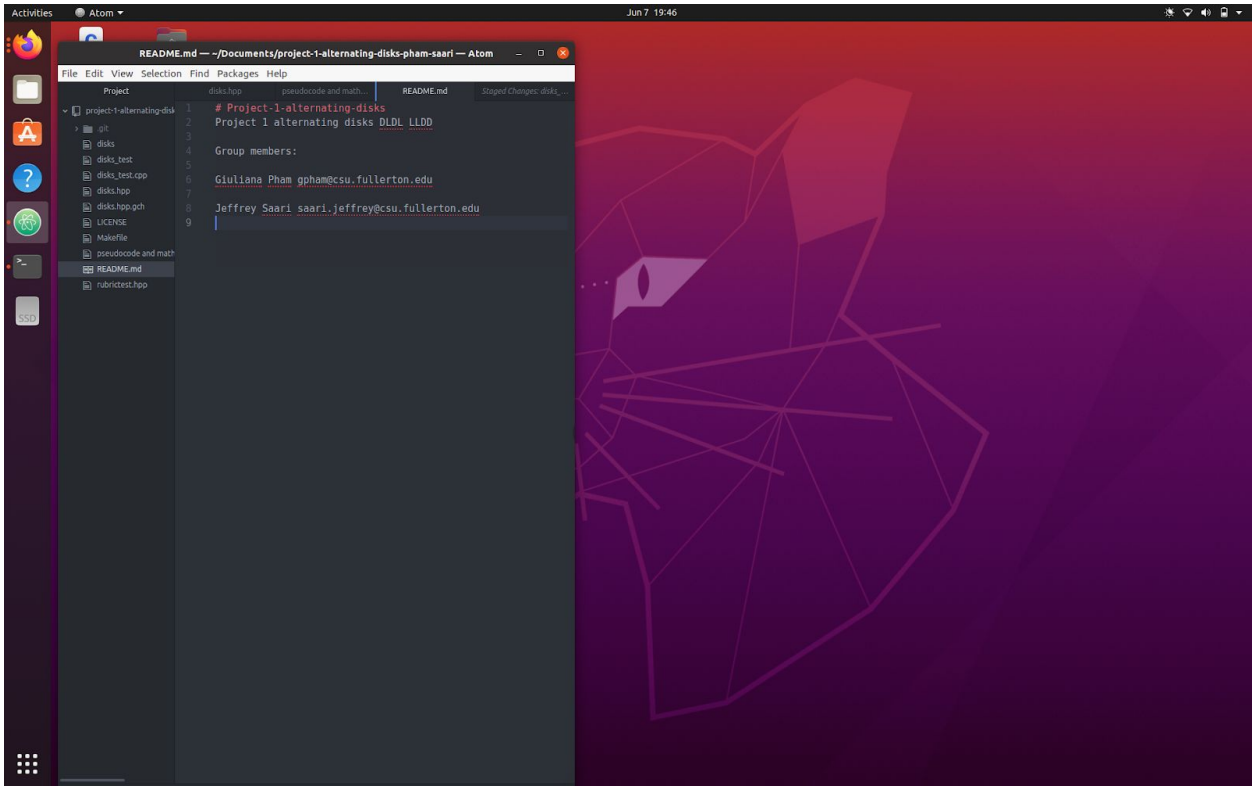
Giuliana Pham gpham@csu.fullerton.edu
Jeffrey Saari saari.jeffrey@csu.fullerton.edu

Project 1 Submission



Left to Right algorithm:
Pseudocode:

```
sorted_disks sort_left_to_right(const disk_state& before) {
  unsigned swaps = 0;
  disk_state sorted = before;
  unsigned n = total_count()/2;
  for (int i=0; i<n; i++) {
          for (int j=i; j<2*n-i-1; j++) {
                  if (sorted[j]==DISK_DARK && sorted[j+1]==DISK_LIGHT) {
                  swap(sorted[j], sorted[j+1]);
                  swaps = swaps + 1;
                  }
          }
  }
  return sorted_disks(sorted, swaps);
}
```

Mathematical Analysis:

```
sorted_disks sort_left_to_right(const disk_state& before) {
  unsigned swaps = 0;                                    //1
  disk_state sorted = before;                            //1
  unsigned n = total_count()/2;                          //3
  for (int i=0; i<n; i++) {                              //n+1 times
        for (int j=0; j<2*n; j++) {                      //2n+1 times
              if (sorted[j]==DISK_DARK && sorted[j+1]==DISK_LIGHT) {//6
              swap(sorted[j], sorted[j+1]);                  //4
              swaps = swaps + 1;                             //2
              }
        }
  }
  return sorted_disks(sorted, swaps);
}
```

S.C. = 1 + 1 + 3 + (n+1)(2n+1)(12) = 5 + (n+1)(2n+1)(12)
O(5 + (n+1)(2n+1)(12))
ignoring additive constants = O((n)(2n)(12))
ignoring multiplicative constants = O((n)(n)) = O(n^2)

Lawnmower Algorithm:
Pseudocode:

```
sorted_disks sort_lawnmower(const disk_state& before) {
  unsigned swaps = 0;
  disk_state sorted = before;
  unsigned n = sorted.total_count()/2;
  size_t ceiling = ceil(n/2);
  for (size_t i=0; i<ceiling+1; i++){
        for (size_t j=0; j<2*n-i-1; j++){
              if (sorted[j]==DISK_DARK && sorted[j+1]==DISK_LIGHT) {
                    sorted.swap(j);
                    swaps++;
              }
        }
        for (size_t k=2*n-i-1; k>0; k--){
              if (sorted[k]==DISK_LIGHT && sorted[k-1]==DISK_DARK) {
                    sorted.swap(k-1);
                    swaps++;
               }
        }
```

```
  }
  return sorted_disks(sorted, swaps);
}
```

Mathematical Analysis:

```
sorted_disks sort_lawnmower(const disk_state& before) {
  unsigned swaps = 0;                                                          //1
  disk_state sorted = before;                                                  //1
  unsigned n = sorted.total_count()/2;                                         //3
  size_t ceiling = ceil(n/2);                                                  //3
  for (size_t i=0; i<ceiling+1; i++){                          //ceiling(n/2)+1 times
        for (size_t j=0; j<2*n-i-1; j++){                      //(2n-i-1)+1 = 2n-i times
              if (sorted.get(j)==DISK_DARK && sorted.get(j+1)==DISK_LIGHT) {   //6
                    sorted.swap(j);                                            //1
                    swaps++;                                                   //2
              }
        }
        for (size_t k=2*n-i-1; k>0; k--){                  //2n-i-1+1 = 2n-i times
              if (sorted.get(k)==DISK_LIGHT && sorted.get(k-1)==DISK_DARK) {   //6
                    sorted.swap(k-1);                                          //2
                    swaps++;                                                   //2
              }
        }
  }
  return sorted_disks(sorted, swaps);
}
```

S.C. = 1 + 1 + 3 +3 + (ceiling(n/2)+1)*((2n-i)*(6 + 1 + 2) + (2n-i)*(6 + 2+ 2))
= 8 + (ceiling(n/2)+1)*(18(2n-i))
O(8 + (ceiling(n/2)+1)*(18(2n-i)))
Ignoring additive constants = O((ceiling(n/2))*(19(2n)))
Ignoring ceiling = O((n/2)*(19(2n)))
Ignoring multiplicative constants = O((n)*(n)) = O(n^2)