There's Treasure Everywhere

Prepared for ACCU 2016

Andrei Alexandrescu, Ph.D.

2016-04-21

Working Title

How to Write and Deliver a Keynote

What’s (in) a Keynote?

- Friend 1: “Just talk about things you like.”
- Friend 2: “Should have no code and no math.”

Things I Like

Things I Like

- Confusing aphorisms

Confusing Aphorisms

When you hear hoofbeats, think horses, not zebras.

Confusing Aphorisms

When you hear hoofbeats, think horses, not zebras.

—Not an African Proverb

Things I Like

- Confusing aphorisms

Things I Like

- Confusing aphorisms
- Fast code

Things I Like

- Confusing aphorisms
- Fast code
- C++ Concepts

Toto Washlet

The Toto Company

- Offers 8 washlet models

The Toto Company

- Offers 8 washlet models
- Of these, 7 have a remote control

The Toto Company

- Offers 8 washlet models
- Of these, 7 have a remote control

- A *remote control*

Washlets with Remotes

- Combine two nice things
- Complex and interesting
- Modern and cool

- Unlikely to be super useful in practice

“Why Concepts didn’t make C++17”

- Not enough time

“Why Concepts didn’t make C++17”

- Not enough time
- “The Concepts TS does not specify any concept definitions”

“Why Concepts didn’t make C++17”

- Not enough time
- “The Concepts TS does not specify any concept definitions”
- “Use of concepts sometimes results in worse error messages”

Upcoming

“Why Concepts did make C++20”

Sentinels

Let's talk about linear find

- Nothing out of the ordinary:

```
Range find(Range, E)(Range r, E e) {
  for (; !r.empty; r.popFront) {
    if (r.front == e) break;
  }
  return r;
}
```

- Implemented many times in many languages

To index or not to index

- Indexed access and pointer arithmetic changed relative speed over years
- Today: indexed access has a slight edge
  - Less aliasing
  - CPU has sophisticated indexed addressing

Linear find using random access

```
Range find(Range, E)(Range r, E e) {
  size_t i = 0;
  for (; i != r.length; ++i) {
    if (r[i] == e) break;
  }
  return r[i .. $];
}
```

- Task: make it faster

Linear find with sentinel

```
Range find(Range, E)(Range r, E e) {
  auto c = r[$ - 1];
  r[$ - 1] = e;
  size_t i = 0;
  {
      scope(exit) r[$ - 1] = c;
      for (;; ++i)
        if (r[i] == e) break;
  }
  if (i + 1 == r.length && c != e)
    ++i;
  return r[i .. $];
}
```

But. . . but. . .

- That doesn't work for all types!
- `r[$ - 1] = c` may throw
- Also, `find` itself doesn't work for all types
  - Need to define only when we should
- Need to choose "right" implementation automatically
  - With sentinel for certain types
  - Conservative for the others

Linear find with sentinel, take 2

```
Range find(Range, E)(Range r, E e)
if (isInputRange!Range &&
  is(typeof(r.front == e) == bool)
) {
  ...
}
```

- `is(typeof(e) == T)` Does expression `e` "work" and has type `T`?
- `e` must *look* like an expression syntactically
- Boolean returned, compilation not stopped
  - SFINAE on steroids
- Still, how to choose with/without sentinel?

