1. 1. Sort the jobs by ascending deadline. For loop i as time, if i is past deadline then add its penalty
   2. Pseudo

penalty = 0

arr = [[4,7], [1,5], [2,3], [1,10], [4,9]]

mergeSort(arr) # basic merge (nlogn), ascending deadline

for i in len(arr):

if arr[i][0] <= i: # if deadline is below time, add penalty

penalty += arr[i][1]

print(penalty)

* 1. Merge sort + for loop = nlogn + n = O(nlogn)

1. Let be the set of activities that end before activity starts.

Consider any non-empty subproblem with activity having the latest start time. Then included in some maximum-size subset of mutually compatible activities of .

Then the following two conditions must hold.

1. is used in an optimal subset of
2. = leaving as the only subproblem, meaning that the greedy solution produces an optimal solution.

Consider any non-empty subproblem with activity having the latest start time. Then included in some maximum-size subset of mutually compatible activities of .

Let be an optimal solution for and be the activity in with the latest start time.

If = then the condition holds

If then construct ’ = - {} U {}

Since ’ is still optimal

The activities in are disjointed since . Since =|’|, we conclude that is a maximum-size subset of mutually compatible activities for and include .

1. schedule.py
   1. Get ints from file into an array, pop and loop master array to segment into sets by set. Merge sort current working sets by ascending start time, reverse array. Loop through adding the job number of the set whose end time is <= to previous start time. Reverse output array, output.
   2. Pseudo

def main():

arr = [] # master array of file ints

with open('act.txt', 'r') as f: # get file as master array

while True:

line = f.readline() # get line from file

if(not line): # check if line is eof

break

for i in line.split(): # get chars between space

arr.append(int(i))

iter = 1

while(arr):

N = arr.pop(0) # get number of sets

sets = [] # array of sets, activity number, start time, end time

for n in range(N):

tmp = []

for n in range(0, 3):

tmp.append(arr.pop(0)) # get the three number of a set

sets.append(tmp)

mergeSort(sets) # basic merge (nlogn), ascending start time

sets.reverse()

ar = [] # get picked jobs

ar.append(sets[0][0]) # take first

num = sets[0][1] # start of last job

for i in range(1, len(sets)):

if(sets[i][2] <= num): # if deadline is less/equal to last start

ar.append(sets[i][0]) # add to picked

num = sets[i][1] # get new start

sets = ar

sets.reverse()

print("Set", iter)

print("Number of activities selected =", len(sets))

print(sets, "\n")

iter += 1

main()

* 1. Running time = Merge sort + for loop = nlogn + n = O(nlogn)