

Generic Programming: Miniproject - exam

Binomial Heap with guaranteed strong exception safety and optimal running time

Søren Christiansen

01.09.08

Agenda

Motivation
Exception-Safety
Running Time
Testing
Conclusion

Agenda

Motivation

Exception-Safety

Running Time

Testing

Conclusion

Motivation

- ▶ A review of the former no-exception safe solution, discovered:
 - ▶ No compliance with C++/STL container requirements
 - ▶ No consistent code comments/Code difficult to maintain
 - ▶ Tests files couldnt compile
 - ▶ *Lacked **Strong** exception-safety*
- ▶ Bue V. rewrote the binomial-heap, and ...
 - ▶ introduces **Strong** exception safety, but..
 - ▶ also an increase in running time costs of $O(N)$, for functions with rollback semantic.
- ▶ Ambition: Experimentation with introducing strong exception-safety, without increasing running time (significantly)
- ▶ Ambition: Exhaustive exception-safety tests.

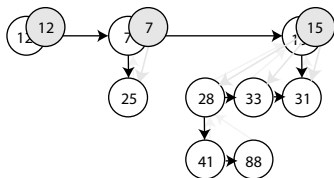
Exception-Safety

- ▶ 3 kinds of guarantees:
 - ▶ Basic: no leaks
 - ▶ Strong: no change in (component/class) state
 - ▶ No-Throw: Can not throw
- ▶ Basic: Copy Construction and swapping (Sutter)
- ▶ Copy-Construct: RAll and base-class encapsulation
- ▶ Swap: No throw()

```
SafeBinomialHeap tmp(*this);  
... unsafe operations  
Swap(tmp);
```

Running Time

- ▶ Strong exception safety $\rightarrow O(N)$ upperbound running time (copy).
- ▶ Copy Constructing: Shallow copy of heap $\rightarrow O(\lg N)$



- ▶ Parent pointers cause troubles: `fix_parent_pointer()`
- ▶ .. or remove parent pointers!
- ▶ `pop()` : mess with children of removed node. easy fix.

Testing

- ▶ A.Abrahams: Generic strong exception-safety test
- ▶ Client code/type that throws everywhere (ThrowType)

```
ThrowType ( T v = T() )  
: p ( CanThrow( v ) )  
{  
}
```

- ▶ Exhaustive, found problem in pop()

Conclusion

- ▶ Found a $O(\lg N)$ strong exception-safe solution
- ▶ Found a way to test for strong exception-safety
- ▶ Found some minor problems. `pop()`, parent pointers
- ▶ Need to integrate this into the existing solution
 - ▶ Added use of allocators
 - ▶ Added a more elegant `past_the_end` solution