Ali Awari – aka87

Take a list of numbers like S = (1, 6, 2, 9, 3) - this list is /not/ sum free, because it contains 1,2,3 and 1 + 2 = 3 ( as well as 3, 6, 9, and 3+6 = 9).  
  
Given a list of numbers, how could you determine whether it is sum-free?

[ Assume you are only worried about two numbers adding up to a third. ]

1. **Given a list of numbers, how could you determine whether it is sum-free?**

You would take the first number in the list, add it to the second, then search for that sum inside the list.

1. **What is the complexity / run time of your approach? What is its big-Oh?**

There will be N!/(n – 2)! different sums, it will take N searches in an unsorted list therefore: N!/(N – 2)!\*N

1. **If the list is sorted, does that simplify things at all?**

Yes it will, the search time will be reduced to log(n), the complexity will be: N!\*log(n)

1. **Is it worth sorting?**

If the number of searches is equal to or exceeds the amount of elements in the list you should sort. Since there will be N searches, you should sort.

1. **What if you wanted to generalize to 3-sums, 4-sums, etc, or arbitrary sized sums?**

**[ example 6 + 2 + 1 = 9 above ]**

You would use the combination formula to determine the amount of sums, with r being the amount of numbers included into the sum N!/(N – r)!