STACK OF PAIRS OF PACKETS

We store references in the stack. We derive an object and own it in the same stack. Making copies of all objects in the stack is costly.

We derive a value into a list of a single element sometimes. This is according to the requirements.

My implementation without CoW is making copies of values in the tight loop. This made the solution take a long time to finish.

Profiling shows my code is spending a lot of time destroying objects. This was happening in the tight loop.

CoW is able to store the reference to an object. But it can also become the owner of a copy of the object. It can instruct a CoW to construct and own an object as well.

It's the first and third use cases that would come in handy for implementing the stack.

I can implement a CoW as a variant of the type and its reference type. The size of a CoW object would be the size of the type plus the size of the metadata.

Storing references in CoW requires use of std::reference_wrapper.

Alternatively, objects and references could live in the stack, wrapped in shared_ptr. Objects will reside in the heap. Their copies will not be necessary. Destroying the shared pointers will be cheap until the final references leave the scope.

An added bonus is I save time implementing CoW. But there may be an overhead of keeping extra copies of references in memory. This is probably not very big.

WATERFALLS CAVE

How easy is it to visualise the state of the cave at each stage of the algorithm?

How useful, or important?