CS 341: Algorithms @ Waterloo **Last Updated:** June 29, 2021
Homework 3 **Due Date: July 7, 2021, 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 10% of the course grade.

---

1. **Programming Problem: Shortest Snowboarding Course** (20 marks)

   It is only the beginning of summer, but Alice already misses winter so much. Knowing that, Bob gives Alice the following problem that is purportedly related to snowboarding.

   Let's model the mountain range as a grid, with pistes represented as H horizontal lines and V vertical lines. The lines form $H \times V$ intersections. We use coordinates (i, j) to denote the point of intersection of the i-th horizontal line (counting from top to bottom) with the j-th vertical line (counting from left to right).

   We want to design a snowboarding course. The starting point is $(1, 1)$ and the ending point is (H, V). At each point you are only allowed to go right or go downwards. That means, at (i, j) you can only go to $(i + 1, j)$ or $(i, j + 1)$. The time needed to travel from (i, j) to $(i + 1, j)$ is denoted as $D_{i,j}$. The time needed to travel from (i, j) to $(i, j + 1)$ is denoted as $R_{i,j}$. One cannot go right at (i, j) in case $j = V$, so in this case $R_{i,j}$ is undefined. Similarly, if $i = H$, then $D_{i,j}$ is undefined. In all other cases, $D_{i,j}$ and $R_{i,j}$ are defined and have positive integer value.

   Let's call a horizontal (resp. vertical) section any segment connecting a point (i, j) to point $(i, j + 1)$ (resp. $(i + 1, j)$). When one is near the end of the current section ready to turn into a perpendicular section, one needs to decelerate, make the turn, and accelerate again. This takes extra time. In this problem, we model the effect of turning as follows: if a section is either immediately preceded or immediately succeeded by a section with perpendicular direction, its travel time is **double** that given by $D_{i,j}$ or $R_{i,j}$. Otherwise, its travel time is unaffected.

   Our task is to find a snowboarding course from $(1, 1)$ to (H, V) that takes the least amount of time.

   Input: The first line of input contains H and V. H lines follow. The i-th of these lines contains $2V$ integers $R_{i,1}, D_{i,1}, R_{i,2}, D_{i,2}, \ldots, R_{i,V}, D_{i,V}$. If $D_{i,j}$ or $R_{i,j}$ is undefined, the corresponding entry is replaced by a -1.

   It is guaranteed that $2 \leq H, V \leq 500$, and all defined $D_{i,j}$ and $R_{i,j}$ are positive and at most 100.

   Output: On the first line, output the time needed to finish the shortest snowboarding course. On the next $H + V - 1$ lines, output the points that the course will pass through in order, starting from $(1, 1)$ and ending at (H, V). Output `i j` for the point (i, j).

   In case there are multiple optimal solutions, any one of them will be accepted.

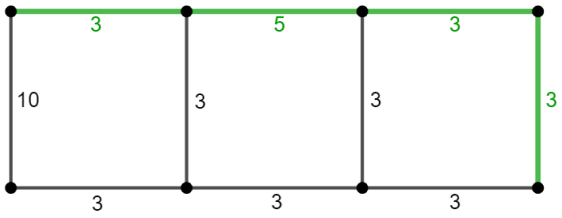
   Sample Input 1:

   ```
   2 4
   3 10 5 3 3 3 -1 3
   3 -1 3 -1 3 -1 -1 -1
   ```

Sample Output 1:

```
20
1 1
1 2
1 3
1 4
2 4
```

Explanation: The figure below shows the grid along with the travel time of each section. The optimal path, highlighted in green, takes time $3 + 5 + 3 \times 2 + 3 \times 2 = 20$. Note that the path $(1, 1) \to (1, 2) \to (2, 2) \to (2, 3) \to (2, 4)$ has a smaller sum of values, but after taking into account the effect of turning, it takes a longer time $(3 \times 2 + 3 \times 2 + 3 \times 2 + 3 = 21)$.



Here are some additional sample inputs (the optimal paths are unique in each case):

Sample Input 2:

```
2 3
1 2 100 1 -1 100
2 -1 100 -1 -1 -1
```

Sample Output 2:

```
108
1 1
2 1
2 2
2 3
```

Sample Input 3:

```
3 3
2 1 100 2 -1 100
1 100 100 20 -1 100
100 -1 1 -1 -1 -1
```

Sample Output 3:

```
46
1 1
2 1
2 2
3 2
3 3
```

2. **Written Problem: Meeting the Deadlines** (10 marks)

It is the midterm period again. You have taken many courses, with assignments and exams lining up. You also have other commitments and would like to find a schedule that allows you to meet all the deadlines. We model this scheduling problem as follows.

We are given n jobs, each with a release time $r_i \in \mathbb{Z}_+$, a deadline $d_i \in \mathbb{Z}_+$ and a processing time $p_i \in \mathbb{Z}_+$. Our task is to output a feasible schedule to finish the jobs so that all the deadlines are met, or determine that such a schedule does not exist. A schedule is a set of disjoint time intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_k, f_k]$ (with $s_1 < f_1 \leq s_2 < f_2 \leq \ldots \leq s_k < f_k$), along with the associated jobs $j_1, j_2, \ldots, j_k \in \{1, \ldots, n\}$ in each time interval such that we process job j_t during interval $[s_t, f_t]$. Note that a job i can be processed in different time intervals (i.e. we do not have to finish a job in one time interval), but in each time unit we can only process one job. Given a schedule, let $T_i := \{t \mid j_t = i\}$ be the set of time intervals that are scheduled to process job i. A schedule is feasible if the following conditions hold for every job $1 \leq i \leq n$:

(a) $r_i \leq s_t$ for each time interval $t \in T_i$, i.e. we can only process job i after it is released in time r_i;

(b) $f_t \leq d_i$ for each time interval $t \in T_i$, i.e. we can only process job i before its deadline in time d_i;

(c) $\sum_{t \in T_i} (f_t - s_t) \geq p_i$, i.e. the total length of the intervals assigned to job i is at least its processing time p_i.

