Algorithms Homework 7 Code Report

Ryan Rau [redacted]

April 17 2020

1 Problem

Consider a modification to the activity-selection problem in which each activity a_i has, in addition to a start and finish time, a value v_i . The objective is no longer to maximize the number of activities scheduled, but instead to maximize the total value of the activities scheduled. That is, we wish to choose a set A of compatible activities such that the summation of v is maximized. Give a polynomial-time algorithm for activities such that P this problem.

2 Solution

2.1 Algorithm

To start I began with the following pseudo code for a normal activity selector:

```
GREEDY-ACTIVITY-SELECTOR(s,f)
    n = s.length
    A = {a1}
    k = 1
    for m = 2 to n
        if s[m] >= f[k]
            A = A U {am}
            k = m
    return A
```

With that as a base, I programmed it in java and got a normal activity selector working.

To make things easier on myself, I also made an activity object that would store all the values and that I could easily pass as one array.

At first I started with a greedy approach, looking at the activity with the highest value, but soon realized that it wouldn't always produce the highest value due the constraints of how the activities run. Additionally, I need to measure if there was multiple solutions. So I stuck with the original greedy approach of taking the activity with the lowest end time and I but the above code into another loop that would vary where the activity sequence would start. With that I was able to measure all sequences and determine if there were duplicates with the same value.

2.2 Time Complexity

The original activity selector runs at a time of O(n) with the addition of my each loop with contains the original problem within it my code runs at O(n^2). However my code expects that the input to be sorted by end time, so prior to finding the max value from a grouping of activities I use the Arrays.sort method to take care of that which would add an additional O(nlogn) to the run time.

2.3 Testing

To test my activity selector, I first mentally worked through the sample inputs we were given to figure out what was the max possible value. With that calculated I then ran those data sets through my program and compared its results with my results.

The following is an example command for how to run my program.

Will find the sequence that produces the highest value and tell whether its unique or not in the output file .

```
javac Homework7.java
java Homework7 input.txt output.txt
```

2.4 Results

Results when given input1.tx:

```
Max value: 3
Sequence IDs: 1 4
IT HAS MULTIPLE SOLUTIONS
```

Result when given input2.txt:

```
Max value: 1095
Sequence IDs: 1 2
IT HAS A UNIQUE SOLUTION
```

2.5 Conclusion

Overall, I was able to find what sequence of activities would produce the greatest value in O(n^2) time along with determine if that solution was unique or not.