

# **Honors Discrete Mathematics: Additional Problems**

Due on Friday, April 28 at 11:59pm

*Professor Gerandy Brito Spring 2023*

**IMPORTANT:** For this assignment, you may NOT collaborate with other students.

**CS 2051 TAs**

## Challenge Problem

### Instructions

Each semester, we release a challenge problem. It's ungraded, but your submission/discussion of this problem will be considered if you ever apply to be a TA for this course. You can find the previous years' problems [here](#).

### The Problem

In the Homework 5 Supplement, we wished to find a solution to the Task Scheduling problem, which we restate here:

We are given a partially ordered set of tasks  $T = T_1, T_2, \dots, T_n$ , where each task  $T_i$  takes 1 unit of time to complete. The tasks are partially ordered by a binary relation  $<$ , which specifies the dependencies between the tasks. Specifically, if  $T_i < T_j$ , then  $T_i$  must be completed before  $T_j$  can be started.

We also have  $k$  processors  $P_1, P_2, \dots, P_k$  available to perform the tasks. A **schedule** for  $T$  assigns each task  $T_i$  to a processor  $P_j$  and a start time  $t_i$  for  $P_j$  to begin working on  $T_i$ . If a processor starts working on  $T_i$  at time  $t_i$ , then  $T_i$  will be completed at time  $t_i + 1$ .

A schedule is **feasible** if it satisfies the following conditions:

- 1 No single processor performs multiple tasks at the same time. In other words, if a processor is assigned to both  $T_i$  and  $T_j$ , where  $i \neq j$ , then the start times for the two tasks must be different.
- 2 For every pair of tasks  $T_i$  and  $T_j$  such that  $T_i < T_j$ , the start time for  $T_j$  is later than or equal to the completion time for  $T_i$ . In other words, if  $T_i$  must be completed before  $T_j$  can start, then the start time for  $T_j$  must be after or at the same time as the completion time for  $T_i$ .

The **latency** of a schedule is defined as the time between the earliest start time for any task and the latest completion time for any task. Our goal is to find a feasible schedule with the smallest possible latency.

In the [solutions](#) for this supplement, we presented and implemented a successful algorithm for this problem. However, it was fairly inefficient, and could be much improved. Your task is as follows: **Describe an efficient solution for the Task Scheduling problem.**

A few notes:

- Consider how to best represent the input data, including possibly using external data structures.
- What is the Big- $O$  time complexity of your solution? What about Big- $\Omega$ ? What input structures bottleneck your complexity?

## Extra Problem [Spring 2022]

No one solved this problem last year, so we are posing it again.

Consider the grid shown below in Figure 1.

```

      G
    G E G
  G E R E G
G E R A R E G
  G E R A N A R E G
    G E R A N D N A R E G
      G E R A N D Y D N A R E G
        G E R A N D N A R E G
          G E R A N A R E G
            G E R A R E G
              G E R E G
                G E G
                  G
  
```

Figure 1: A diamond-shaped, narcissistic grid of letters.

Suppose you start at any G, and can move only left, right, down, or up to adjacent letters. In how many ways can the palindrome GERANDYDNAREG be read if the same letter cannot be used more than once in each sequence? **Prove your answer.**

*(Hint): You can program a script to get your answer, and then work backwards to discover its origins.*