static if to the rescue

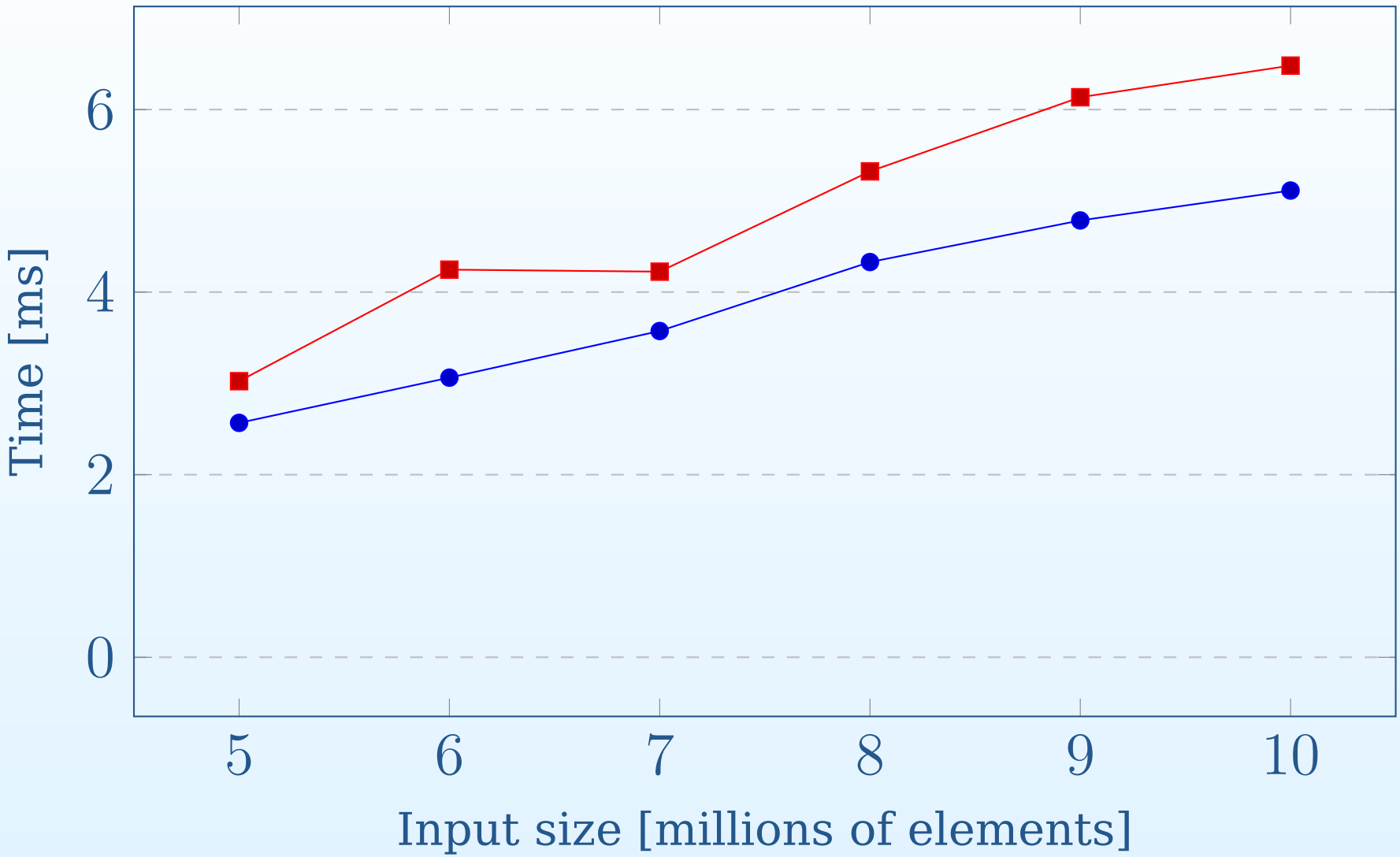
```
Range find(R, E)(R r, E e)
if (isInputRange!R && is(typeof(r.front == e) : bool)) {
  static if (isRandomAccessRange!R && hasSlicing!R) {
    static if (is(typeof(
      () nothrow { r[0] = r[0]; }
    ))) {
      ... sentinel implementation ...
    } else {
      ... indexed implementation ...
    }
  } else {
    ... conservative implementation ...
  }
}
```

