interval-scheduling

January 28, 2026

[1]:
```
import random
```

[9]:
```
random.sample([1,2,3,4,5], 2)
```

[9]:
```
[2, 1]
```

[2]:
```
# for this test, we'll model start times and end times as integers between 0␣
 ↪and 100
def random_request():
    return sorted(random.sample(range(100),2))
```

[3]:
```
def make_requests(n):
    # output: n random requests

    # longer way to do this
    # requests = []
    # for i in range(n):
    #     requests.append(random_request())
    # return requests

    # faster
    return [random_request() for i in range(n)]
```

[4]:
```
make_requests(5)
```

[4]:
```
[[52, 53], [16, 53], [7, 47], [57, 78], [35, 51]]
```

[5]:
```
def plot_requests(requests):
    # input: a list of requests
    # output: none
    # print an ascii plot of the requests
    for r in sorted(requests, key=lambda x: x[1]):
        print(" "*(r[0]) + "-"*(r[1]-r[0]))
```

[6]:
```
R = make_requests(5)
print(R)
plot_requests(R)
```

```
[[27, 67], [18, 90], [24, 35], [60, 73], [1, 69]]
                          -----------
                             ---------------------------------------
 ---------------------------------------------------------------------
                                                                  -------------
-----------------------------------------------------------------------------
```

[8]:
```
sr = sorted(R, key=lambda x : x[1])
print(sr)
```

```
[[24, 35], [27, 67], [1, 69], [60, 73], [18, 90]]
```

[11]:
```
def greedy_solution(requests):
    # input: a list of requests
    # output: a list of non-overlapping requests representing our solution

    # sort by earliest end time
    sorted_requests = sorted(requests, key=lambda x : x[1])

    solution = []
    # two lines
    # req = sorted_requests.pop(0)
    # solution.append(req)
    solution.append(sorted_requests.pop(0))

    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        # request = [start_time, end_time]
        if request[0] >= solution[-1][1]:
            # request does not overlap the last meeting in solution
            solution.append(request)

    return solution
```

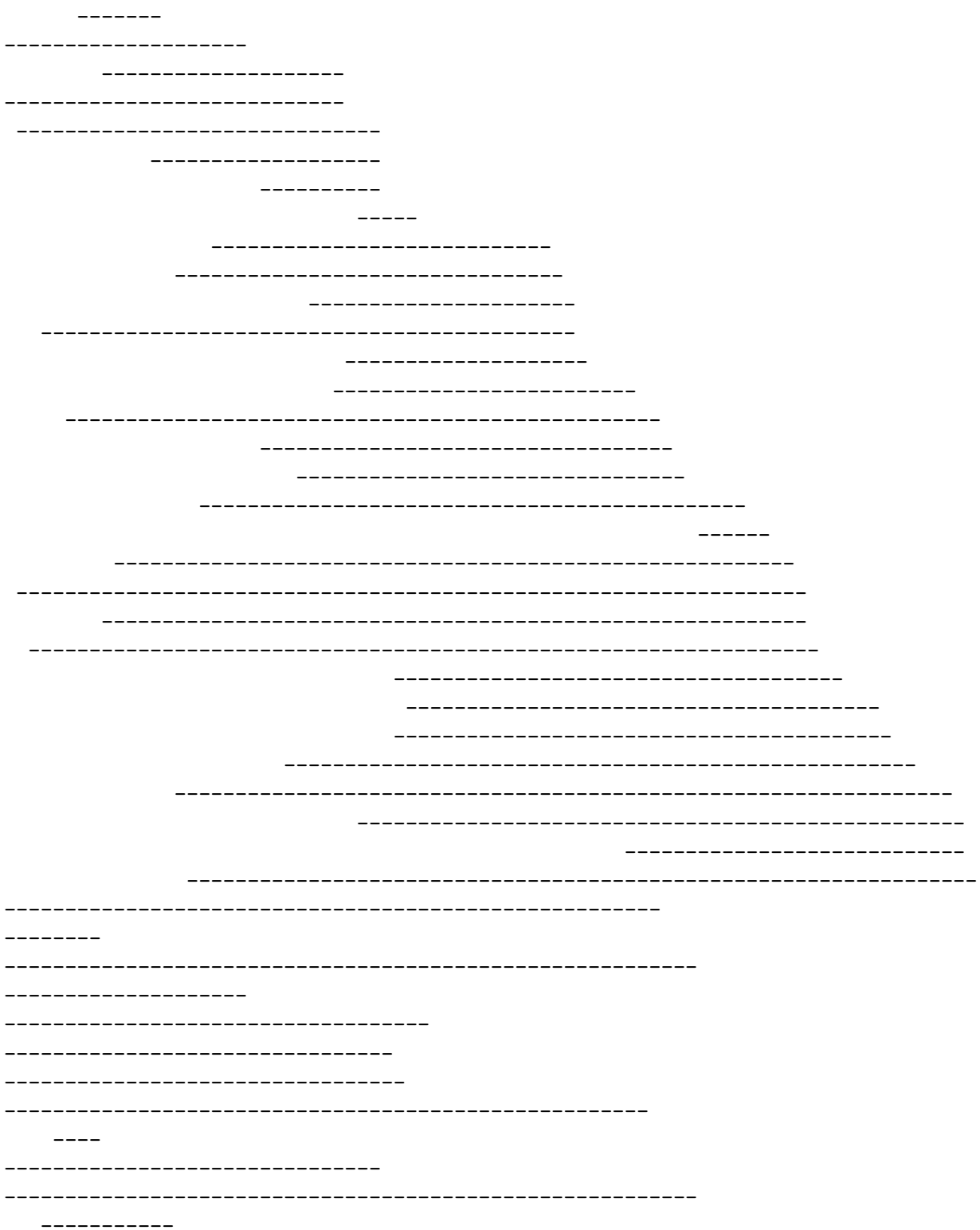
[22]:
```
requests = make_requests(50)
greedy_sol = greedy_solution(requests)
```

