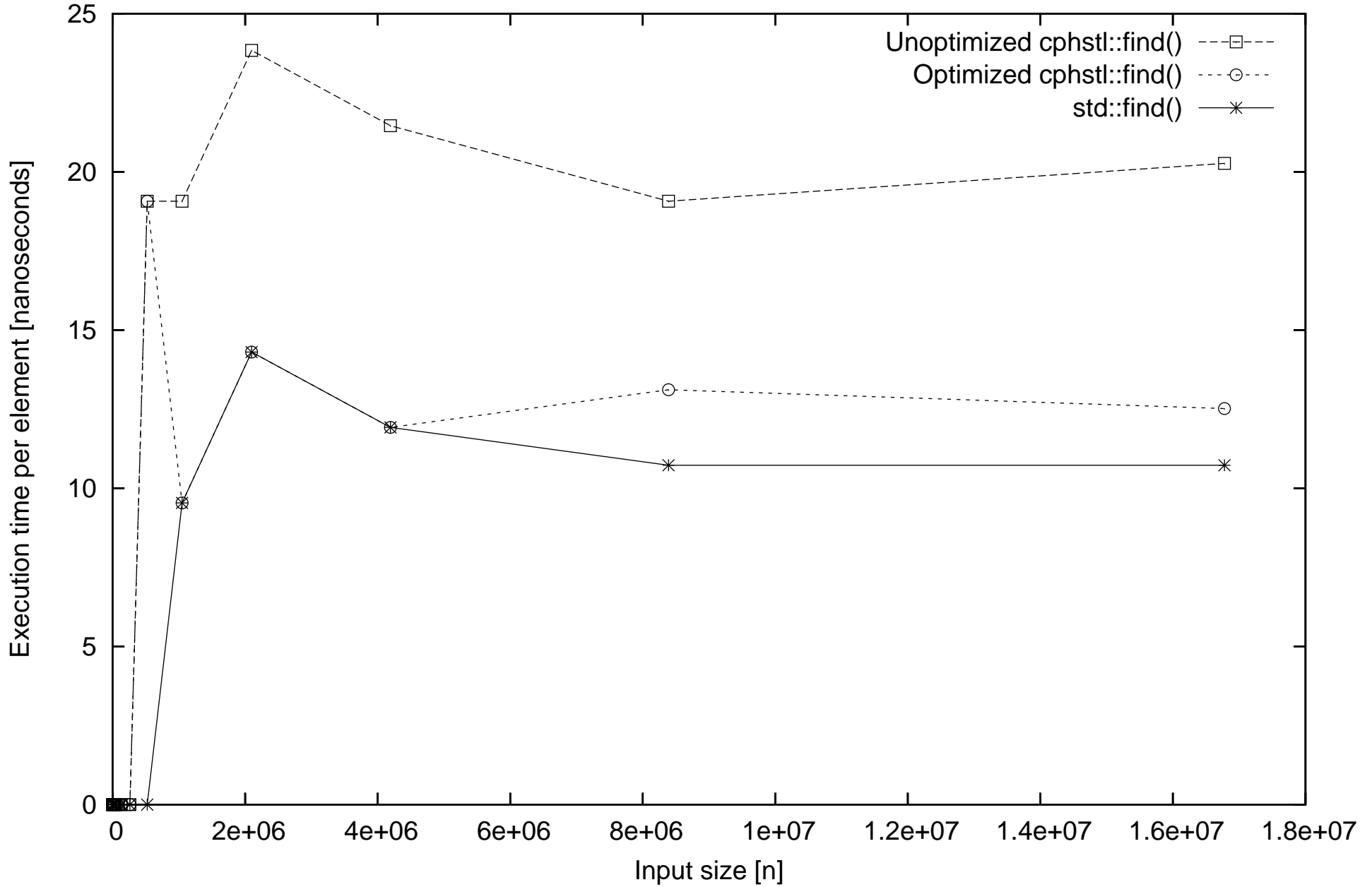
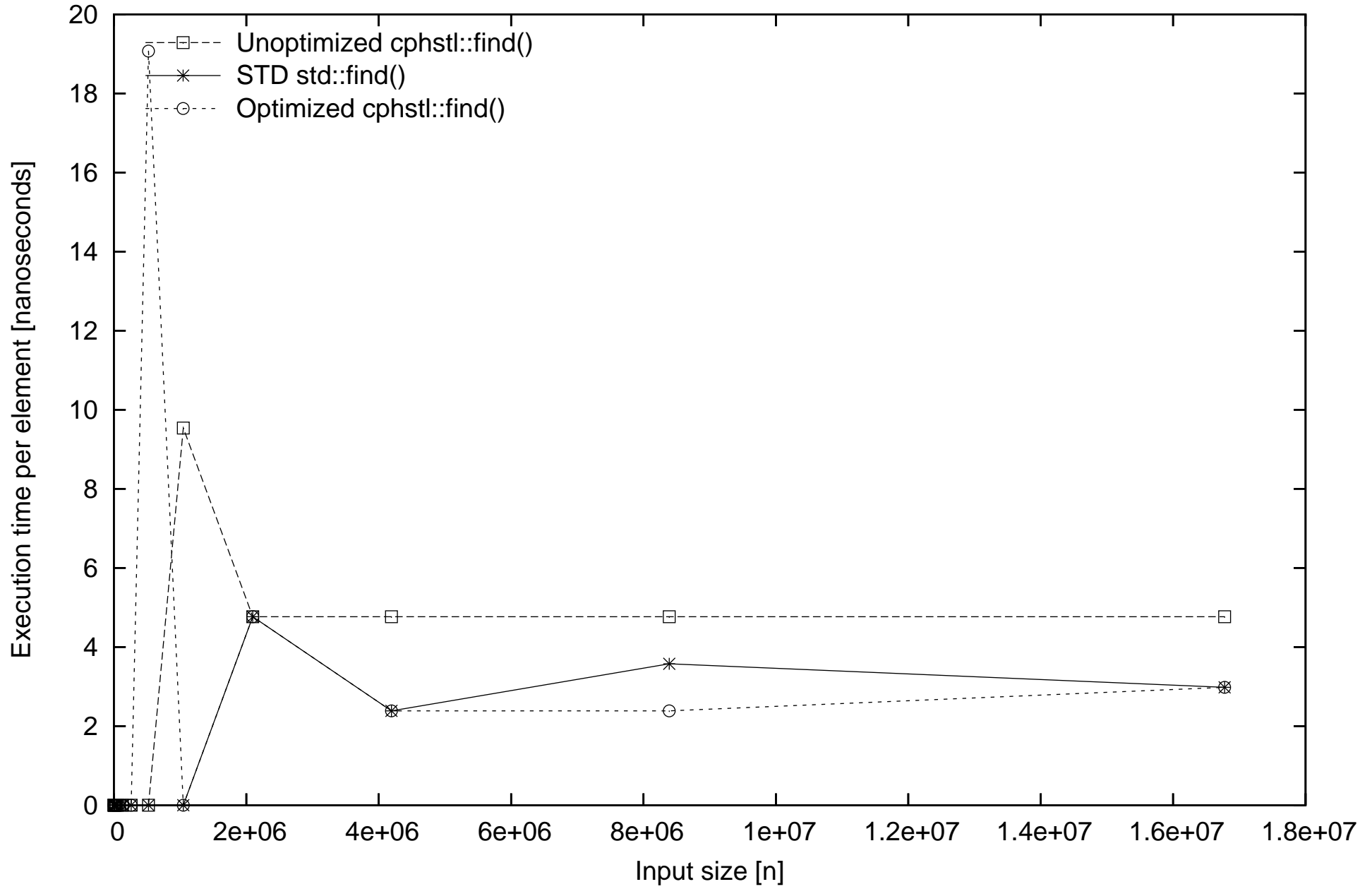


Runtime of an unsuccessful search for a vector of size n



Runtime of an unsuccessful search for a vector of size n



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

```
#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.cpp
```

```
shell> ./a.out
```

i

d

```
St7complexIdE
```

Without "::"s:

```
sqrt.cpp: In function 'int main()':
```

```
sqrt.cpp:23: error: call of overloaded 'sqrt(std::complex<doub
ambiguous
```