Comparing Our Heaps: Min vs Max

Assignment 2

Min-Heap: Ruslan Dussenbayev
Max-Heap: Asset Iglikov
Group: SE-2434
Course: Design & Analysis of Algorithms

What we built

I built a Max-Heap (biggest element on top), my partner built a Min-Heap (smallest on top). Basically the same thing, just flipped.

My Max-Heap:

- Keeps track of the maximum
- Has increase-key operation
- Works with integers only
- Lots of tests (30+)

Partner's Min-Heap:

- Keeps track of the minimum
- Has decrease-key operation
- Works with any comparable type (generic)
- Has a merge function to combine two heaps

Performance comparison

We tested both with 100,000 random numbers. Here's what happened:

What we measured	Min-Heap	Max-Heap	Difference
Insert time	1,180 ms	1,150 ms	Basically the same
Extract time	2,680 ms	2,720 ms	Also the same
Key update time	420 ms	410 ms	Still the same
Number of comparisons	~782K	~779K	Within 1%
Number of swaps	~391K	~390K	Within 1%

Bottom line: They perform identically. The 2-3% differences are just noise from how we ran the tests.

Complexity - they're the same

Operation	Time complexity	Who's faster?
Insert	O(log n)	Tie
Extract	O(log n)	Tie
Update key	O(log n)	Tie
Peek	O(1)	Tie
Space used	O(n)	Tie

Only real difference: partner has a merge operation (O(n) to combine two heaps). I didn't implement that.

What's different in the code

Partner's approach:

- Used generics so it works with any type
- Always tracks metrics (comparisons, swaps, etc.)
- Has that merge feature
- Growth strategy: array grows by 1.5x

My approach:

- Integer only (simpler but less flexible)
- Metrics are optional (faster when you don't need them)
- No merge operation
- Same 1.5x growth

Similarities:

- Both use arrays (not pointers)
- Both use bit shifts for speed (>> 1 instead of /2)
- Both handle null values safely
- Both have proper error messages

What could be better

For partner's Min-Heap:

1. Metrics slow everything down by 15-20% even when you don't need them. Should make them optional.
2. The indexOf() function is slow (O(n)) which makes decrease-key slow when you don't know the index.

3. Array never shrinks, wastes memory if you delete a lot.

For my Max-Heap:

1. Should support generics like partner's does.
2. Missing the merge operation - that's actually useful.
3. Also doesn't shrink the array.

What we both need:

- Array shrinking when size drops
- Maybe different growth rates depending on use case

When to use which

Use Min-Heap if you need:

- Smallest element repeatedly (like Dijkstra's algorithm)
- To merge multiple heaps
- Generic type support
- Decrease-key operation

Use Max-Heap if you need:

- Largest element repeatedly (like finding top scores)
- Fastest possible speed (no metrics overhead)
- Simpler code
- Increase-key operation

Honestly though, pick based on whether you need min or max. Performance is identical.

What we learned

Things that surprised us:

- The 15-20% overhead from always tracking metrics. Seems small but adds up fast.
- How close our performance numbers were despite different coding styles.
- That the theoretical O(log n) actually holds up in practice.

Partner learned:

- How to use generics properly
- That metrics tracking has a real cost
- How merge operations work

I learned:

- Value of extensive testing (30+ tests caught bugs)
- How to make optional features (metrics)
- That simpler isn't always better (partner's generics are nice)

Working together

What went well:

- We both finished on time
- Code review was helpful for both of us
- Our benchmarks were easy to compare

What was challenging:

- Different design choices (generics vs simple)
- Coordinating benchmark formats
- Finding time to meet and discuss

Next time:

- Agree on interfaces earlier
- More frequent check-ins
- Shared test cases from the start

Final thoughts

Both heaps work great. The choice between them is just about what you need (min vs max), not about speed. We both got similar grades (A-) and both implementations are good enough for real use.

Main takeaway: Sometimes there's no "better" algorithm - just different tools for different jobs.

We both approve this summary ;)