CS 310
Assignment 113
February 18, 2022

Jingbo Wang

The `pq.h` algorithm given in the assignment could Provide a priority queue based on an unsorted vector and the value in order. This is accomplished by `PriorityQueue` class.

The input size of the `pq.h` algorithm is n, defined on line 18 of the program.

This algorithm has distinct best and worst cases.There are both best and worst cases run:

In function `void push(unsigned priority)`:

- Line 30: a `push_back` function call, 1 operation for each times, n operations.
- Line 33: a `bubble_up` function call, 1 operation for each times, n operations.

In function `unsigned pop()`:

- Line 43: an element assignment for `max_value`, 2 operations for each times, $2n$ operations.
- Line 46: an element assignment for `array.at(0)`, 3 operations for each time, $3n$ operations.
- Line 49: a `pop_back` function call, 1 operation for each times, n operations.
- Line 52: a `percolate_down` function call, 1 operation for each times, n operations.

In function `bool is_finish(size_t position)` is used in function `void percolate_down(size_t position)`:

- 161: an element assignment for `done`, 12 operations.

In function `void bubble_up(size_t position)` is Resurrection function, so we have:

- Line 95: a `if` statement header call, 1 operation.
- Line 97: a `if` statement header call, 4 operations.
- Line 99: a `swap` function call, 2 operations.
- Line 104: an element assignment for `parent`, 3 operations.

For the function `void bubble_up(size_t position)` we have best case:

When the Line 95: `if` is false, the total operations: 1 operation.

$$
\begin{aligned}
T(n) &\geq 1 \\
&\in \Omega(1)
\end{aligned}
$$

For the function `void bubble_up(size_t position)` we also have worse case:

All `if` statement are always true:

number of recursive calls(a): 1

size of each recursive call($\frac{n}{b}$): $n/2$

Total operations(k): $1 + 4 + 2 + 3 = 10$

d: 0

So, we have:

$$T(n) \geq 1 \times T(\frac{n}{2}) + 10n^0$$

Because of $1 = 2^0$:

$$\begin{aligned} T(n) &\leq 1 \times T(\frac{n}{2}) + 10n^0 \\ &\in O(\lg n) \end{aligned}$$

We run n times for function `void bubble_up(size_t position)`, we have:

$$\begin{aligned} T(n) &\geq 1 \\ &\in \Omega(n) \\ T(n) &\leq 10 \lg n \\ &\in O(\lg n) \end{aligned}$$

In function `void percolate_down(size_t parent)`:

- Line 118: a `if` statement header call, and includes a `is_finish` function call, 17 operations.
- Line 120–122: an element assignment for `position`, 8 operations.
- Line 124: a `if` statement header call, 3 operations.
- Line 126: a `swap` function call, 2 operations.
- Line 128: an element assignment for `parent`, 3 operations.
- Line 133: a `swap` function call, 2 operations.
- Line 135: an element assignment for `parent`, 3 operations.
- Line 142: a `if` statement header call, 8 operations.
- Line 145: a `swap` function call, 2 operations.

For the function `void percolate_down(size_t parent)` we have best case:

When the Line 119: `if` is false, the total operations: 5 operation.

$$\begin{aligned} T(n) &\geq 5 \\ &\in \Omega(1) \end{aligned}$$

For the function `void percolate_down(size_t position)` we also have worse case:

All `if` statement are always true:

number of recursive calls(a): 1

size of each recursive call($\frac{n}{b}$): $n/2$

Total operations(k): $17 + 5 + 8 + 3 + 2 + 3 + 2 + 3 + 8 + 2 = 53$

d: 0

So, we have:

$$T(n) \leq 1 \times T(\frac{n}{2}) + 53n^0$$

Because of $1 = 2^0$:

$$\begin{aligned} T(n) &\leq 1 \times T(\frac{n}{2}) + 11n^0 \\ &\in O(\lg n) \end{aligned}$$

We run n times for function `void percolate_down(size_t position)`, we have:

$$\begin{aligned} T(n) &\geq 5 \\ &\in \Omega(n) \\ T(n) &\leq 53 \lg n \\ &\in O(\lg n) \end{aligned}$$

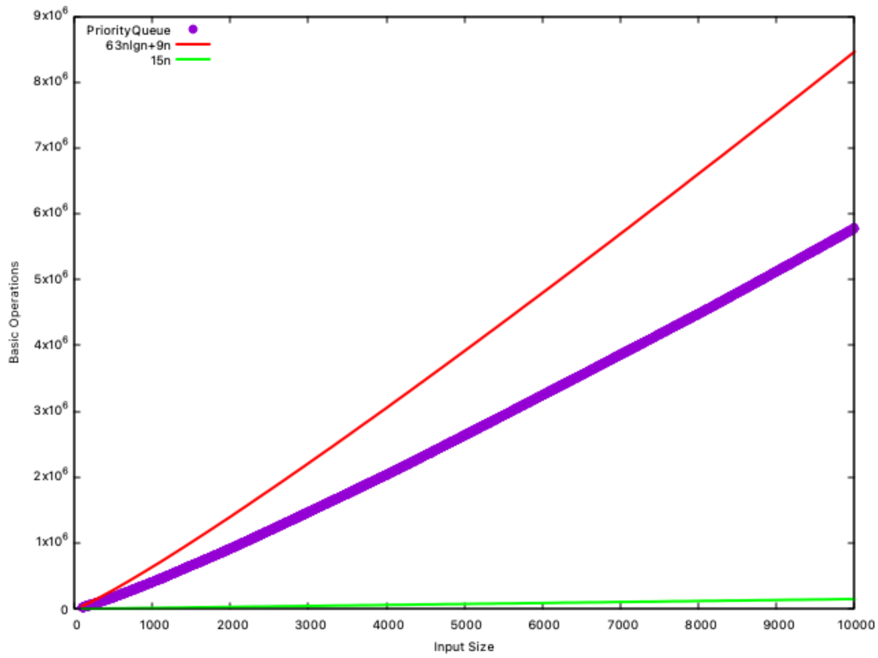
For the `PriorityQueue` class call both `push` and `pop` in the `analyze_pq.cpp` running n times, we have:

$$\begin{aligned} T(n) &\geq n + n + 2n + 3n + n + n + n + 5n \\ &\geq 15n \\ &\in \Omega(n) \\ T(n) &\leq n + n + 2n + 3n + n + n + 10n \lg n + 53n \lg n \\ &\leq 63n \lg n + 9n \\ &\in O(n \lg n) \end{aligned}$$

The program was run with the command:

```
for n in $(seq 100 10 10000)
do
    ./program $n
    ./program $n
    ./program $n
done 2> result.dat
```

in order to generate a set of points. The resulting data were plotted, giving the following. Also plotted on the same axes are the scaled standard functions $63n \lg n + 9n$ and $15n$ which illustrate that $63n \lg n + 9n$ that above the algorithm is worst case, and $15n$ below the algorithm is the best case.



We see that the plot confirms the theoretical analysis above.

It is same like we did in class.