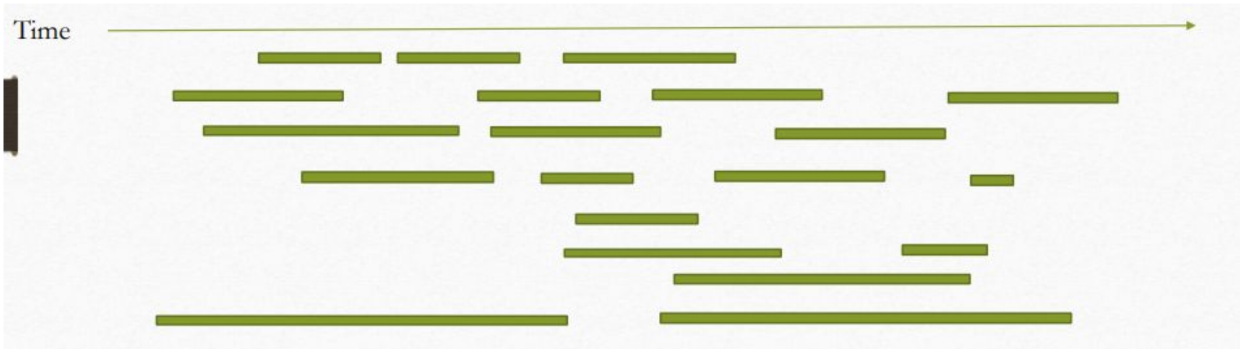
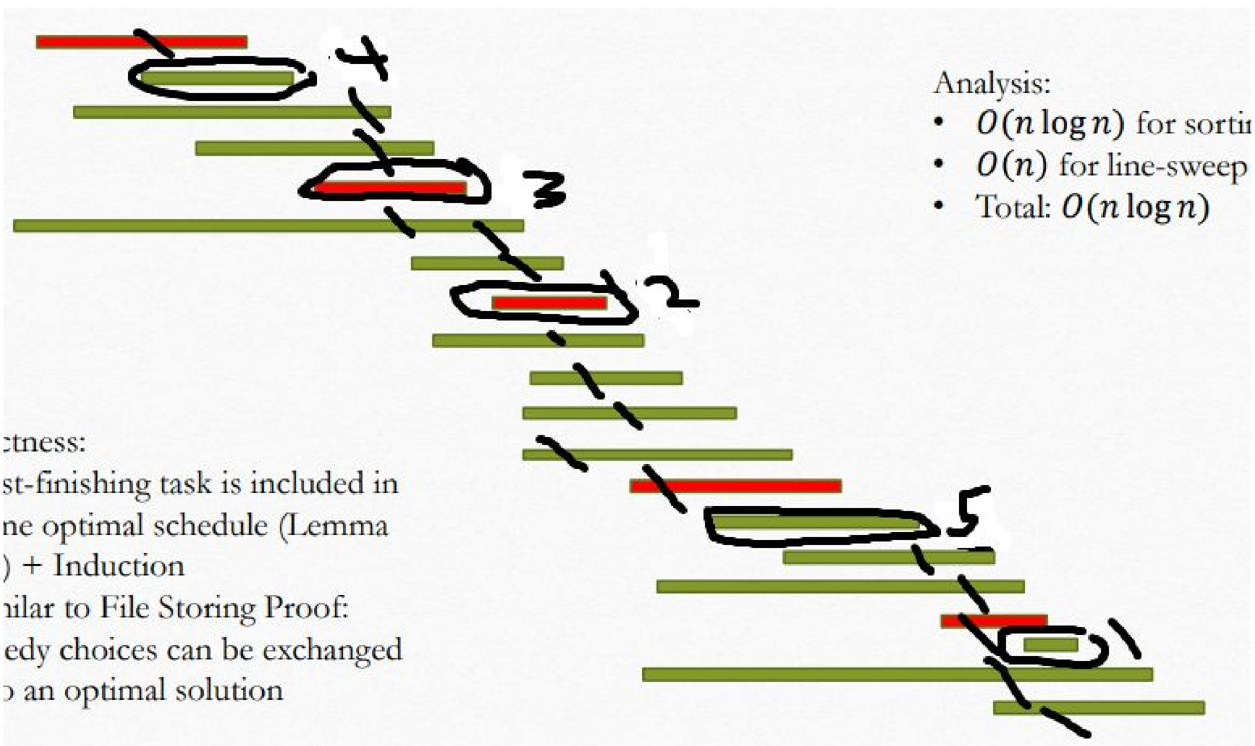
Carson Sklare