Here is an example with $n = 3$ jobs, and a corresponding feasible schedule:

i	r_i	d_i	p_i	Intervals for job i
1	0	8	5	$[0, 3], [4, 5], [6, 7]$
2	3	6	2	$[3, 4], [5, 6]$
3	6	9	2	$[7, 9]$

Design the fastest algorithm that you can for the problem of outputting a feasible schedule or determining that a feasible schedule does not exist. You must prove the correctness and analyze the time complexity of your algorithm.

3. **Written Problem: Sketching Data** (10 marks)

One of the major ways of dealing with big data sets is sketching: the approximation of the data set by a smaller but approximately similar object. In this problem, we will see a way of sketching a two-dimensional plot. The input consists of a set of points $(x_1, y_1), \ldots, (x_n, y_n)$, where each x_i and y_i is a real number. You may assume that all of the x_i are distinct, and that they are in order so that $x_1 < x_2 < \cdots < x_n$.

We will approximate this plot by a piece-wise constant function, that we call a *sketch*. You can see an example of a sketch in Figure 1 below. A sketch is specified by a partition of the data into consecutive intervals along with a value for each interval. That is, we will partition $\{1, \ldots, n\}$ into consecutive intervals. As one interval will start right after another ends, it suffices to specify the first index in each interval. A partition with k intervals should be specified by their starting indices, $1 = s_1 < s_2 < \ldots < s_k \leq n$. The end index of interval j will then be $f_j = s_{j+1} - 1$ if $j < k$ and $f_j = n$ if $j = k$.

We then need to specify the height of the sketch in each interval. For the jth interval, we then approximate the points (x_i, y_i) for $s_j \leq i \leq f_j$ by a horizontal line segment of height h_j. So, a sketch is fully specified by $s_2, \ldots, s_k$ and $h_1, \ldots, h_k$. If we use more partitions in our sketch, we can represent the data better.

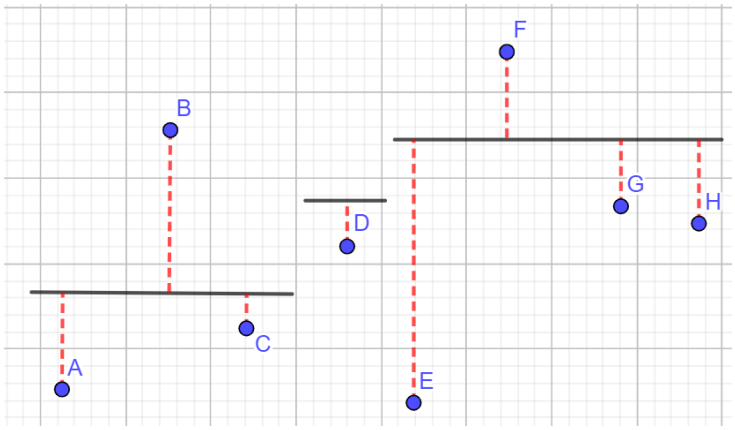


Figure 1: A sketch of $n = 8$ data points. It has 3 segments. The red lines represent the error in the approximation of every point. Note that this is not the best sketch of these points.

We now define the *error* of a sketch to be the maximum error that it makes at any point:

$$\max_{1 \le j \le k} \max_{i : s_j \le i \le f_j} |h_j - y_i|.$$

Design the fastest algorithm that you can for the following problem: given the points as input and an error tolerance ϵ, find a sketch of error at most ϵ that uses as few segments as possible. This is, you must minimize k, and make sure that for every j and every $s_j \le i \le f_j$, $|h_j - y_i| \le \epsilon$. You must prove the correctness and analyze the time complexity of your algorithm.

4. **Written Problem: Verifying Shortest Path Tree** (10 marks)

   The shortest paths problem arises in many applications. For this reason, people spend a lot of time optimizing code for it. It sometimes happens that the fastest code is sometimes wrong. Or perhaps the fastest code is based on a randomized algorithm that may make mistake with a certain low probability. If we want to run the fastest code, then we should check that its output is correct. If it is not correct for some input, then we could run a slower, reliable routine. This will still be a net gain on average if we can quickly check the correctness of the code.

   Imagine that you have been given a fast but untrustworthy piece of code for computing shortest path trees. That is, the code takes as input a directed graph $G = (V, E)$, edge lengths l, and a start vertex s. It returns a tree that is supposed to be a shortest path tree from s. Your job is to design an algorithm that checks if the tree really is a shortest path tree. Your algorithm should run in time $O(n + m)$. It should output "yes" for all shortest path trees, and "no" for all trees that are not shortest path trees. Your algorithm is given access to G, l, s and the tree. You must prove the correctness and analyze the time complexity of your algorithm.