Input: *dataset*: array of numbers
Input: *t*: max partition size
Output: minimum inequality score for an array of numbers

```
Algorithm: EP
n = size of dataset - 1
ep = Array(n,n)
Initialize ep to infinity
return MemoEP(dataset, t, 0, n)
```

Input: *datasetC* : array being passed, will change per recursive call
Input: *startingIndex:* used to keep track of what row of ep is being manipulated
Input: *n*: size of original array
Output: minimum value in the *startingIndex* row of the array *ep*

```
Algorithm: MemoEP(dataset, t, startingIndex, n)
int nextIndex = startingIndex;
int sum = 0;
if startIndex <= n
  | for i=0 to the end of the someArray
  |  |   sum += datasetC[i]
  |  |   if(sum <= t)
  |  |    |  if ep[startingIndex][0...n] all != infinity
  |  |    |   |
  |  |    |   |   return min ep[startingIndex][0....n]
  |  |    |   |
  |  |    |  else
  |  |    |   |
  |  |    |   |   ep[startingIndex][nextIndex] =
  |  |    |   |   POW((tsum(datasetC[startingIndex...nextIndex]), 2)
  |  |    |   |   + MemoEP(datasetC[i+1...n], t, nextIndex + 1, n)
  |  |    |   |
  |  |    |  end
  |  |   else
  |  |    | ep[startingIndex][nextIndex] -= 1
  |  |   end
  |  |   nextIndex++
  | end
else
 |   return 0;
end
return min ep[startingIndex][startingIndex...n];
```