Why performance engineering? :: Performance is the **currency** of programming

- what software properties are more important than performance?: modularity, correctness, clarity, reliability, etc.
- With more currency, you can trade for more things. Performance makes possible create this trade offs without sacrificing usability.

Bentley optimization rules

Data structures

Packing & Encoding

Packing:: Store more than one data value in a machine word.
Encoding:: Convert data values into a representation requiring fewer bits.

It depends on the model representation you are using for your data structure: You have to choose wisely the DT you are using. For example, for storing dates you can use strings as "2021, February 12" which it requires 17 bytes, or you can use an int representation for the whole date from 4096 B.C to 4096 A.C can be stored in 3 times less space than using the string approach. The problem we can have with this is that the decoding approach also requires more power. Some times, packing/encoding are the problem that can be optimized.

Augmentation :: Add information to a data structure to make common operations do less work.

- Precomputation:: Perform calculations in advance to avoid doing them at "mission-critical" times.
- Compile-time initialization
- Meta-programming
- Caching:: Store results that have been accessed recently so that the program need not to compute them again.
- Sparcity:: Don't compute zero values.
- Compressed Sparced Row

Logic

- **Constant folding and propagation**:: Evaluate constant expressions and substitute the result into further expressions, all during compilations. Compiler sometimes does this optimization for us.
- **Common-subexpression elimination**:: Avoid computing the same expression multiple times by evaluating the expression once, and store the result for later use. Compiler sometimes does this optimization for us.

```
From:
- a = b + c
- b = a - d
- c = b + c
- d = a - d

To:
```

```
- a = b + c
- b = a - d
- c = b + c
- d = b
```

Algebraic identities:: Replace expensive algebraic expressions with simpler equivalents that require less work.

Example: When checking for collisions we can use the Pythagorean's theorem so that $h = sqrt(x\_0 - x\_1)^2 + (y\_0 - y\_1)^2 + (z\_0 - z\_1)^2$ but as mentioned in the class, the square root operation is a relative expensive operation. One simple thing we can use is using the identity $sqrtuleqv\$isthesamethan\$uleqv^2$, so we can use the expression $h^2 = (x\_0 - x\_1)^2 + (y\_0 - y\_1)^2 + (z\_0 - z\_1)^2$ instead.

- **Short-Cirtuiting**