Date: 10 - 02 - 2026
Assignment No.: 2

Name: Varghese K James
Reg No.: 3122245001192

Assignment 3: Algorithms Using Backtracking

Competitive Programming

Question 1

Aim: You are given an integer array of 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.

Algorithm

MinTime Backtrack (job[0...n-1], k)
//Input: jobs[0..n-1] of n positive integers & k.
//Output: min of max workload of any worker

```
workers[0..k-1) ← [0, 0, ..., 0]
res ← ∞

Backtrack (0, res)
return res

Backtrack (cur-job, res)
    if cur-job = n then
        res ← min (res, max (worker))
        return
    seen ← ϕ
    for i ← 0 to k-1 do
        if workers[i] ∈ seen then
            continue
        if worker[i] + job [cur-job] ≥ res then
            continue

        add worker[i] to seen
        workers[i] ← workers[i] + job [cur-job]

        Backtrack (cur-job + 1, res)

        workers[i] ← worker[i] − jobs [cur-job]
```

Code

```
class Solution:
    def minimumTimeRequired(self, jobs: List[int], k: int) -> int:
        workers = [0]*k

        self.res = sys.maxsize

        def backtrack(cur_job):
            if cur_job == len(jobs):
                self.res = min(self.res, max(workers))
                return

            seen = set()
            for i in range(k):
                if workers[i] in seen: continue
                if workers[i] + jobs[cur_job] >= self.res: continue
                seen.add(workers[i])
                workers[i] += jobs[cur_job]
                backtrack(cur_job+1)
                workers[i] -= jobs[cur_job]

        backtrack(0)
        return self.res
```

Time complexity

Recurrenc relation is:

$$T(j) = k \cdot T(j-1) + c$$

By using substituition method,

$T(n) = k \cdot T(n-1) + c$

$$T(n) = k[k \cdot T(n-2) + c] + c = k^2 T(n-2) + kc + c$$

$$T(n) = k^2[k \cdot T(n-3) + c] + kc + c = k^3 T(n-3) + k^2 c + kc + c$$

$$T(n) = k^n T(0) + c \sum_{i=0}^{n-1} k^i$$

Date: 10 - 02 - 2026
Assignment No.: 2

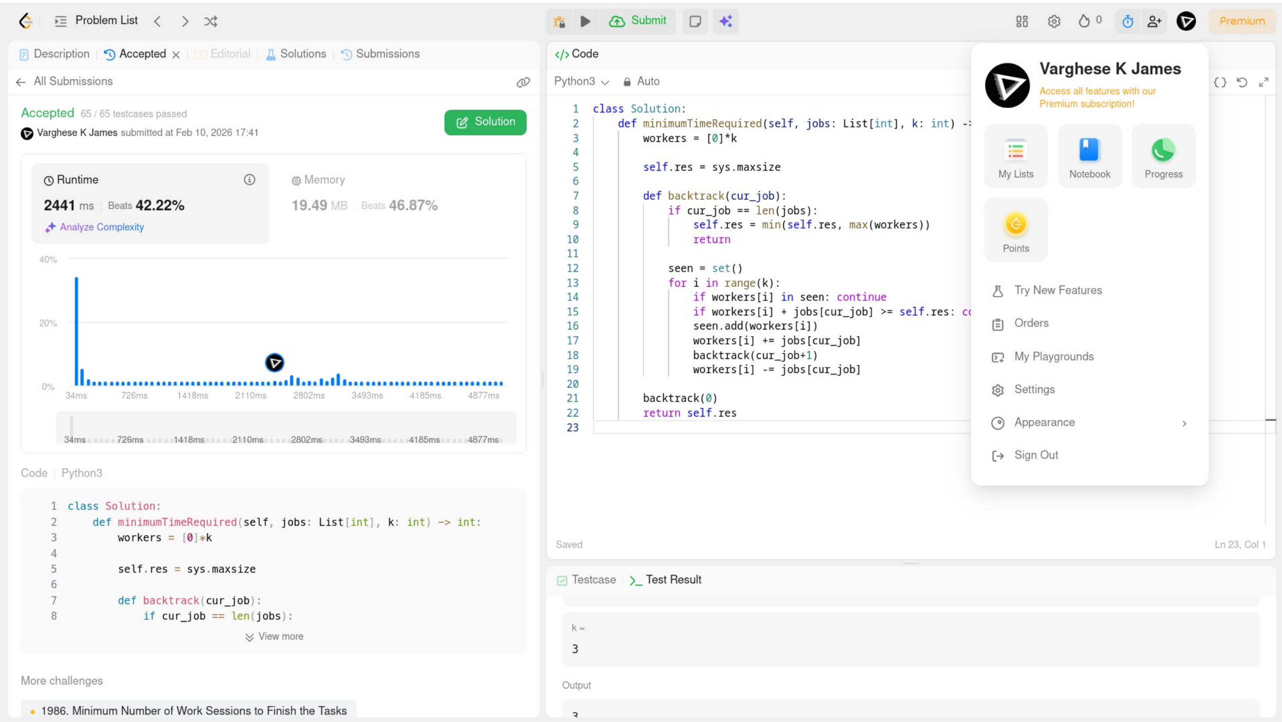
Name: Varghese K James
Reg No.: 3122245001192

$$T(n) = k^n \cdot k + c\left(\frac{k^n - 1}{k - 1}\right)$$

Using asymptotic notation it becomes k^n, hence:

$$O(k^n)$$

Screenshot of Output



Learning Outcomes

- Learned about using backtracking algorithms to solve questions involving permutations and combination
- Learned about optimisations done to solve competitive programming questions.