AKHIL KUMAR

192211208

CSA0656

Minimum Time to Finish all Jobs

This presentation focuses on a common optimization problem: assigning jobs to workers to minimize the maximum working time. We'll explore how to achieve this using dynamic programming.

Problem Statement

We are given a set of jobs with their respective completion times, and a fixed number of workers. The goal is to assign each job to exactly one worker, ensuring that the maximum working time among all workers is minimized.

Input

An array 'jobs' representing the completion times of each job.

An integer 'k' representing the number of workers.

Output

The minimum possible maximum working time across all workers.

Dynamic Programming

Dynamic programming is a powerful technique for solving optimization problems by breaking them down into smaller, overlapping subproblems. This approach helps us efficiently build the optimal solution by leveraging previously calculated results.

1 **Bottom-Up**

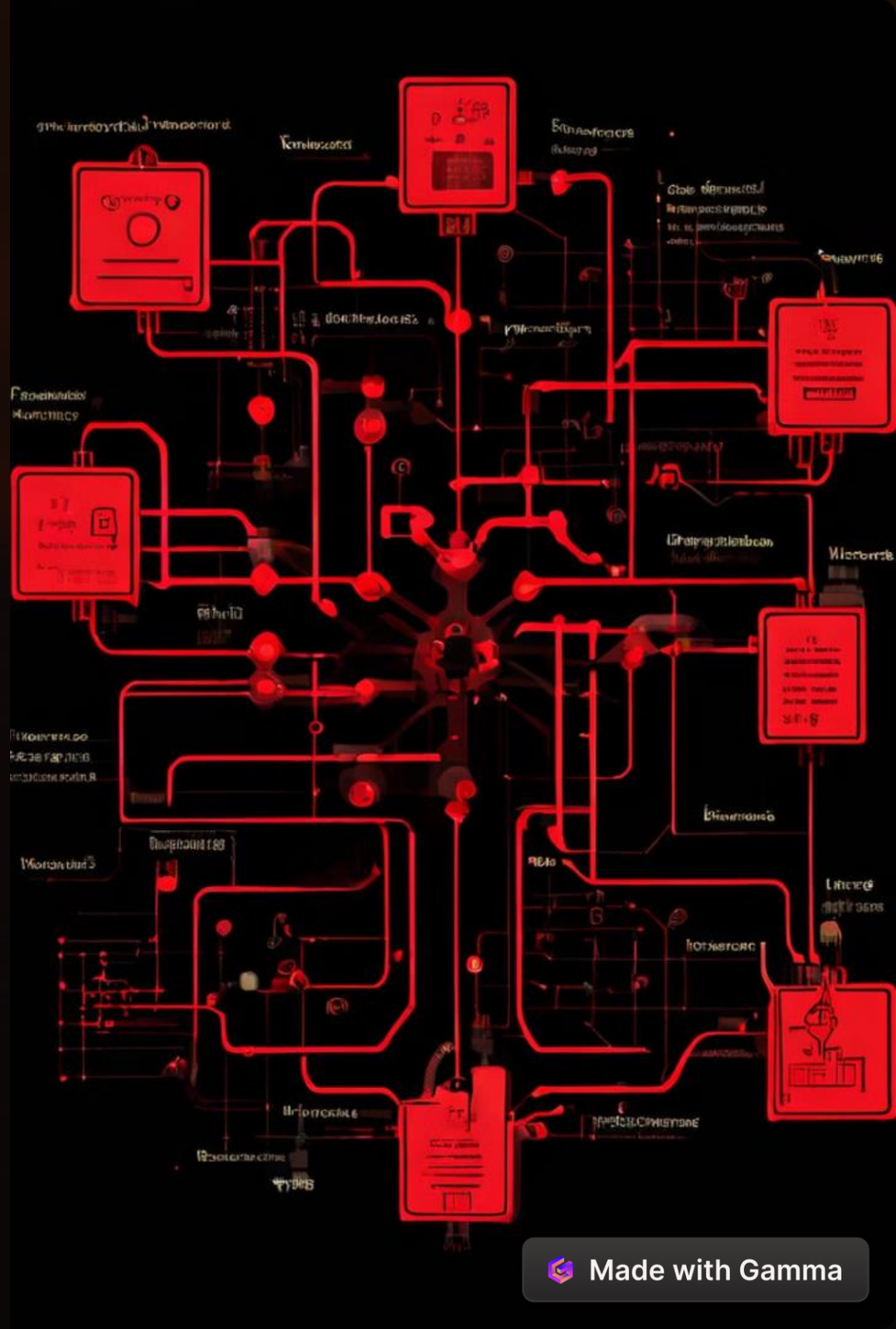
We start by solving the smallest subproblems and progressively build up the solution for larger subproblems.

2 **Memoization**

We store the results of previously solved subproblems to avoid redundant computations.

3 **Optimal Substructure**

The optimal solution for the entire problem can be constructed from the optimal solutions of its subproblems.



