Programming Assignment 2
Design and Analysis of Algorithms

Trasula Umesh Karthikeya
22B0913

1 Code Explanation

Given question is to find minimum memory peak for all permutations of given processes possible and formulae for memory peak management is as follows:

$$\max_{k \in \{1,2,\ldots,n\}} \left\{ m\left(i_k\right) + \sum_{1 \leq p < k} m\left(i_p, i_k\right) + \sum_{k < q \leq n} m\left(i_k, i_q\right) + \ sum_{1 \leq p < k} \sum_{k < q \leq n} m\left(i_p, i_q\right) \right\}.$$

Let's first find time complexity for calculating memory peak for a particular permutation,

For calculating $m\left(i_k\right)$ we need $O(1)$ time complexity.

For calculating

$$\sum_{1 \leq p < k} m\left(i_p, i_k\right)$$

we need $O(n)$ time complexity.

For calculating

$$\sum_{k < q \leq n} m\left(i_k, i_q\right)$$

we need $O(n)$ time complexity.

For calculating

$$\sum_{1 \leq p < k} \sum_{k < q \leq n} m\left(i_p, i_q\right)$$

we need $O(n^2)$ time complexity.

Therefore for calculating

$$\max_{k\in\{1,2,\ldots,n\}}\left\{m\left(i_k\right)+\sum_{1\le p<k} m\left(i_p,i_k\right)+\sum_{k<q\le n} m\left(i_k,i_q\right)+\sum_{1\le p<k}\sum_{k<q\le n} m\left(i_p,i_q\right)\right\}.$$

we need at least $O(n^3)$ time complexity.

If we do brute force by calculating all $n!$ permutations then overall time complexity for solving the question would be $O(n!\cdot n^3)$ But we can optimize this to $O(2^n\cdot n^3)$ time complexity by using dynamic programming and bit mappings.

Here I am using idea of bit mapping, if a number k is n-bit format such that k = 001010..0101 (n-bits) then k contains the processes numbers where 1 is present in that n-bit format of k from left side. For example let k = 3 then k = 00000...011 here 1 is present at 1st and 2nd positions of the n-bit format of k from left side, therefore my subset contain only processes 1, 2

If a k is number such that $0 <= k < 2^n$ and k is written in n-bit format then my dp with this number k as the index represents the minimum memory peak for all permutations of the processes in the subset that k contains. Hence $dp[2^n-1]$ represents the minimum memory peak for all permutations of the n processes and that is our answer.

To find $dp[m]$ that is minimum memory peak for all combinations of the processes that subset m contains. Let subset m contain processes $a_1, a_2, \ldots a_j$. Now what I am doing is finding the k^{th} process of the processes that subset m has by maintaining a_i as the last process where $1 <= i <= j$. Now I am assuming that my k^{th} process is a_i and finding the memory peak. Here I am using the fact that memory peak of the k^{th} process is same for all permutations of the processes until we change the relative position of a process with the $ikth$ process. i.e, memory peak of kth process is

$$m\left(i_k\right)+\sum_{1\le p<k} m\left(i_p,i_k\right)+\sum_{k<q\le n} m\left(i_k,i_q\right)+\sum_{1\le p<k}\sum_{k<q\le n} m\left(i_p,i_q\right)$$

Now in the subset k i have removed i^{th} process and calculated memory peak by assuming k^{th} process as i^{th} process. Now to find the K^{th} process for the subset k we need find $max(dp[l],\ \ calculated\_sum)$ where l represents the subset of all processes except i^{th} process. And the I will update the dp as minimum of the present memory peak and previous dp of the subset k.

I am also avoiding the pairs where j^{th} process comes after i^{th} and i^{th} process depends on j^{th} process.

Therefore overall time complexity is $O(2^n\cdot n^3)$

Now I have optimized it to $O(2^n \cdot n^2)$ by pre computing the terms

$$\sum_{1 \leq p < k} \sum_{k < q \leq n} m\left(i_p, i_q\right)$$

Therefore now overall order is $O(2^n \cdot n^2)$