C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

Greedy 1: To check whether the scheduling problems are possible, we can use a greedy algorithm:

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

Greedy 1: To check whether the scheduling problems are possible, we can use a greedy algorithm:

Simply always work on the task with the earliest deadline. This can be implemented by keeping the active tasks in a priority queue, adding new tasks once you reach their start time.

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

Greedy 1: To check whether the scheduling problems are possible, we can use a greedy algorithm:

Simply always work on the task with the earliest deadline. This can be implemented by keeping the active tasks in a priority queue, adding new tasks once you reach their start time.

Greedy 2: To check for the maximum possible overlap, consider omitting the requirement that each task is completed. Then we can use the following greedy algorithm:

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

Greedy 1: To check whether the scheduling problems are possible, we can use a greedy algorithm:

Simply always work on the task with the earliest deadline. This can be implemented by keeping the active tasks in a priority queue, adding new tasks once you reach their start time.

Greedy 2: To check for the maximum possible overlap, consider omitting the requirement that each task is completed. Then we can use the following greedy algorithm:

Simply only work on tasks if both problems have an active task, and then work on those with the earliest deadline. The implementation is very similar to Greedy 1.

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Problem: Solve two simultaneous preemptive scheduling problems, maximizing the overlapping running time.

Greedy 1: To check whether the scheduling problems are possible, we can use a greedy algorithm:

Simply always work on the task with the earliest deadline. This can be implemented by keeping the active tasks in a priority queue, adding new tasks once you reach their start time.

Greedy 2: To check for the maximum possible overlap, consider omitting the requirement that each task is completed. Then we can use the following greedy algorithm:

Simply only work on tasks if both problems have an active task, and then work on those with the earliest deadline. The implementation is very similar to Greedy 1.

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

Observation: Start with the schedules obtained by Greedy 2, and consider fixing the schedules of each problem separately. This can be modeled as a bipartite maximum matching problem, subdividing the available time and the required time of each task into individual milliseconds!

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

Observation: Start with the schedules obtained by Greedy 2, and consider fixing the schedules of each problem separately. This can be modeled as a bipartite maximum matching problem, subdividing the available time and the required time of each task into individual milliseconds!

Proof of claim: Starting with the matching obtained by Greedy 2, the Ford-Fulkerson algorithm can turn this into a maximum matching. Since this algorithm never unmatches a vertex, we never lose any of the overlap!

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

Observation: Start with the schedules obtained by Greedy 2, and consider fixing the schedules of each problem separately. This can be modeled as a bipartite maximum matching problem, subdividing the available time and the required time of each task into individual milliseconds!

Proof of claim: Starting with the matching obtained by Greedy 2, the Ford-Fulkerson algorithm can turn this into a maximum matching. Since this algorithm never unmatches a vertex, we never lose any of the overlap!

Running time: It suffices to run Greedy 1 and Greedy 2, giving a total running time of $\mathcal{O}(n \log n + m \log m)$.

Update: After the contest, Greedy 2 was proven to be incorrect. Fixing it requires much more complicated techniques.

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

Observation: Start with the schedules obtained by Greedy 2, and consider fixing the schedules of each problem separately. This can be modeled as a bipartite maximum matching problem, subdividing the available time and the required time of each task into individual milliseconds!

Proof of claim: Starting with the matching obtained by Greedy 2, the Ford-Fulkerson algorithm can turn this into a maximum matching. Since this algorithm never unmatches a vertex, we never lose any of the overlap!

Running time: It suffices to run Greedy 1 and Greedy 2, giving a total running time of $\mathcal{O}(n \log n + m \log m)$.

Update: After the contest, Greedy 2 was proven to be incorrect. Fixing it requires much more complicated techniques.

Extra challenge: The presented solution is non-constructive, but it is also possible to solve the problem constructively with the same asymptotic time complexity!

C: Circadian Rhythm

Problem author: Reinier Schmiermann

Claim: If both scheduling problems are possible, then it is also possible with the maximum overlap found by Greedy 2.

Observation: Start with the schedules obtained by Greedy 2, and consider fixing the schedules of each problem separately. This can be modeled as a bipartite maximum matching problem, subdividing the available time and the required time of each task into individual milliseconds!

Proof of claim: Starting with the matching obtained by Greedy 2, the Ford-Fulkerson algorithm can turn this into a maximum matching. Since this algorithm never unmatches a vertex, we never lose any of the overlap!

Running time: It suffices to run Greedy 1 and Greedy 2, giving a total running time of $\mathcal{O}(n \log n + m \log m)$.

Update: After the contest, Greedy 2 was proven to be incorrect. Fixing it requires much more complicated techniques.

Extra challenge: The presented solution is non-constructive, but it is also possible to solve the problem constructively with the same asymptotic time complexity!

Statistics: 4 submissions, 0 accepted, 2 unknown