1. Recurrance relation : $T(n) = 2T(n-1)+1$ $T(0)=1$

$T(n) = 2T(n-1)+1$

$= 2.(2(T(n-1))+1)+1 = 4T(n-1)+2+1$

$\vdots$

$2^k T(n-k) + 2^{k-1} + 2^{k-2} \dots 1$

$n-k=0$ $k=n$ $2^{k-1}+2^{k-2} \dots 1 = \frac{2^{k-1}(1-\frac{1}{2}^k)}{1-\frac{1}{2}} = 2^k-1$

$2^n + 2^n - 1 = \underbrace{\Theta(2^n)}$

The function I provided as python code basically create all possible subsets and cheks the result( calc_discount = O(1)). The time complexity is the same for best, worst and average case because this is exhaustive algorithm and we check all possible scenerious.

Best case : $\Theta(2^n)$

Wors case : $\Theta(2^n)$

Average case : $\Theta(2^n)$

2. This algorith needs to check for every possible user, process, processor triple (Which is quite exhaustive search). I gave process processor cost matrix for every user in the code. These random generated matrices will be the individual costs for every users. Algorithm finds the best possible combination using these matrices. Here are the time complexities.

Best case: $n.n^n$

Worst case: $n.n^n$

Average case: $n.n^n$

3. The logic of this algoritm is we need to explore all possible sceneriious and find minimum cost. So the best, wors and the average case will be the same.
The algorithm tries n part for the firs position n-1 for the second etc.
So:

Best case: $\Theta(n!)$

Worst case: $\Theta(n!)$

Average case: $\Theta(n!)$

4. The logic of this algorithm similar to the firs one. We need to check all possible combinations so that we can find minimum coin required (exhaustive search).
For a set which has n elements; we will have $2^n$ subset. The algorithm also does the same approach. I provided set of coins. The algoritm tries all possible subsets of this coin set. Compares the results (number of coins used) and prints the best set.

Average case: $(2^n)$ Best case: $(2^n)$ Worst Case: $(2^n)$

5.

The recurrence relation: $T(n) = 2T(n/2) + O(1)$

$T(n) = 2T(n/2) + 1$ $\qquad$ $T(1) = 1$

$= 2(2T(n/4) + 1) + 1 = 4T(n/4) + 2 + 1$

$= 2(2(2(T(n/8) + 1) + 1) + 1 = 8T(n/8) + 4 + 2 + 1$

$\vdots$

$2^k \cdot T(n/2^k) + 2^{k-1} + 2^{k-2} \dots 1$

$2^k = n \qquad k = \log_2 n \qquad 2^{k-1} + 2^{k-2} \dots 1 = 2^{k-1}\dfrac{(1 - \frac{1}{2}^k)}{1 - \frac{1}{2}} = 2^k - 1$

so, $2^k \cdot T(n/2^k) + 2^{k-1} + 2^{k-2} \dots 1 \rightarrow 2^{\log_2 n} + 2^{\log_2 n} + 1 = 2n - 1 = \Theta(n)$

Average Case Complexity = $\underline{\Theta(n)}$