Date No.

4. To find the approximation algorithm,
(1) sort both sets from greatest to lowest value.
(2) select the maximum number in set A, which is the people who owe money, and pay it to the people with maximum money owed in set B.
(3) Keep iterating until the remaining value in both sets are 0.

Running time: Let k be the larger size of both sets, which is $\max(n,m)$.
The running time is $k(\log k)$, for sorting.

Approximation ratio:
The optimal solution is k transactions, with $k = \max(n,m)$. this is because every people who owe money has to pay at least once, and every people who are owed should receive at least once.
The greedy algorithm sorts both sets from largest to smallest. At the first iteration, the people who pay the most is selected, and pay the people who owed the most. 3 cases are possible:
(1) He fully settled his debt, but the people who are owed still need more money before he fully claim his money. He claim from others.
(2) He paid the other people, the other people fully claimed, but the people who is paying is still left with some money. He pay others.
(3) Both sides fully settled their debts, and one has nothing left to pay, and the other has nothing left to claim. This is the optimal case.

We can observe that in 3 cases, either one side or both sides are reduced to 0. the algorithm runs until all values are 0, which is at most $(n+m-1)$ times, which at the last iteration both values are reduced to 0.
Since k is the max of (n,m), 2 cases are possible:

$k=n, n>m$	$k=m, m>n$
$n+m-1$	$n+m-1$
$< n+n-1$	$< m+m-1$
$< 2n$	$< 2m$

∴ Therefore the greedy algorithm has approximation ratio of at most 2.