we swap.
                if ((sorted.get(i) == DISK_DARK) && (sorted.get(i + 1) == DISK_LIGHT)){ 2
                    sorted.swap(i); 1
                    numSwap++; 1
                }
            }

            //start from the disk before the rightmost disk to the left
            for (size_t j = i-1; j > before.total_count() - i - 1; j--){ 1
                //if the one before and the one after it is like DL then we swap. if not keep going.
                if ((sorted.get(j) == DISK_DARK) && (sorted.get(j + 1) == DISK_LIGHT)){ 2
                    sorted.swap(j); 1
                    numSwap++; 1
                }
            }
        }
    }

    return sorted_disks(sorted, numSwap); 2n
}
```

The left-to-right algorithm:

$$\begin{aligned}T(n) &= 2n + 1 + 4n + (2n)(2n)(1 + 2 + 1+1) + 2n \\ &= 20n^2 + 8n + 1\end{aligned}$$

$O(T(n)) = O(\max(20n^2, 8n, 1)) \rightarrow$ belongs to $O(n^2)$ by dominated term and dropping constant factor

The lawnmower algorithm:

$$\begin{aligned}T(n) &= 2n + 1 + 4n + (2n)[1+(2n)(1 + 2 + 1+1) + (2n)(1 + 2 + 1+1)] + 2n \\ &= 8n+1+(2n)[1 +10n +10n] = 8n +1 + 2n[20n +1] \\ &= 40n^2 +10n +1\end{aligned}$$

$O(T(n)) = O(\max(40n^2, 10n, 1)) \rightarrow$ belongs to $O(n^2)$ by dominated term and dropping constant factor