2. Using example given in lecture, shown optimal strategy gives



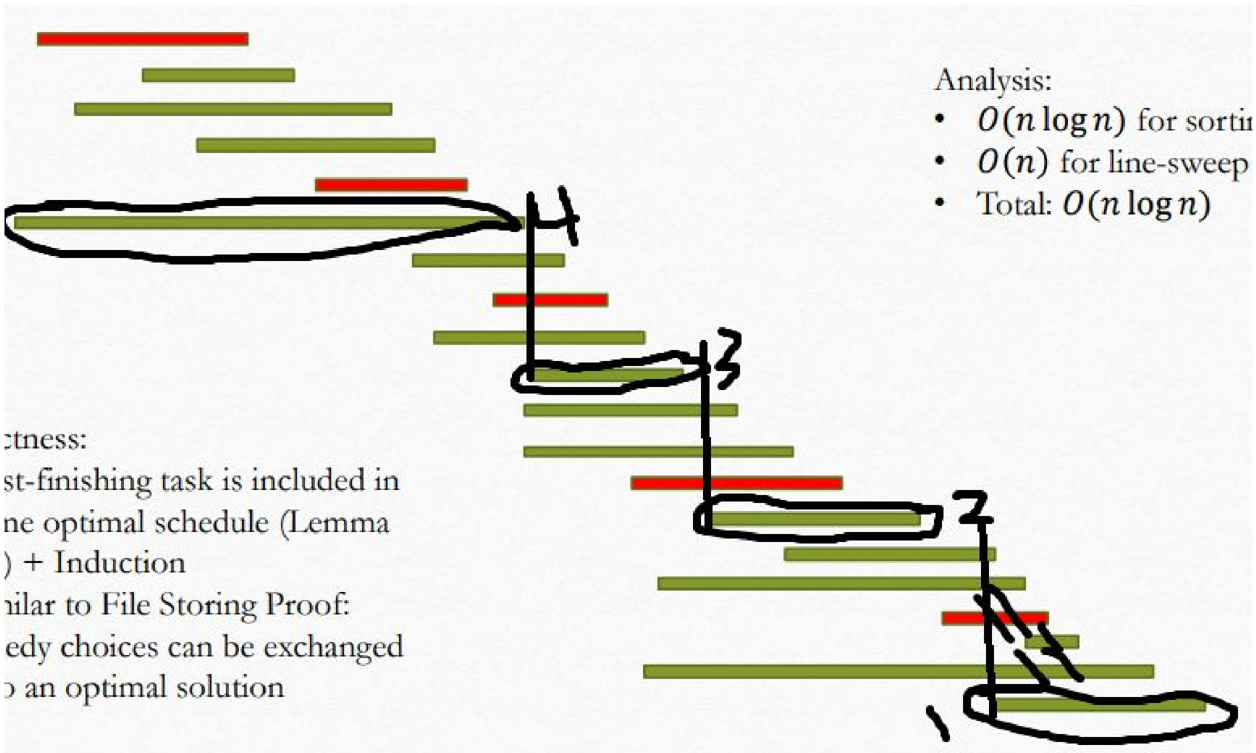
2a. [Pick Shortest] Choose the shortest task (that is, finish time-start time is as small as possible), discard all conflicting tasks, then recurse.