Please note

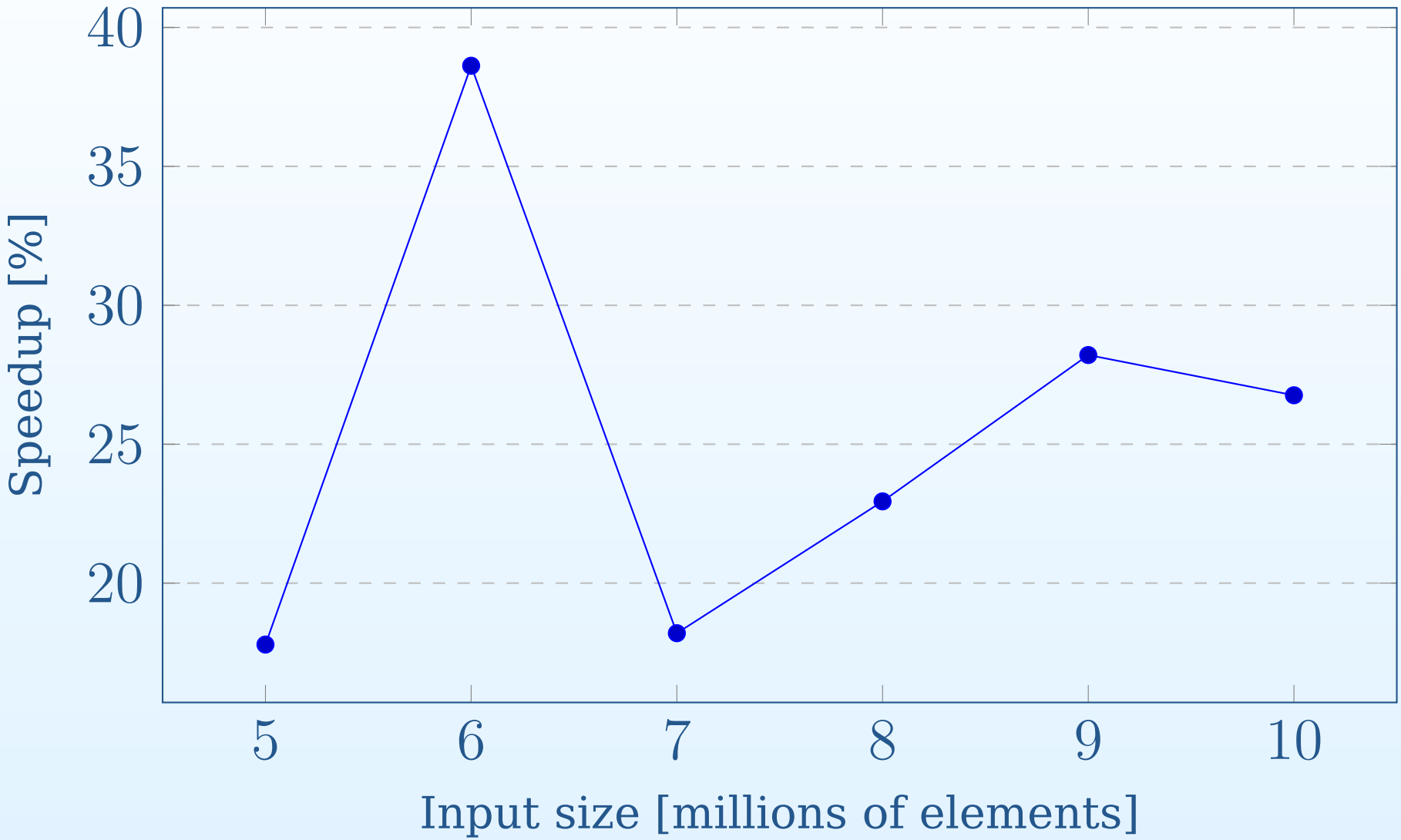
- Sentinel technique old as dust
- Yet not applicable directly to generic code
- Must use proper introspection techniques
- `is` and `static if` widely applicable
  - Mold *structure* depending on capabilities
  - stdlib: `static if` every 88 lines as wc counts

- Does the optimization help?

Find with Sentinel



Relative Speedup



Sentinels

- Widely applicable technique
- Let’s use them for something more complex
- How about Tony Hoare’s partition?
  - Workhorse of sorting and the selection problem

Baseline (1/2)

```
size_t pivotPartition(Range)(Range r, size_t pivot) {
  r.swapAt(pivot, 0);
  size_t lo = 1, hi = r.length - 1;
  loop: for (;; lo++, hi--) {
    for (;; ++lo) {
      if (lo > hi) break loop;
      if (r[lo] >= r[0]) break;
    }
    // found the left bound:  r[lo] >= r[0]
```

Baseline (2/2)

```
    // found the left bound:  r[lo] >= r[0]
    for (;; --hi) {
      if (lo >= hi) break loop;
      if (r[0] >= r[hi]) break;
    }
    // found the right bound: r[hi] <= r[0]
    // swap them & make progress
    r.swapAt(lo, hi);
  }
  --lo;
  r.swapAt(lo, 0);
  return lo;
}
```

Make It Faster

Accelerating `pivotPartition`

- Currently: find from left, find from right, swap, make progress
- Done when indexes meet
- Many index checks interspersed with actual work
- Can’t plant a sentinel in the “middle”—final pivot position unknown

Key Idea

- Plant sentinels at both ends
- Create a “vacancy” in the range
- Break `swapAt` into two assignments
- An assignment fills a vacancy and leaves another one

  Half swap!
- Work becomes matter of moving the vacancy around
- At the end fill back the vacancy

Implementation (1/3): Setup

```
size_t pivotPartition(Range)(Range r, size_t pivot) {
  if (r.length <= 1) return 0;
  // Put pivot at the front
  r.swapAt(pivot, 0);
  auto p = r[0];
  // Plant the pivot in the end as well as a sentinel
  auto save = r[$ - 1];
  r[$ - 1] = p; // r[$ - 1] is now vacant
```

