1 Project Instructions

1. The project can be done in teams of two. It is enough if your team turns in one project report.

2. You are responsible for completely understanding the algorithms which you will be implementing. You are also responsible for understanding the project description well. If you feel there are parts of the paper or the project description which is not well specified, you should consult the TA or instructor well in advance.

2 Introduction

The goal of this project is to explore two different algorithmic approaches for the weighted interval scheduling problem (WISP). Recall that an instance of WISP is defined as follows: we are given a set of intervals \( \{1, 2, \ldots, n\} \). Each interval has a start time \( s_i \), finish time \( f_i \), and a weight \( w_i \). There is a single resource to schedule these intervals. A subset of intervals is valid if they are all mutually disjoint (i.e., do not overlap in time) and hence can be scheduled using the single resource. Our objective is to pick a valid subset with the maximum total weight. Recall that we saw an optimal algorithm in the class for this problem which is based on dynamic programming. The paper by Erelebach and Spieksma discusses a greedy algorithm for a generalized version of WISP. In particular, the greedy algorithm does not assume that the number of resources is one but considers a more general case where the number of resources is \( m \).

3 What you need to do

Your first task is to read sections 1 and 2 from this paper and understand the greedy algorithm. The terminology and the assumptions made in the paper is different from the ones used in the book. In particular, the following are some points to note about the greedy algorithm.

1. The paper assumes that the input consists of several jobs, each of which consists of many intervals. The algorithm is required to pick at most a single interval from each job such that the final set of intervals can be scheduled using \( m \) resources (i.e., their depth must be at most
In other words, the goal of the greedy algorithm is to obtain the maximum weight subset of intervals with depth \( m \) such that at most one interval is picked from each job.

In this project, we restrict our attention to the special case where \( m = 1 \). This substantially simplifies the implementation of the greedy algorithm. Further, for the purposes of this project, we assume that each job consists of only a single interval. This is a very simplified version of the problem which is addressed by the paper. This is also the version for which the dynamic program discussed in class yielded an optimal solution.

2. The crux of the greedy algorithm is detecting the minimum weight conflict set for each interval. Notice that for the special case considered in this project, \( m = 1 \), this can be implemented in a very simple way.

Your second task is to implement the greedy algorithm and the dynamic programming algorithm in a language of your choice. Your implementation of the greedy algorithm should also take the parameter \( \alpha \) (see the paper) as part of the input. You will also generate a set of ten random inputs for testing your algorithm (the inputs will be specified shortly). For each of these inputs, you should measure the value of the solution returned by the greedy algorithm for ten different values of \( \alpha \): \{0.1, 0.2, \ldots, 1.0\}; you should also measure the solution value returned by the dynamic programming algorithm. Further, you should also measure the run times of each of these algorithms on the inputs (make sure that the units of time you use is good enough to distinguish the run time between for various inputs). Thus, for each input you need to report a total of 22 different measurements. Since the number of inputs is ten, you will tabulate a total of 220 measurements in your report.

The ten inputs are generated as follows. The \( i^{th} \) input consists of 100 \( i \) intervals. For each input, the start times, finish times, and weights of the intervals are generated randomly as follows. For an interval \( j \), the start time \( s_j \) is a real number (float) chosen uniformly at random in the range \([0, 20]\). Its length \( l_j \) is chosen uniformly at random in the range \([0, 1]\). Hence, its finish time \( f_j = s_j + l_j \) is in the range \([0, 21]\). Its weight \( w_j \) is chosen uniformly in the range \([0, 1]\).

4 Deliverables

Your project report consists of the following parts:

- The pseudo-code for the two algorithms in sufficient detail (please do not give C/C++/Java style code here). (3 points)

- Analysis of the run-times for both the algorithms. You should present the runtimes in the “big-O” notation for both the algorithms. The pseudo-code should be in sufficient detail so that your run-time analysis becomes clear immediately. (2 points)

- The tables which list the measurements for solution values and run-times. (10 points)

- A listing of your actual code. (No points for this although this must be part of your project submission). Do not email your code to the instructor or the TA!