Weekplan: Predecessors

Philip Bille

References and Reading

[1] Scribe notes from MIT


Exercises

1 The Google Egg Interview Problem You are given 2 identical eggs and a 100-floor building. You want to compute the highest floor from which an egg (identical to yours) can be dropped without breaking. Solve the following exercises.

1.1 How few drops can you do it with? You are allowed to break the 2 eggs in the process.
1.2 Give a bound on the number of drops for a building with $x$ floors.
1.3 [*] Show how to achieve a good bound (maybe roughly the same as in 2?) even when you do not know the number of floors in advance.

2 Range Reporting Give a data structure for a set $S \subseteq U = \{0, \ldots , u-1\}$ of $n$ values that supports the following operation:

- report$(x, y)$: return all values in $S$ between $x$ and $y$, that is, the set of values $\{z \mid z \in S, x \leq z \leq y\}$.

The goal is a data structure with fast output-sensitive query bounds, that is, the query time should be on the form $O(f(n, u) + \text{occ})$, where occ is the number of elements returned by the query and $f(n, u)$ is a fast as possible.

3 van Emde Boas Bounds Show that $T(u) = T(\sqrt{u}) + O(1) = O(\log \log u)$. Hint: consider the binary representation of $u$.

4 X-fast Tries Let $S = \{8, 9, 13, 16, 20, 26\}$ be a set $S$ from a universe $U = \{0, \ldots , 31\}$. Solve the following exercises.

4.1 [$w$] Draw the x-fast trie of $S$, including the trie structure and the contents of the dictionary.
4.2 [$w$] Show each step of predecessor searches for 8, 14, 21, and 30.

5 Z-Fast Tries An fellow student suggest a modification of the y-fast trie which he proudly names the z-fast trie. The z-fast trie partitions $S$ into groups of $\log^6 u$ consecutive values (recall y-fast tries partitions into groups of $\log u$ values). How does z-fast tries compare to y-fast tries?
6 Dynamic Y-Fast Tries  Solve the following exercises.

6.1 Show how to add insert and delete operation to the presented static solution for y-fast tries. Predecessor queries should take $O(\log \log u)$ expected time and updates should take $O(\log \log u)$ amortized expected time, i.e., any sequence of $k$ updates should take $O(k \log \log u)$ expected time. The space should be $O(n)$.

6.2 A friend of yours is not happy with y-fast tries and want to make x-fast tries dynamic instead. He claims that he can maintain the x-fast trie data structure in the same time bounds as above. Prove or disprove his claim.

7 Shortest Paths  Let $G$ be a graph with $n$ vertices and $m \geq n$ weighted edges. The edge weights are from the set $U = \{0, \ldots, u - 1\}$ and $u > m$. Show how to compute the shortest path between two vertices in $O(m \log \log u)$ expected time.

8 The Bomberman Problem  Let $A$ be a $2D$ array of size $u \times u$. We consider efficient data structures for placing and exploding bombs within $A$. Let $b_{i,j}$ and $b'_{i',j'}$ be two bombs at positions $(i, j)$ and $(i', j')$ in $A$ and let $t$ be an integer, $1 \leq t \leq u$. We define the bombs to be connected with threshold $t$ if one of the following holds:

- $i = i'$ and $|j - j'| \leq t$,
- $j = j'$ and $|i - i'| \leq t$, or
- if there is a bomb $b''_{i'',j''}$ such that both $b$ and $b'$ are connected with threshold $t$ to $b''$.

We want to support the following operations on $A$:

- $\text{place}(i, j)$: Place a bomb at position $(i, j)$ in $A$.
- $\text{explode}(b_{i,j}, t)$: Remove all bombs connected with threshold $t$ to $b_{i,j}$.

Given a data structure that supports the above operations efficiently. The complexity for explode should depend on the number $k$ of bombs removed.

9 List Jumping  Let $L$ be a list of $n$ sorted integers in increasing order from the range $U = \{0, \ldots, u - 1\}$. We are interested in supporting successor queries on $L$ when already have a pointer to some element within $L$. The time for successor should depend on the distance between the query and element we have a pointer to. Specifically, we want to support the following operation on $L$. Let $e$ be an element of $L$ and let $x$ be an integer from $U$ such that value of element $e$ is smaller than $x$.

- $\text{succ}(x, e)$: Return the successor of $x$

Solve the following exercises. Define $d(x, e)$ to be the number of elements between $e$ and $x$, i.e., the number of elements in $L$ after $e$ that are smaller than $x$. Define $D(x, e)$ to be the difference between the value of $e$ and $x$.

9.1 Show how to augment $L$ with additional pointers to support $\text{succ}(x, e)$ in $O(n \log n)$ space and $O(\log d(x, e))$ time.

9.2 [*] Improve the above bound. Give a data structure that supports $\text{succ}(x, e)$ in $O(n)$ space and $O(\log d(x, e))$ time. $\text{Hint}$: start by building a complete binary tree on top of $L$. Connect nodes on the same level.

9.3 [**] Give a compact data structure that supports $\text{succ}(x, e)$ in $O(\log \log D(x, e))$ time. Assume you can support successor queries for sets of size $O(\log u)$ in constant time and linear space (such a data structure is called a fusion node or atomic heap). $\text{Hint}$: Combine the idea from exercise 2 with y-fast tries.