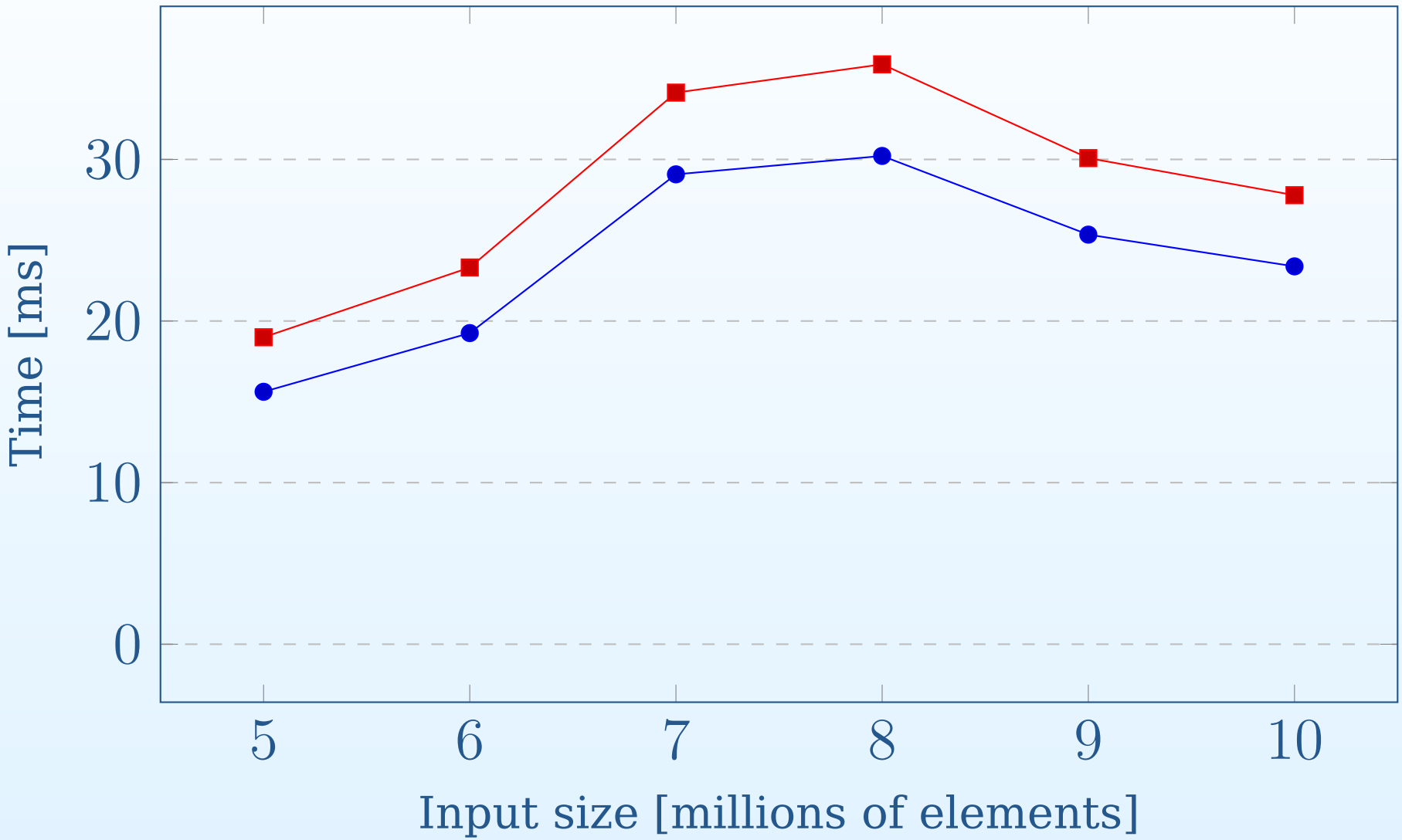
Implementation (2/3): Core

```
size_t lo = 0, hi = r.length - 1;
for (;;) {
  do ++lo; while (r[lo] < p);
  r[hi] = r[lo];
  do --hi; while (p < r[hi]);
  if (lo >= hi) break;
  r[lo] = r[hi];
}
```

