Lecture 07 - Backtracking - Weighted Interval

March 2, 2022

[1]:
```
# ipynb
import random
from itertools import chain, combinations
```

[2]:
```
def powerset(iterable):
    """
    Return all subsets of the input.
    """
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s) + 1))

print(list(powerset([1,2,3])))
```

```
[(), (1,), (2,), (3,), (1, 2), (1, 3), (2, 3), (1, 2, 3)]
```

[3]:
```
def random_request():
    """
    Create one random request [[start, end], value], where start and
    end are integers in range(100), and the value is a real number
    between 0 and 10.

    Only returns requests of length >= 5
    """
    req = [sorted(random.sample(range(100), 2)), random.random() * 10]
    while req[0][1] - req[0][0] < 5:
        req = [sorted(random.sample(range(100), 2)), random.random() * 10]
    return req
```

[4]:
```
def make_requests(n):
    """
    Return <n> requests.
    """
    return [random_request() for i in range(n)]
```

[5]:
```
def compatible(r1, r2):
    """
    Determine whether two requests are compatible.
    """
```

```
    return r2[0][1] <= r1[0][0] or r2[0][0] >= r1[0][1]
```

[6]:
```
def is_compatible(request, solution):
    """
    Determine whether <request> is compatible with <solution>, which
    is a list of requests.
    """
    r = request
    return all(compatible(r, s) for s in solution)
```

[7]:
```
def valid_solution(solution):
    """
    Determine if a set of requests is valid (each request valid
    with all other requests).
    """
    return all(
        is_compatible(solution[i], solution[i + 1 :]) for i in␣
 ↪range(len(solution) - 1)
    )
```

[8]:
```
def score(solution):
    """
    Return the total value of <solution>.
    """
    return sum(r[1] for r in solution)
```

[9]:
```
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))
            + ")"
        )
```

[10]:
```
def greedy(requests, sort_key):
    sorted_requests = sorted(requests, key=sort_key)  # O(n*log(n))
    solution = []
    solution.append(sorted_requests.pop(0))

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

    return solution
```

```
shortest = lambda x: x[0][1] - x[0][0]
most_value = lambda x: -x[1]
density = lambda x: -(x[1]) / (x[0][1] - x[0][0])
```

[11]:
```
def brute_force(requests):
    all_poss = powerset(requests)
    best_score = 0
    best_sol = []
    for sol in all_poss:
        if not valid_solution(sol):
            continue
        sc = score(sol)
        if sc > best_score:
            best_score = sc
            best_sol = sol
    return best_sol
```

[12]:
```
def backtracking(remaining_requests):
    """
    find next valid requests
    add it or not, return which of those leads to the largest score
    if no other valid requests, just return score
    """
    rr = list(remaining_requests)

    #if not rr:
    if len(rr) == 0:
        return []

    to_add = rr.pop(0)

    remaining_valid_requests = [
        req for req in rr if compatible(to_add, req)
    ]
    version_accept = [to_add] + backtracking(remaining_valid_requests)

    version_reject = backtracking(rr)

    #return max([version_accept, version_reject], key=score)
    if score(version_accept) > score(version_reject):
        return version_accept
    return version_reject
```

[13]:
```
R22 = make_requests(22)
```

```python
s1 = greedy(R22, shortest)
s2 = greedy(R22, most_value)
s3 = greedy(R22, density)
print("greedy done")
```

```
greedy done
```

[14]:
```python
s4 = brute_force(R22)
print("brute force done")
```

```
brute force done
```

[15]:
```python
s5 = backtracking(R22)
print("backtracking done")
```

```
backtracking done
```

[16]:
```python
sols = [s1, s2, s3, s4, s5]
scores = [score(s) for s in sols]
```

[17]:
```python
scores
```

[17]:
```
[31.471392174298487,
 30.710935928729906,
 30.710935928729906,
 32.91882447708572,
 32.91882447708572]
```

[18]:
```python
for s in sols:
    plot_requests(s)
    print("\n","|"*110,"\n")
```

```
      -------------   (9.09)
                      -----------   (2.07)
                                      --------   (9.79)
                                                              ---------   (2.78)
--------------------   (7.74)

 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

      -------------   (9.09)
                      -----------   (2.07)
                                      --------   (9.79)
--------------   (9.76)

 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

      -------------   (9.09)
```

```
                    -----------   (2.07)
                                  --------   (9.79)
--------------   (9.76)

 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

      -------------   (9.09)
                    -----------   (2.07)
                                  --------   (9.79)
                                             ----------------------   (4.23)
--------------------   (7.74)

 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

      -------------   (9.09)
                    -----------   (2.07)
                                  --------   (9.79)
                                             ----------------------   (4.23)
--------------------   (7.74)

 ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||
```