In this example I used the sorted version that goes from start to end and was able to get 5 tasks done, fifth task was shown after going through all the small tasks and finding a non conflicting task. I see this as a greedy strategy.



2b.[Pick Last Starter] Choose the task that starts last, discard all conflicting tasks, then recurse.

For this method I used a similar tactic of using a sorted list, I used lines as a guide to see when I can mark down the next task. In this example I was able to get 4 tasks. I found it not optimal since the fourth task was so large that it conflicted with the rest of the list. Because of that, if more larger tasks are introduced in a calendar, this method would be very inefficient.



A: Y > X > Z X: A > B > C
B: Z > X > Y Y: A > C > B
C: Y > X > Z Z: C > A > B

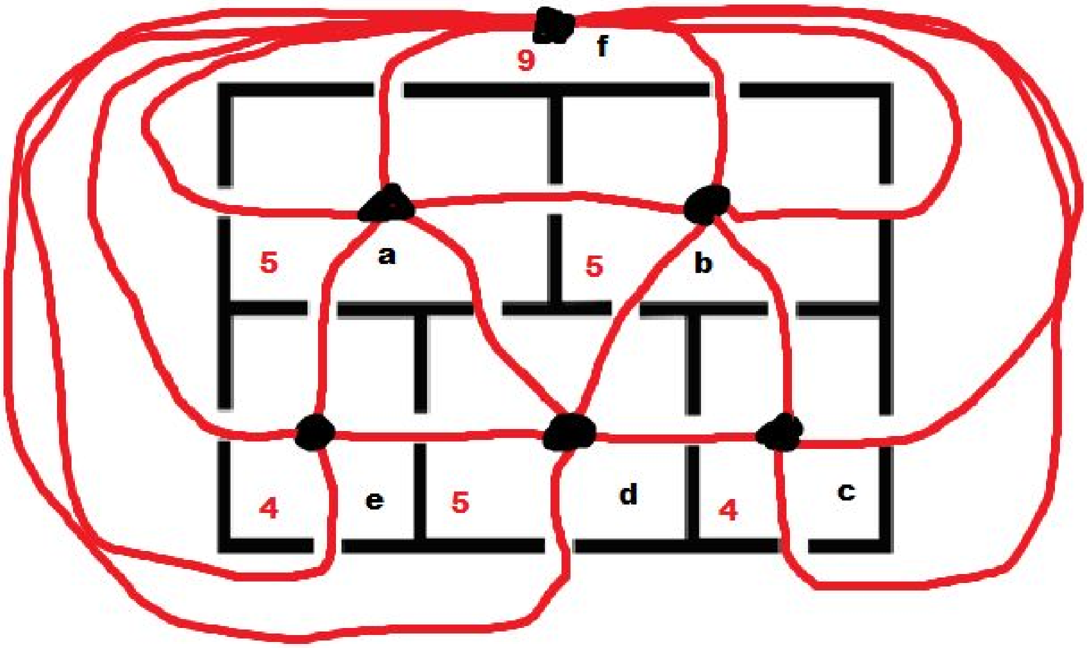
3a [2pt] The matching AX, BY, CZ is not stable, find an unstable pair for this matching.

AX, CZ, A and Y prefere each other more and C and Z prefer each other more

b. Run the Gale-Shapley Algorithm

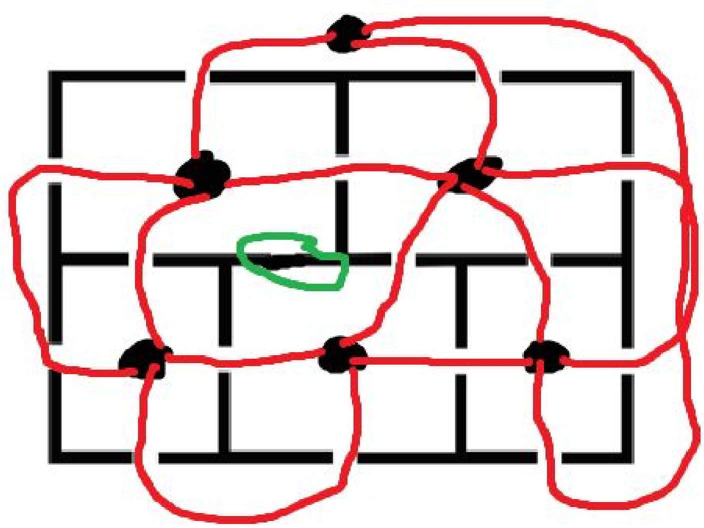
1. **X offers job to A, A accepts**
    a. **[AX]**
2. **Y offers job to A, A accepts new offer**
    a. **[AY]**
3. **X offers job to B, B accepts offer**
    a. **[AY, BX]**
4. **Z offers job to C, C accepts offer**
    a. **[AY, BX, CZ]**

4a. Picture (red text = degree, black text = vertices, red drawing = edges connecting)

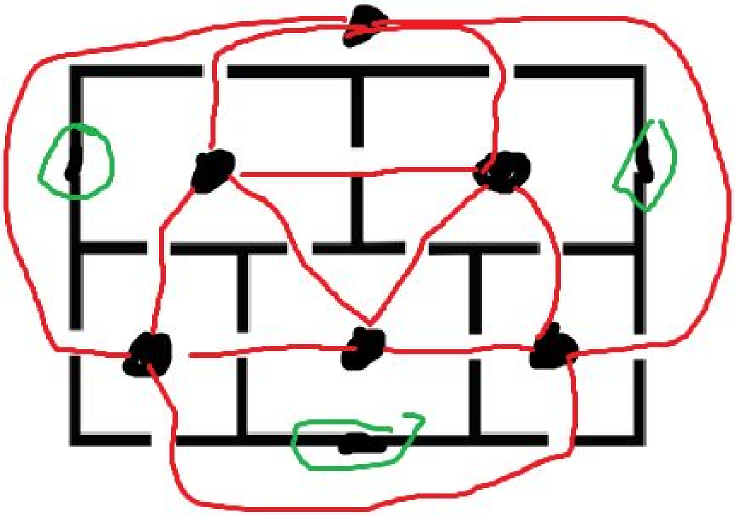


4b. **There is no euler tour because there are more than 2 vertices with odd degrees. There is no euler circuit because not all the vertices are even**

4c. **An example I used was locking one door on the top right of verticy d, this allowed there to be even degrees for both a and d, so there would only be odd degrees for f and b. I was able to use each door once shown below.**



4d. **By closing 3 doors, one on the bottom of d, one on the left of a, and one on the left of b, I was able to make all the degrees even. I was then able to make a euler circuit shown below.**



5. Code

```
def numberMaze(l):
    lbag =  [(0,0)]
    marked= []
    while len(lbag) > 0:
        (x,y) = lbag.pop()
        s = l[x][y]
        marked.append((x,y))
        for (x2,y2) in [(-s,0),(0,s),(s,0),(0,-s)]:
            x3 = x+x2
            y3 = y+y2
            if (x3,y3) not in marked and 0 <= x3 < len(l) and 0 <= y3 < len(l):
                marked.append((x3,y3))
                lbag.append((x3,y3))
    if((len(l)-1,len(l)-1) in marked):
        return True
    else:
        return False
```

Test runs (l is original working one, l2 is the changed one)

```
-------------------------------------------------------- RESTART: C:/Users/Cal
>>> l = [[3,5,7,4,6],[5,3,1,5,3],[2,8,3,1,4],[4,5,7,2,3],[3,1,3,2,0]]
>>> numberMaze(l)
True
>>> l2 = [[1,5,7,4,6], [5,3,1,5,3], [2,8,3,1,4], [4,5,7,2,3], [3,1,3,2,0]]
>>> numberMaze(l2)
False
```