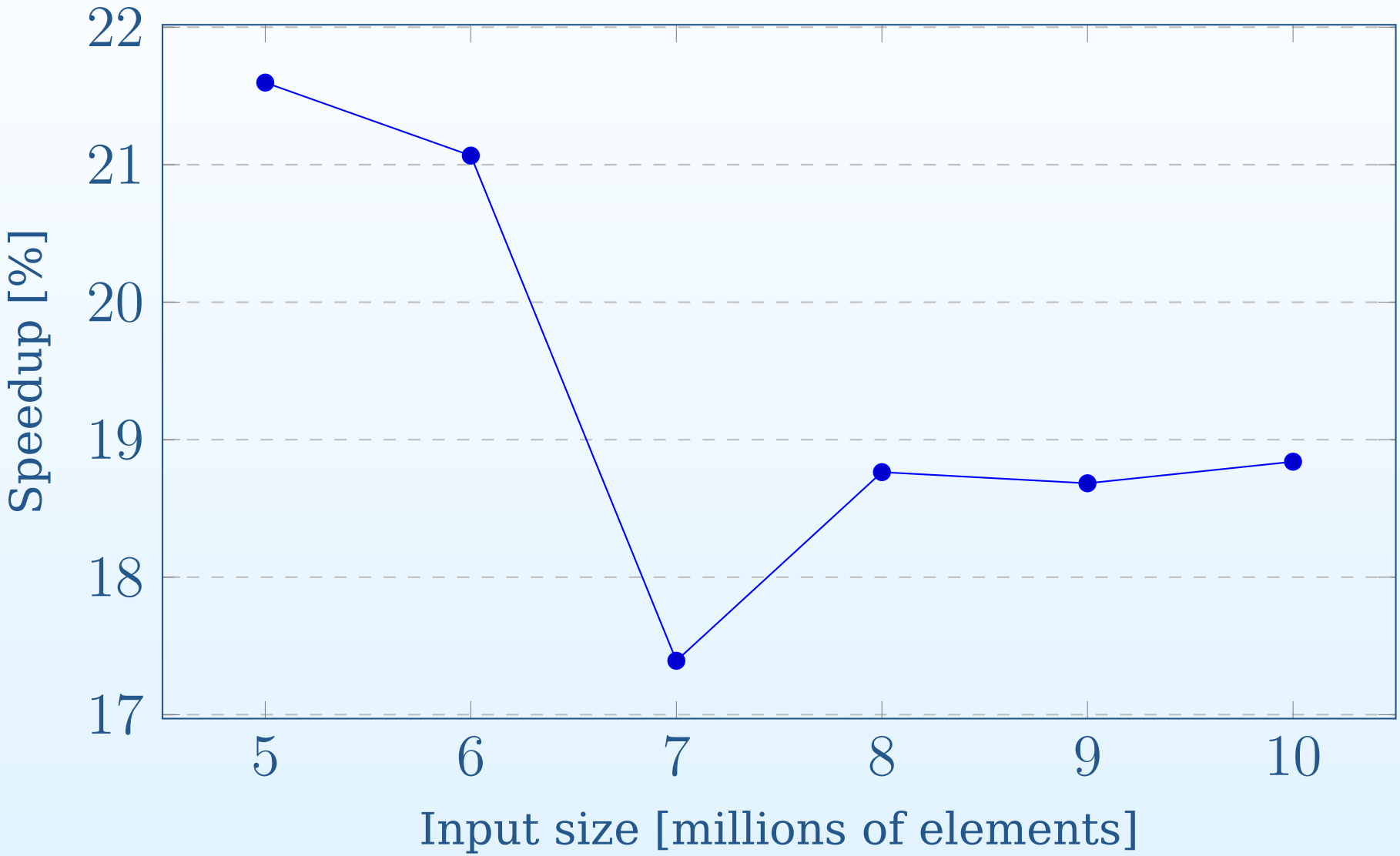
Implementation (3/3): Fixup

```
    assert(lo - hi <= 2);
    assert(p >= r[hi]);
    if (lo == hi + 2) {
      assert(r[hi + 1] >= p);
      r[lo] = r[hi + 1];
      --lo;
    }
    r[lo] = save;
    if (p < save) --lo;
    assert(p >= r[lo]);
    r.swapAt(0, lo);
    return lo;
  }
```

Partition



Partition Speedup



Credit

- Thought I invented this!
- Nope: “Engineering of a Quicksort Partitioning Algorithm”, Abhyankar & Ingle, Journal of Global Research in Computer Science, Feb 2011
  - “Vacancy” idea, cannot choose any pivot

More Sentinels

- These sentinel-based optimizations are not cherry picked!
- Dot product
- Set intersection
- Merge sorted lists
- Lexing and parsing

There's Treasure Everywhere

- You just have to find it

There's Treasure Everywhere

- You just have to find it
- (by using introspection)

Fastware

- A cornucopia of optimization techniques

- Benchmarking Techniques
- Strength Reduction
- Cache Friendliness
- Indirect Write Elision
- Memoization and Obliviation
- Hoisting and Lowering
- . . . and many more!

Destroy!