5.3 MINIMIZE MAXIMUM WORKING TIME AMONG WORKERS

Question:

You are given an integer array jobs, where jobs[i] is the amount of time it takes to complete the ith job. There are k workers that you can assign jobs to. Each job should be assigned to exactly one worker. The working time of a worker is the sum of the time it takes to complete all jobs assigned to them. Your goal is to devise an optimal assignment such that the maximum working time of any worker is minimized. Return the minimum possible maximum working time of any assignment.

AIM

To implement a dynamic programming solution in Python that minimizes the total production time across three assembly lines, considering station times, transfer times, and task dependencies.

ALGORITHM

1. Sort jobs in descending order to assign larger jobs first (helps prune faster).
2. Use a recursive function dfs(i) to assign the i-th job to one of the k workers.
3. Track current workload of each worker in workers[].
4. At each step, try assigning the job to each worker:
    - Skip if assigning exceeds current best known answer.
    - Skip duplicate states (e.g., assigning to two workers with same load).
5. Update the global minimum when all jobs are assigned.
6. Return the minimum possible maximum workload.

PROGRAM

```
def minimum_working_time(jobs, k):
    jobs.sort(reverse=True)
    ans = sum(jobs)

    def backtrack(i, workers):
        nonlocal ans
        if i == len(jobs):
            ans = min(ans, max(workers))
            return
        seen = set()
        for j in range(k):
            if workers[j] in seen:
                continue
            seen.add(workers[j])
            workers[j] += jobs[i]
            if workers[j] < ans:
                backtrack(i + 1, workers)
            workers[j] -= jobs[i]

    backtrack(0, [0] * k)
    return ans

jobs = list(map(int, input("Enter job times separated by space: ").split()))
k = int(input("Enter number of workers: "))
print("Minimum possible max working time:", minimum_working_time(jobs, k))
```

Input:

Enter job times separated by space: 1 2 4 7 8

Enter number of workers: 2

Output:

```
Enter job times separated by space: 1 2 4 7 8
Enter number of workers: 2
Minimum possible max working time: 11
>>>
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: $O(k^n)$, but pruned significantly by sorting and skipping redundant states.
- Space Complexity: O(k), for tracking worker loads.