Weighted Interval Scheduling (post-class version)

January 31, 2025

[2]:
```
import random

def random_request():
    return [sorted(random.sample(range(100),2)), random.random()*10]
```

[3]:
```
R = random_request()
print(R)
print(R[0])
print(R[1])
```

```
[[54, 73], 2.655663065195429]
[54, 73]
2.655663065195429
```

[4]:
```
def make_requests(n):
    return [random_request() for i in range(n)]
```

[5]:
```
make_requests(3)
```

[5]:
```
[[[22, 33], 7.147000831149654],
 [[74, 95], 6.225874716836338],
 [[29, 41], 9.100380730769405]]
```

[6]:
```
def compatible(r1, r2):
    # returns True if the meetings DON'T overlap, returns
    # False if they do
    return r2[0][1] <= r1[0][0] or r2[0][0] >= r1[0][1]

def is_compatible(request, solution):
    # [solution] is a list of requests
    # returns True if [request] does not overlap with ANY of
    # the requests in [solution]

    # return all(compatible(request, r) for r in solution)

    for r in solution:
        if not compatible(r, request):
            return False
```

```
    return True
```

[7]:
```
def plot_requests(requests):
    for r in sorted(requests, key=lambda x : x[0][1]):
        print(" "*(r[0][0]) + "-"*(r[0][1]-r[0][0]) + "  (" +␣
 ↪str(round(r[1],2)) + ")")
    #print("total value:",sum(r[1] for r in requests))
    total_value = sum(r[1] for r in requests)
    print(f"total value: {total_value}")
```

[8]:
```
# best = most valuable
def greedy(requests):
    sorted_requests = sorted(requests, key=lambda r: r[1], reverse=True)
    # sorted_requests now holds all of our input data in order
    # from most profitable to least profitable

    solution = []
    solution.append(sorted_requests.pop(0))

    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        if is_compatible(request, solution):
            solution.append(request)

    return solution
```

[9]:
```
requests = make_requests(100_000)
```

[10]:
```
#plot_requests(requests)
```

[11]:
```
sol = greedy(requests)
print(sol)
print(len(sol))
```

```
[[[31, 86], 9.99981311969099], [[23, 31], 9.999402408682357], [[3, 21],
9.996360859932542], [[91, 92], 9.99557350829777], [[87, 90], 9.986668108917856],
[[0, 1], 9.983479756607625], [[95, 99], 9.970952209309194], [[1, 2],
9.959708662058022], [[93, 95], 9.958716926593937], [[90, 91], 9.95398332938079],
[[21, 22], 9.89464972932516], [[22, 23], 9.861395510832248], [[92, 93],
9.738340522260902], [[2, 3], 9.430269181586771], [[86, 87], 9.249837482903994]]
15
```

[12]:
```
plot_requests(sol)
```

```
-  (9.98)
 -  (9.96)
  -  (9.43)
```

```
   ------------------  (10.0)
                    -  (9.89)
                     -  (9.86)
                      --------  (10.0)
-------------------------------------------------------  (10.0)
      -  (9.25)
       ---  (9.99)
          -  (9.95)
           -  (10.0)
            -  (9.74)
             --  (9.96)
               ----  (9.97)
total value: 147.97915131638015
```

[ ]:

[13]:
```
sum(s[1] for s in sol)
```

[13]:
```
147.97915131638015
```

[14]:
```
# best = most valuable
# best = shortest
# best = most value-dense (highest value/duration)
```

[15]:
```
def greedy(requests, sort_function):
    sorted_requests = sorted(requests, key=sort_function)
    solution = []
    solution.append(sorted_requests.pop(0))

    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        if is_compatible(request, solution):
            solution.append(request)

    return solution
```

[16]:
```
# request = [[start, end], value]
most_value = lambda req : -req[1]
shortest = lambda req : req[0][1] - req[0][0]
density = lambda req : -req[1]/(req[0][1] - req[0][0])
```

[17]:
```
print(most_value)
print(most_value([[10,20], 15.1]))
```

```
<function <lambda> at 0x1061ffec0>
-15.1
```

[18]:
```
requests = make_requests(1000)
```

[19]:
```
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
s3 = greedy(requests, density)
```

[20]:
```
plot_requests(s1)
```

```
   ---  (8.03)
      -----  (9.95)
------------------------------------------------------------------------------
(9.99)
            --  (9.58)
               ----  (7.93)
total value: 45.47875708604689
```

[21]:
```
plot_requests(s2)
```

```
----  (2.75)
    -  (5.26)
          -  (6.99)
             ---  (2.49)
                -  (7.59)
                    -  (6.93)
                      -  (0.94)
                       -  (1.49)
                        --  (5.28)
                            -  (1.15)
                             -  (7.41)
                                  -  (9.23)
                                    --  (7.19)
                                         -  (3.78)
                                              --  (0.21)
                                                  -  (1.0)
                                                   -  (1.56)
                                                    -  (5.41)
                                                       -  (6.0)
                                                         -  (1.89)
                                                            -  (3.5)
                                                             -  (7.14)
                                                               ---  (7.09)
                                                                  --  (0.26)
                                                                    ---  (5.38)
                                                                       --  (0.99)
                                                                         --
(5.07)                                                                     --
(6.47)                                                                       ---
(1.56)
```

```
    --  (8.12)
        --  (4.19)
            --  (9.58)
               ---  (1.78)
total value: 145.7031804316558
```

[22]:
```
plot_requests(s3)
```

```
----  (2.75)
    -  (5.26)
          -  (6.99)
             ---  (2.49)
                -  (7.59)
                    -  (6.93)
                      -  (0.94)
                       -  (1.49)
                        --  (5.28)
                           --  (9.26)
                             -  (7.41)
                                  -  (9.23)
                                    --  (7.19)
                                         -  (3.78)
                                              ---  (7.46)
                                                  -  (1.0)
                                                   -  (9.03)
                                                    -  (5.41)
                                                      -  (6.0)
                                                        -  (1.89)
                                                           -  (3.5)
                                                            -  (7.14)
                                                               ----  (9.49)
                                                                     ---  (9.74)
                                                                        --
(5.07)
                                                                            --
(6.47)
-----  (8.71)
    --  (8.12)
        --  (9.94)
            --  (9.58)
               ----  (7.93)
total value: 193.09525334628933
```

[ ]:
```

```

[24]:
```
requests = make_requests(1000)
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
```

```
s3 = greedy(requests, density)
def score(sol):
    return sum(s[1] for s in sol)
print([score(s1), score(s2), score(s3)])
```

```
[35.64139129194101, 164.8683591578198, 194.8334429216566]
```

[ ]:

[ ]: