Litterature

At the lecture we will talk about data structures for partial sums and for dynamic arrays. See litterature on the course webpage.

Exercises

1 $[w]$ **Fenwick Trees** Consider an array $A = [-, 4, 3, 1, 5, 8, 4, 1, 3]$. Solve the following exercises.

1.1 Draw the Fenwick tree F for A.

1.2 Compute the indices in F that are accessed on a sum(5) query. Verify that the corresponding entries add up to the correct sum.

1.3 Compute the indices in F that are changed on an update$(5, 8)$ query. Update the entries and verify that the resulting array is a correct Fenwick tree.

2 Partial Sums With More Operations Consider extending the Fenwick tree to support the following new operations.

- access(i): return $A[i]$.
- report(i, j): return $A[i] + A[i+1] + \cdots + A[j]$.
- select(j): return the smallest i such that sum$(i) \geq j$.

Solve the following exercises.

2.1 Show how to implement access in $O(\log n)$ time.

2.2 Show how to implement report in $O(\log n)$ time.

2.3 Show how to implement select in $O(\log n)$ time. Assume entries in A are all positive.

3 Fenwick Tree Analysis and Implementation Solve the following exercises. Let F be the Fenwick tree of an array A of length n.

3.1 Discuss why the array F is called a tree.

3.2 Show that any interval $[1, i]$ is covered by $O(\log n)$ partial sums stored in F.

3.3 Consider the index computation for sum. Let i be a positive integer whose rightmost 1-bit is at position k, i.e., if $i = 10 = 1010_2$ then $k = 1$ and $2^k = 2^1 = 2$. Show that $j = i \,\&\, \bar{i} + 1$ is the integer 2^k. Here $\bar{i}$ the bitwise negation of i, i.e., if $i = 10$ then $\bar{i} = 0101_2$.

4 $[w]$ **Dynamic Arrays** Consider a 2-level rotated array data structure representing the array $A = [8, 11, 2, 3, 4, 9, 8, 1]$. Solve the following exercises.

4.1 Draw the 2-level rotated array. Use $\sqrt{n} \approx 3$.

4.2 Show how to compute the rotated array R_j and index k in R_j corresponding to an index i in A. Both should be constant time.

4.3 Show the result of each of the operations insert$(5, 42)$, delete(2), delete(6).

5 Dynamic Arrays on the Fringe Consider the dynamic array problem with the restriction that insertion and deletions always occurs within the leftmost or rightmost 42 positions of the array. Give a simple solution to the problem with this restriction.

6 Dynamic Arrays with Fast Updates Suppose we insist on a data structure for dynamic arrays that support updates in constant time. What is the fastest time for access you can achieve? For simplicity, ignore deletions and only consider access and insert.

7 In-Place Dictionaries Let S be a set of n integers. Recall that a *dictionary* supports the following operations.

- member(x): determine if $x \in S$.
- insert(x): set $S = S \cup \{x\}$.
- delete(x): set $S = S \backslash \{x\}$.

We consider in-place dictionaries. Solve the following exercises.

7.1 [w] Suppose that we only want to support member. Give a in-place data structure that supports member in $O(\log n)$ time. *Hint:* binary search.

7.2 Let A be a rotated sorted array of length n containing the integers in S, i.e., A is a circular shift of the sorted array of the integers in S. Show how to search in A in $O(\log n)$ time. Your solution should be in-place. Furthermore, assume that you cannot afford to store the circular shift in the data structure. *Hint:* peaks.

7.3 [$*$] Show how implement an in-place dictionary that supports member in $O(\log n)$ time and insertions and deletions in $O(\sqrt{n} \log n)$ time.