[ ]:

[ ]:

[ ]:

[19]:
```
R200 = make_requests(200)

s1 = greedy(R200, shortest)
s2 = greedy(R200, most_value)
s3 = greedy(R200, density)
print("greedy done")
```

```
greedy done
```

[20]:
```
# s4 = brute_force(R200)    # 10^48 years!
# print("brute force done")
```

[21]:
```
s5 = backtracking(R200)
print("backtracking done")
```

```
backtracking done
```

[22]:
```python
sols = [s1, s2, s3, s5]
scores = [score(s) for s in sols]
```

[23]:
```python
scores
```

[23]:
```
[60.90592824864833, 53.84833491265685, 66.78294427356698, 75.10629493261888]
```

[24]:
```python
for s in sols:
    plot_requests(s)
    print("\n","|"*110,"\n")
```

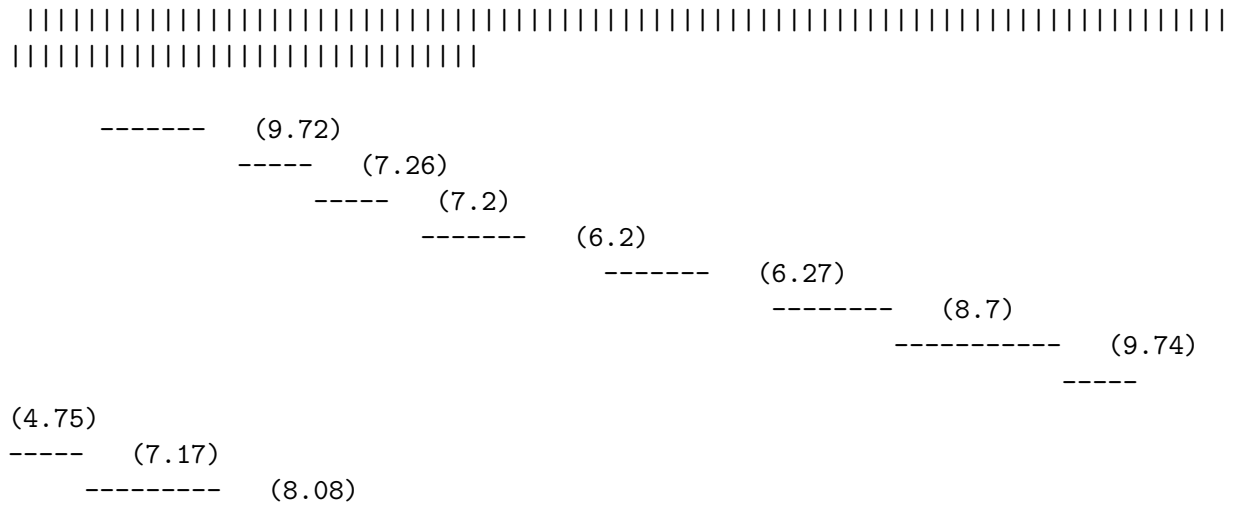
```
 ------    (4.58)
               -----   (7.26)
                    -----   (7.2)
                           -------   (6.2)
                                     -----   (5.95)
                                                   --------   (8.7)
                                                               ------   (4.69)
                                                                      -----
(4.75)
-----   (7.17)
   -------   (4.41)

 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

      -------   (9.72)
               -----   (7.26)
                    -----   (7.2)
                             ---------   (9.99)
                                          -------------------------------
(9.89)
------------------   (9.78)

 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
|||||||||||||||||||||||||||||||

   ------    (8.45)
               -----   (7.26)
                    -----   (7.2)
                           -------   (6.2)
                                     -----   (5.95)
                                                   --------   (8.7)
                                                                  --------
(7.76)
-----   (7.17)
     ---------   (8.08)
```


(9.72)
(7.26)
(7.2)
(6.2)
(6.27)
(8.7)
(9.74)
(4.75)
(7.17)
(8.08)

[ ]:

[ ]:

[ ]: