Interval Scheduling

•••

Marlie Russell
Griffin Brome
Pawan Bilkhu
Khoi Ngo
Russell Morgan

Topic Introduction

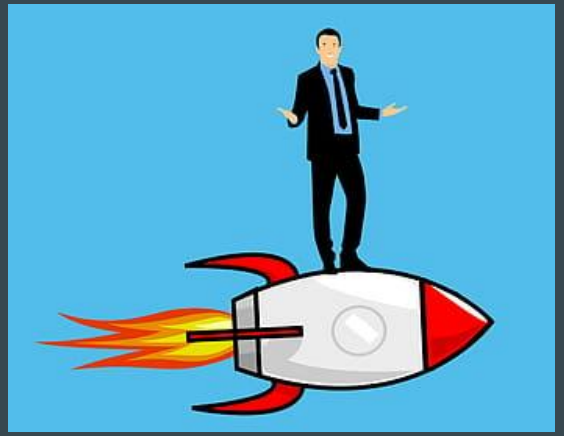
- Interval Scheduling Problem:
  - Given a set of arbitrary tasks with varying time intervals, each with defined start and finish times; how can we schedule the different tasks considering that some tasks may have overlapping schedules?
- Our Objective:
  - How can we schedule as many compatible tasks as possible?

But first we must consider....

How to Schedule?

Lowest duration

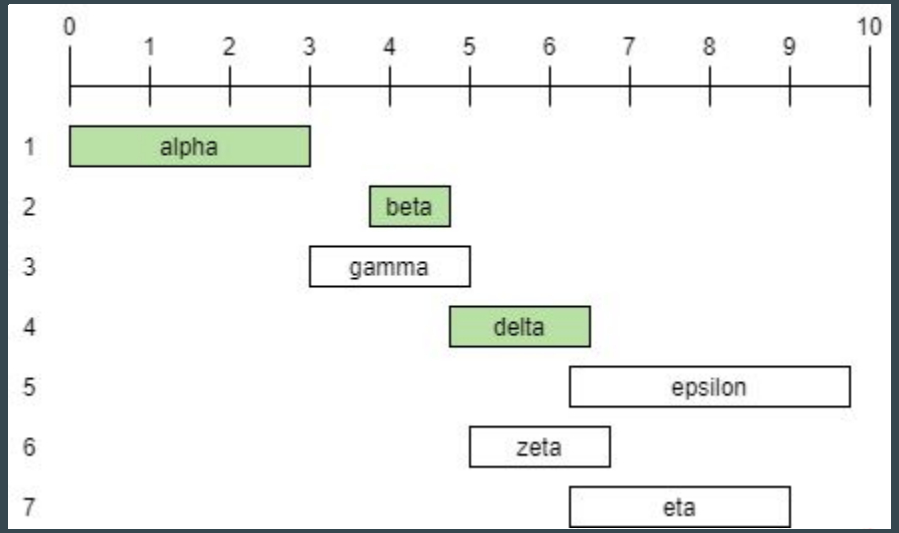
Fewest conflicts



Lowest start time

How to Schedule?

Animation of Algorithm



Assumptions

Greedy Approach - fitting in the most tasks

GREEDY-ACTIVITY-SELECTOR(s, f)

1. n= s.length
2. A = $\{a_1\}$
3. k = 1
4. for m = 2 to n:
5. if s[m] $\geq$ f[k]
6. A = A $\cup$ $\{a_m\}$
7. k = m

Proof of Correctness

- Tasks sorted by increasing finishing time
- Only get tasks where s[i] > f[k] after initial task
- We will always get the best optimal set since the first task is the best choice

Analysis of Complexity

- We will iterate through the set of all tasks
- Each task will be looked at once and only once
- We need to check if each is non-compatible with the current last choice
- This is seen in the for loop over lines 4-7 in the pseudo code
- Due to this we achieve a runtime of $\Theta(n)$

Alternate Approaches

Recursive Dynamic Programming
- Worse space complexity

Brute Force
- Worse time complexity

Sorting activities other than by increasing finish time also warrants different approaches to the problem

Where to go next?

What if the algorithm did not know the full set of intervals from the beginning, and could only see a subset at a time?

What if two tasks could be performed at once?

What if each task had a weight?

The End - Questions?