Notes for the Analyze Project

1. Piecemeal growth. Get little pieces to work, one at a time
2. You may start with loops: If you don't know how to do it with an algorithm, do it with a loop first.
3. To smooth using a transform, you have to get the index ("iter") in the example. If you use a lambda expression, pass the integer by reference. Then you can get the address of it, and assign it to a pointer (something like: (int& i) { int* iter = &I; …} Then you can use the pointer as if it's an array, as in the example smoothing code. (If you don't use a lambda, it's the same thing.)
4. You do a lot with the index, or the position number. When you first find the index of the start of each pulse, you can do it with a for_each, but you have to go through some gyrations to get the index. But if you use a for-loop, you have the index already. So USE A FOR LOOP. The code is cleaner.
5. I found it useful to build a little struct that holds things like the index and value of the start of a pulse, and the same thing for its peak.
6. I found it useful to first find all the possible pulses, and then throw away the ones that happen before the peak of the previous one. To do so, I used uniq – but note that uniq reorders, but does not resize.
7. In finding piggyback pulses, since you are looking at adjacent pulses, I used adjacent_find, because it gives me both pulses. (Just have it return false always.)
8. Don't hard-code path names ("." Should work)