Part 2

# Scenario

Your company and team have most of their projects in C#. You are hitting some performance issues with a key algorithm the company has implemented in a C# project. The algorithm is being CPU bound, you have tracked the problem down to some hot spots of code and believe the bottleneck is in the garbage collector wasting a lot of time on relatively short lived objects.

# Question

Assuming the business need was to have the algorithm running as fast as possible please discuss at least one potential approach / solution to this situation that you might take.

Please explain the potential costs/risks and benefits associated with the approach(s) and what assumptions they make / which situations they might fit best.

We may use three or four potential solutions:

1) Help the garbage collector: nullify all the objects after their usage, in order to have the least references to them active. This approach may help the GC to understand which objects are to be collected, and therefore get to run faster. It holds the lower risk on its implementation, but also the benefits may also be marginal, unless the short lived objects hold a lot of references to them and maybe some “island” references, which normally are the most time consuming for the GC to detect them.

2) Small code refactoring to avoid so many short-lived objects, in a way of reusing some of those variables between algorithm runs: This approach may give some medium to big improvements, depending on the possibility of making such modifications more or less deeply in the algorithm. It also holds higher risk than option 1, being higher the more we refactor the code to reuse variables. In this case, a correct variable cleanup procedure is a must to avoid problems.

3) Full algorithm rewrite: This one is highly costy, unless we are talking about a simple algorithm that is not very well optimized. It also carries one of the higher risks with the change, but in algorithms that are called in a high number of times (i.e., functions that are recursively called by another algorithms) it may worth doing it.

4) Full/partial algorithm rewrite in another language: Typically the solution with highest cost and risk, but also one that may optimize every aspect of the algorithm. For instance, writing the same algorithm in C++ would take completely apart the GC problem, given we have manual memory managing. However, doing this kind of action is not advisable unless we have critical performance requirements that are not covered using the solutions above.