Identifying Subproblems

The subproblems in this problem involve assigning a subset of jobs to a subset of workers. We can define a subproblem as assigning the first 'i' jobs to 'j' workers.

Subproblem	Description
dp[i][j]	The minimum possible maximum working time for assigning the first 'i' jobs to 'j' workers

Recurrence Relation

The recurrence relation captures the relationship between the solution of a larger subproblem and its smaller subproblems. This helps us iteratively build the optimal solution.

1

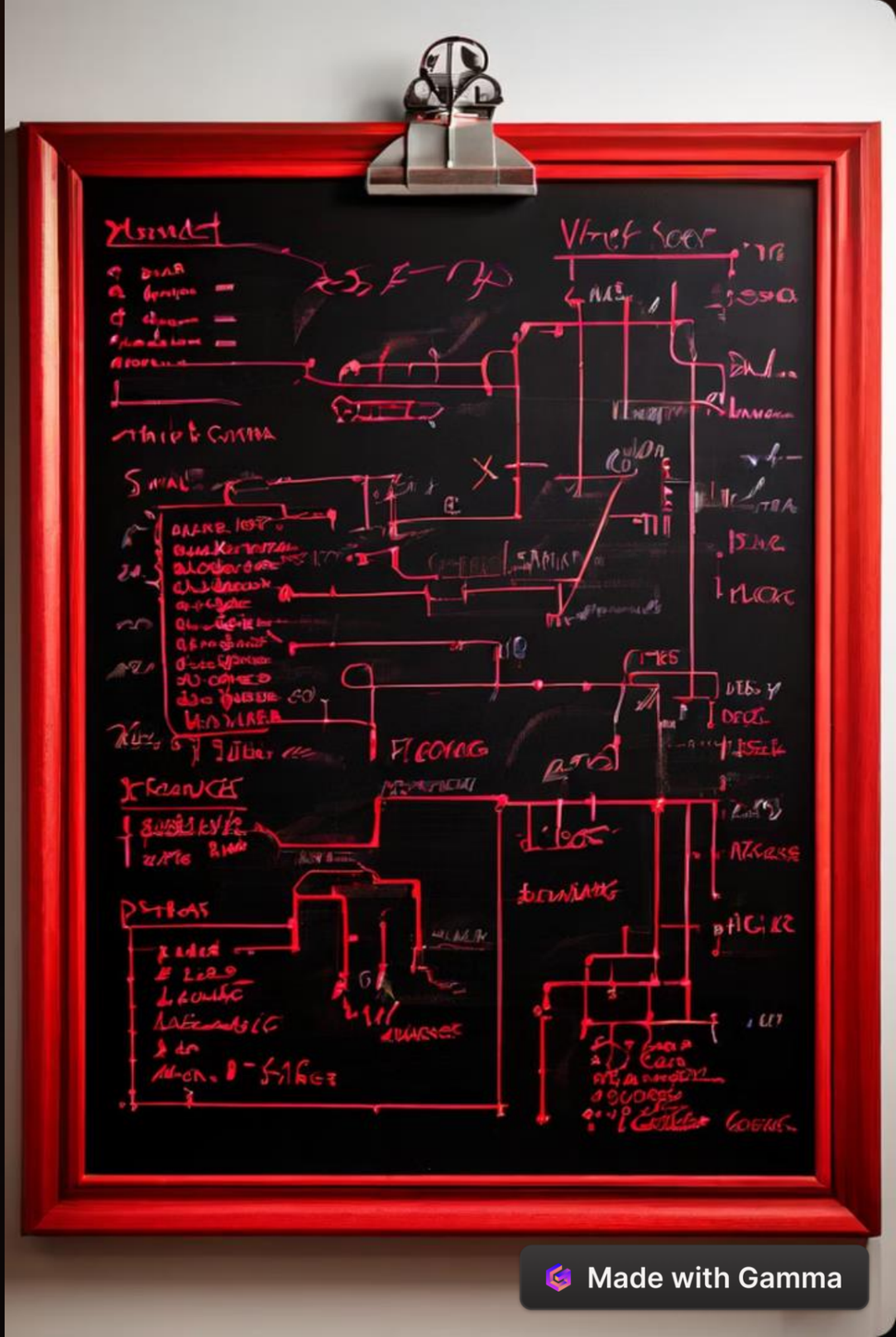
Base Case

dp[0][j] = 0, as no jobs need to be assigned.

2

Recursive Step

dp[i][j] = min(max(dp[i - 1][j], jobs[i - 1]), max(dp[i][j - 1], jobs[i - 1]))



Implementing the Solution

To find the minimum possible maximum working time for finishing all jobs with k workers, we use a combinatorial approach to optimize job assignments. The goal is to distribute jobs such that the maximum workload of any worker is minimized.

