Lab wk1-1: Interval Scheduling and Euclidean Algorithm

Deliverable: Submit you answer sheet at the end of lab

Writing algorithms

A Note about Writing Algorithms: When writing pseudo-code do **not** give a complete program. Give just enough detail that your intentions are clear and unambiguous, but do not provide extraneous details (complex syntax and/or variable declarations) which are **otherwise obvious**. English explanations, examples, and figures should be given to illustrate your intentions (and can help in assigning partial credit).

Interval scheduling

Given a list of tasks where each task has with a time interval during which it would be completed. For instance, one task might run from 2:00 to 5:00 and another task might run from 6:00 to 8:00. The goal is to maximize the number of executed tasks without overlapping any of the tasks.

Thus a request corresponds to an interval of time. We say that a subset of requests is **compatible** if no two of them overlap in time and our goal is to accept as large a **compatible** subset as possible. A **compatible** set of **maximum size is called optimal**.

- Your goal is to write Pseudo code to determine the maximum number of compatible tasks for tasks given by (1, s1, f1), (1, s2, f2), …, (1, sn, fn).

There are three "obvious" greedy approaches to solving this problem. Today you will only consider these three greedy approaches. The steps in trying to find a greedy algorithm for a problem are usually:

1. Brainstorm different possible greedy approaches.
2. Look for counterexamples to rule out each approach.
3. Try to prove correctness of the approach. This may require developing pseudo code for the algorithm.

The greedy approaches you will evaluate are:

- create a subset as follows: **sort tasks by start time** and go through the list of tasks in order and add a task to the subset only if it is compatible with the tasks already in the subset ***Earliest start time first***
  - Find a counterexample and draw a picture showing that the algorithm does not find the largest size compatible set.
- create a subset as follows: **sort tasks by duration** and go through the list of tasks in order and add a task to the subset only if it is compatible with the tasks already in the subset ***Shortest duration first***
  - Find a counterexample and draw a picture showing that the algorithm does not find the largest size compatible set.
- create a subset as follows: **sort tasks by finish time** and go through the list of tasks in order and add a task to the subset only if it is compatible with the tasks already in the subset ***Earliest finish time first***
  - This algorithm always finds the maximum compatible set (i.e. it is correct). Make an argument as to why this algorithm is correct? Find someone in the class to play devil's advocate?
  - Write pseudo-code that describes the algorithm.
    - Step 1 sort the intervals. You do not need to write a sorting algorithm, just say what algorithm you would use?
    - Step 2: Given a compatible set of the intervals in order of their finish times and another interval, write an algorithm to check if the new interval is compatible with the existing compatible set. How many comparisons does this take in the best case? In the worst case?
    - Step 3: Finally write a loop that uses the sorted intervals sorted by finish time and the algorithm from Step 2 to specify an algorithm that finds a maximum compatible set.
  - What is the overall complexity of the full algorithm and why?

Euclid's algorithm:

In mathematics, the Euclidean algorithm[a], or Euclid's algorithm, is an efficient method for computing the greatest common divisor (GCD) of two numbers, the largest number that divides both of them without leaving a remainder. It is named after the ancient Greek mathematician Euclid, who first described it in Euclid's Elements (c. 300 BC).

The Euclidean algorithm has many theoretical and practical applications. It is used for reducing fractions to their simplest form and for performing division in modular arithmetic. Computations using this algorithm form part of the cryptographic protocols that are used to secure internet communications, and in methods for breaking these cryptosystems by factoring large composite numbers.

```
private static int gcdIterative (int a, int b) {
     int temp = 0;
     while (b != 0) {
          remainder = a % b;
          a = b;
          b = remainder;
     }
     return a;
}

private static int gcdRecursive (int a, int b) {
     if (b == 0)
          return a;
     return gcdRecursive(b, a % b);
}
```

Find gcd(31415, 14142) by applying Euclid's algorithm.

1. Euclid's algorithm, as presented in Euclid's treatise, uses subtractions rather than integer divisions. Write pseudocode for this version of Euclid's algorithm.

Euclid's game (starts with two unequal positive integers on the board. Two players move in turn. On each move, a player has to write on the board a positive number equal to the difference of two numbers already on the board; this number must be new, i.e., different from all the numbers already on the board. The player who cannot move loses the game. Should you choose to move first or second in this game?