CS 341: Algorithms @ Waterloo **Last Updated:** Mar 1, 2023
Homework 3 **Due Date: Mar 10, 2023, 11:00pm**

You are allowed to discuss with others but are not allowed to use any references other than the course notes and the three reference books. Please list your collaborators for each question. You must write your own solutions. See the course outline for the homework policy.

The full mark is 50. This homework is counted 8% of the course grade.

---

1. **Programming Problem: Holiday Scheduling** (16 marks)

   Bob is a University of Gooseland student. Like many other U Gooseland students, he has a lot of things to do and not a lot of time to do them.

   A long-awaited long weekend is coming up and it is time for Bob to make plans on what to do. He has N activities on the list – which could be to study for an upcoming midterm, meditate, go skiing, go on a date, etc. The activities are conveniently labeled 1 to N. For the i-th activity, there is a start time S_i, an end time T_i, a value V_i, and an energy requirement E_i.

   At time 0, Bob's energy level is M. Between the time he starts and ends activity i, his energy level drops by E_i. (The rate of decrease does not matter.) Every unit of time that he is not doing anything, his energy level increases by 1. It is possible for his energy level at a certain time to be greater than the initial energy level M.

   Bob's goal is to fit some of the listed activities in his schedule, so that:

   - The scheduled activities do not overlap (note: when you finish an activity, it is fine to start the next one immediately);
   - His energy level remains non-negative at all times;
   - The total value of the scheduled activities is maximized.

   Help Bob optimize his long weekend!

   Input: The first line has two integers N and M, the number of activities and the initial energy level. N lines follow. The i-th line has four integers S_i, T_i, V_i, E_i, respectively the start time, end time, value, and energy requirement of the i-th activity.

   It is guaranteed that $1 \leq N \leq 100$, $0 \leq M \leq 500$; $0 \leq S_i < T_i \leq 432$, $1 \leq V_i \leq 10000$, and $0 \leq E_i \leq 1000$ for $1 \leq i \leq N$. It is **not** guaranteed that the activities are sorted by start or end time.

   Output: Output an optimal schedule. On the first line, output two integers K and V, where K is the total number of scheduled activities and V is their total value. On the second line, output K integers, the scheduled activities in chronological order (i.e. the earlier activity should get output first). If K equals 0, output a blank line.

   If there are multiple schedules that maximizes V, any of them will be accepted.

   Sample Input 1:

   ```
   4 100
   0 10 1000 101
   10 20 100 55
   20 30 200 55
   ```

```
144 154 500 115
```

Sample Output 1:

```
2 700
3 4
```

Sample Input 2:

```
4 0
144 154 10 115
0 10 1000 101
10 20 100 55
20 30 200 55
```

Sample Output 2:

```
1 10
1
```

2. **Written Problem: Maximum Interval Coloring** (12 marks)

   In class, we have studied the interval scheduling problem and the interval coloring problem. Here we consider a common generalization of these two problems. We are given n intervals $[s_i, f_i]$ where each s_i and f_i are positive integers with $s_i < f_i$, and a positive integer k. Our goal is to use k colors to color as many intervals as possible, so that each interval receives at most one color (could have no color on an interval) and no two overlapping intervals can receive the same color (two intervals $[s_i, f_i]$ and $[s_j, f_j]$ are overlapping if $[s_i, f_i] \cap [s_j, f_j] \neq \emptyset$). Notice that the interval scheduling problem is the special case when $k = 1$, and the interval coloring problem is the special case to find the minimum k so that all the intervals can be colored. You can think of this problem as using k rooms (k colors) to schedule as many activities (intervals) without time conflicts as possible.

   Design an efficient algorithm to find such a coloring. You will get full marks if the time complexity of the algorithm is $O(n \log n)$ and the proofs are correct. To implement the algorithm efficiently, you can use a balanced search tree data structure (such as AVL tree) and assume all the operations (add, delete, search) can be done in $O(\log |T|)$ time where $|T|$ is the size of the search tree.

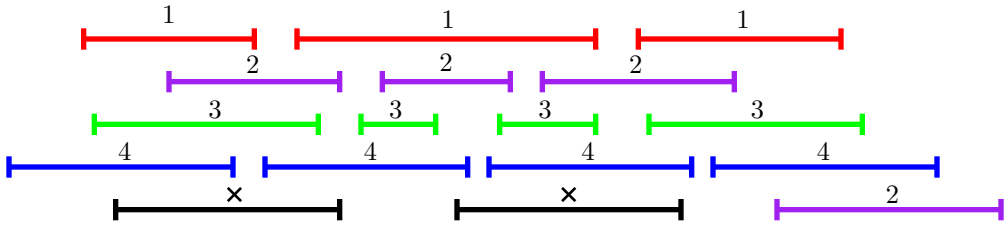

Figure 1: For this example, k=4. We can color all but two of the intervals using four colors as shown in the figure, and this is the maximum possible.

3. **Written Problem: Thickest Paths** (10 marks)

Imagine that an online video provider would like to find paths of highest bandwidth to send its videos to the receivers. This can be modeled as a graph problem. We are given a directed graph $G = (V, E)$ where every directed edge e has a positive integer thickness b_e (representing the bandwidth of a link). Given a source vertex s and a receiver vertex t, let $\mathcal{P}_{s,t}$ be the set of all directed paths from s to t in G. We would like to find a path in $\mathcal{P}_{s,t}$ that maximizes the minimum thickness on the path, i.e.,

$$\max_{P \in \mathcal{P}_{s,t}} \min_{e \in P} b_e.$$

Design an efficient algorithm to find a thickest path from s to t for all $t \in V - s$. You will get full marks if the time complexity of the algorithm is $O((|V| + |E|) \log |V|)$ and the proofs are correct.

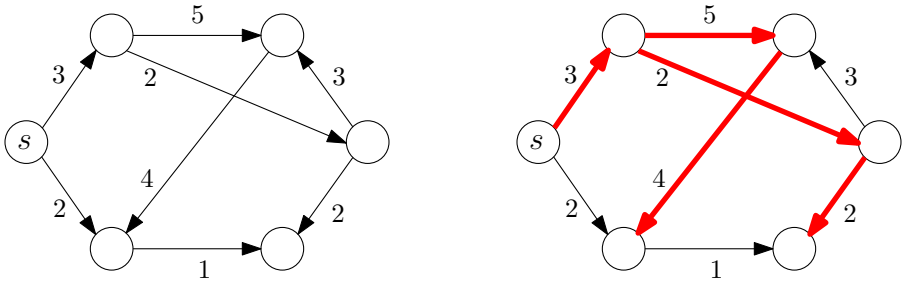

Figure 2: The highlighted edges on the right form the thickest paths from s to all other vertices.

4. **Written Problem: Updating Minimum Spanning Tree** (12 marks)

We must often perform computations on inputs that change over time. In this problem, rather than just computing a minimum spanning tree of a graph, we keep track of a minimum spanning tree as the graph changes. We are given a weighted graph $G = (V, E, c)$ and a minimum spanning tree T of G. Suppose that the cost of an edge uv is changed to a new number $x > 0$. Design the fastest algorithm that you can that computes a minimum spanning tree of the new graph, prove its correctness and analyze its time complexity. (You may consider the case when the cost of uv is increased and the case when the cost of uv is decreased separately.)