Runtime of an unsuccessful search for a vector of size $n$

- Unoptimized `cphstl::find()`
- Optimized `cphstl::find()`
- `std::find()`

![Graph showing execution time per element vs. input size](image-url)
Execution time per element [nanoseconds]

Runtime of an unsuccessful search for a vector of size $n$

Unoptimized `cphstl::find()`
STD `std::find()`
Optimized `cphstl::find()`
Partial Specialization

template <typename T>
class X {  
    // Most general version.  
}

template <typename T>
class X<T*> {  
    // Version for general pointers.  
}

template <>  
class X<void*> {  
    // Version for one specific pointer type.  
}

One specialization is more specialized than another if every argument that matches its specialization also matches the other, but not vice versa. The most specialized version is preferred over the others in declarations and in overload resolution.
Example from Bjarne’s book

```cpp
#include <typeinfo>
#include <iostream>
#include <complex>

template <typename T>
T sqrt(T x) {
    std::cout << typeid(x).name() << std::endl;
}

template <typename T>
std::complex<T> sqrt(std::complex<T> x) {
    std::cout << typeid(x).name() << std::endl;
}

double sqrt(double x) {
    std::cout << typeid(x).name() << std::endl;
}

int main() {
    std::complex<double> z;
    sqrt(2);
    sqrt(2.0);
    ::sqrt(z);
}

/*
Output:
shell> g++ sqrt.c++
shell> ./a.out
i
d
St7complexIdE

Without "::"s:

sqrt.c++: In function ‘int main()’: 
sqrt.c++:23: error: call of overloaded ‘sqrt(std::complex<double>)’ ambiguous
*/
```

Performance Engineering Laboratory 1