CS 61B Spring 2017 — Final Discussion

Discussion 14: April 26, 2017

1 Sixty-one

1.1 You have been hired by Alan to help design a priority queue implementation for *Kelp*, the new seafood review startup, ordered on the timestamp of each `Review`.

Describe a data structure implementation that supports the following operations.

- `insert(Review r)` a `Review` in $O(\log N)$.
- `edit(int id, String body)` any one `Review` in $\Theta(1)$.
- `sixtyOne()`: return the sixty-first latest `Review` in $\Theta(1)$.
- `pollSixtyOne()`: remove and return the sixty-first latest `Review` in $O(\log N)$.

Maintain a max-heap called `firstSixtyOne` with 61 `Reviews`, a min-heap called `olderReviews` with all the rest, and a `HashMap` mapping any given integer `id` to its corresponding `Review`.

1.2 Give the *amortized runtime analysis* for push and pop for the priority queue below.

```java
class TwinListPriorityQueue<E implements Comparable> {
    ArrayList<E> L1, L2;
    void push(E item) {
        L1.push(elem);
        if (L1.size() >= Math.log(L2.size())) {
            L2.addAll(L1);
            mergeSort(L2);
            L1.clear();
        }
    }
    E pop() {
        E min1 = getMin(L1);
        E min2 = L2.poll();
        if (min1.compareTo(min2) < 0) {
            L1.remove(min1);
            return min1;
        } else {
            L2.remove(min2);
            return min2;
        }
    }
}
```

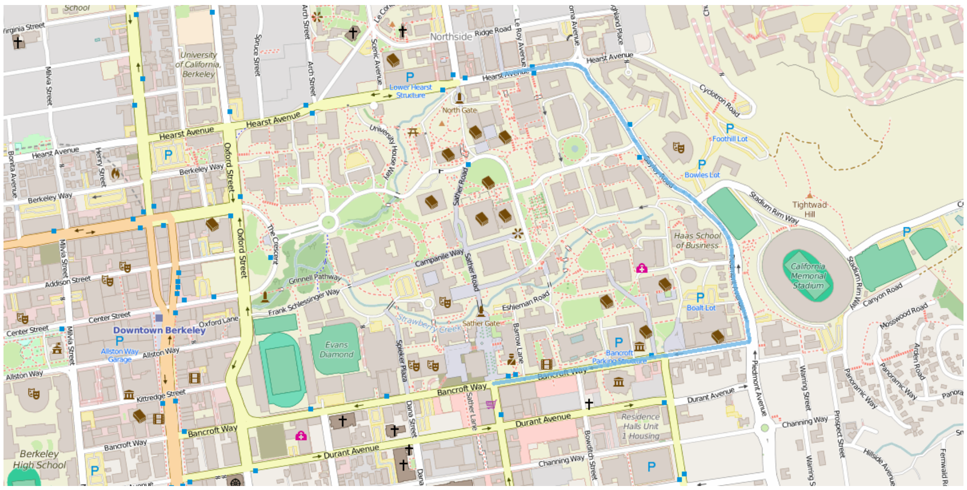
Let N be the number of elements in the priority queue. Then the amortized runtime for push is in $O(N)$ as the cost for every $\log N$ insertions is in $O(\log N \cdot 1 + 1 \cdot N \log N)$ which simplifies to $O(N)$. Note that the size of `L1` is always constrained to be in $O(\log N)$.

The amortized runtime for `pop` is also in $O(N)$. `getMin` on the unsorted list, `L1`, is in $O(\log N)$, as with `L1.remove(min1)`. Polling from the front of `L2` is in $\Theta(1)$. The most expensive component is `L2.remove(min2)` which is in $O(N)$.

2 From humble beginnings

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("hello world");
    }
}
```

3 To where we are now



4 And what lies ahead

61C Machine abstractions and optimizing programs in the real world. My favorite lower-division course: it'll change the way you think about data structures.

70 Mathematical rigor and learning more than you knew you could ever learn.

152 Like CS 61C, but turned up to 11. More analysis and open-ended projects.

161 Security: from social engineering & cost-benefit analysis to theory & crypto.

168 Network systems and Internet architecture. Scott Shenker cares so much about you that you won't even mind that the content is very dry and uninteresting.

170 Graph algorithms and techniques for problem solving like divide-and-conquer.

184 Databases like project 2, but with more of an interest in performance.

188 Artificial intelligence: assignments are fun, but *don't underestimate the exams*!

194-26 Computational photography, or the best class you'll take your senior year.

195 Ethics course. Spend another semester having fun with John, Josh, and Dan!

197 Academic interning for CS 61B.

198 Computer Science Mentors for CS 61B.

370 Intro to Teaching Computer Science with Chris Hunn, the best person ever.