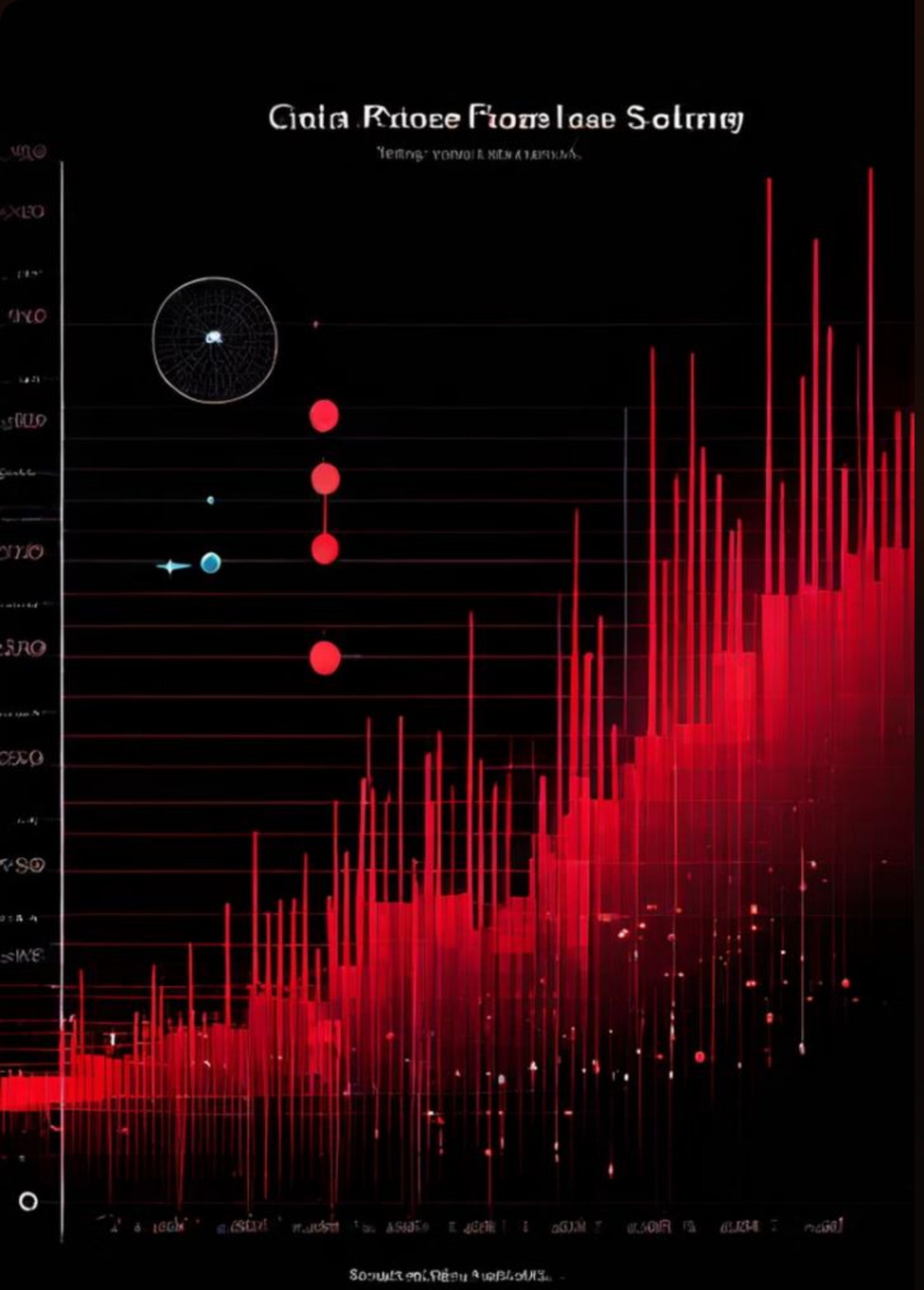
The solution involves exploring different ways to assign jobs to workers and calculating the resulting maximum working time for each assignment. By evaluating all possible assignments, we can identify the configuration that minimizes this maximum working time. This approach is feasible for small numbers of jobs due to its exponential time complexity.

In summary, the optimal assignment of jobs is achieved by exploring various combinations, with the final result being the minimal maximum working time across all feasible assignments. This method ensures that the workload is balanced among workers, leading to efficient job completion.

Step-by-Step Solution

- Identify All Occurrences of Each Job
- Initialize Worker Loads
- Sort Jobs in Descending Order
- Use Backtracking to Generate Possible Assignments
- Evaluate Assignments

1. Choose the Optimal Assignment



Time and Space Complexity

The algorithm's time complexity is O(n * k), where 'n' is the number of jobs and 'k' is the number of workers. This is because we iterate through all possible subproblems.

1 **Time Complexity**

O(n * k)

2 **Space Complexity**

O(n * k)

Conclusion

Dynamic programming is a powerful technique that allows us to solve complex optimization problems by breaking them down into smaller subproblems and efficiently building up the optimal solution.

Efficiency

Dynamic programming provides an efficient solution by avoiding redundant computations.

Optimization

This approach can be applied to various optimization problems where the optimal solution can be derived from the optimal solutions of its subproblems.

Implementation

The dynamic programming solution can be implemented using a table or memorization techniques.

THANK YOU