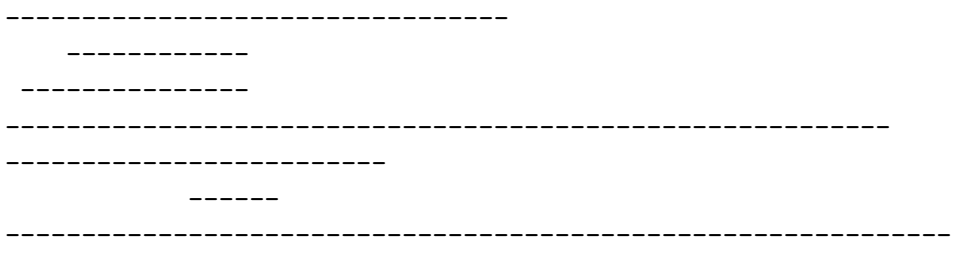
[23]:
```
print(requests)
print("\n")
print(greedy_sol)
```

```
[[1, 31], [23, 75], [0, 20], [65, 85], [60, 91], [40, 98], [33, 72], [29, 79],
[15, 80], [6, 13], [9, 65], [21, 55], [57, 63], [16, 61], [8, 28], [1, 66], [51,
79], [84, 96], [54, 87], [17, 45], [83, 94], [5, 54], [73, 98], [92, 98], [62,
95], [35, 92], [36, 98], [25, 47], [81, 96], [3, 47], [29, 34], [35, 88], [32,
73], [27, 52], [8, 66], [54, 86], [12, 31], [84, 88], [0, 28], [28, 48], [2,
67], [14, 78], [74, 82], [32, 69], [50, 85], [26, 83], [24, 56], [21, 31], [14,
46], [27, 81]]
```

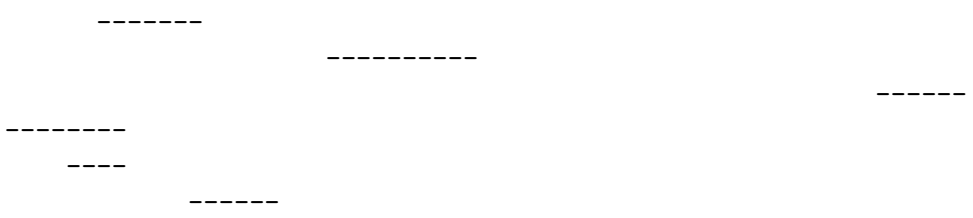
```
[[6, 13], [21, 31], [57, 63], [74, 82], [84, 88], [92, 98]]
```

[24]:
```
plot_requests(requests)
```





[25]:
```
plot_requests(greedy_sol)
```



[26]:
```
len(greedy_sol)
```

[26]: 6

[ ]:
```

```