1 Divide and Conquer

- Divide and Conquer Topics:
  - Median Finding
  - Convex Hull
  - FFT
  - vEB trees

- Paradigm: a problem of size $n$ is split into $a$ problems of size $n/b$, after solving each of these we then merge the answers to solve the original problem. This allows us to write the recurrence $T(n) = aT(n/b) + cost\_split + cost\_merge$. Recursion is relatively easy to code and also allows for easy analysis! Oh man.

- Master Theorem. If the recursion follows $T(n) = aT(n/b) + f(n)$ then
  - if $f(n) = \Theta(n^{\log_b(a)} \log^k(n))$ then $T(n) = n^{\log_b(a)} \log^{k+1}(n)$
  - elseif $f(n) = o(n^{\log_b(a)})$ then $T(n) = n^{\log_b(a)}$
  - elseif $f(n) = \omega(n^{\log_b(a)})$ then $T(n) = f(n)$

2 vEB Trees

- Recursive tree data structure.

- For universe size $u$, achieves $O(\log \log u)$ time for Insert, Delete, Search, Successor, Predecessor. (A common mistake is confusing the $u$ for $n$. $n$ is typically the number of elements currently stored in the vEB tree.)

- vEB tree $V$ contains:
  - Universe size $V.u$
  - $V.min$ which is the minimum element in $V$ and is not recursively stored
  - Summary vEB $V.summary$
  - A list $V.cluster$ where $|V.cluster| = \sqrt{u}$ and each $V.cluster[i]$ is a vEB of universe size $\sqrt{u}$

- Has recurrence $T(u) = T(\sqrt{u}) + O(1) = O(\log \log u)$
3 Fast Fourier Transform (FFT)

- Achieves $O(n \log n)$ to multiply two $O(n)$ degree polynomials $A(x) \cdot B(x) = C(x)$.

- The idea is as follows:
  - Evaluate $A(x)$ at $2n + 1$ different $x$’s (FFT)
  - Evaluate $B(x)$ at the same $2n + 1$ $x$’s (FFT)
  - Multiply these $2n + 1$ evaluations to get the evaluations for $C(x)$
  - Finally interpolate these $2n+1$ evaluations to obtain $C(x)$ (Inverse FFT)

- FFT evaluates the polynomial $A(x)$ using Divide and Conquer as follows:

  $A(x) = A_{\text{even}}(x^2) + x \cdot A_{\text{odd}}(x^2)$

  $A_{\text{even}}(x) = \sum_{k=0}^{\lfloor \frac{n}{2} - 1 \rfloor} a_{2k} \cdot x^k$

  $A_{\text{odd}}(x) = \sum_{k=0}^{\lfloor \frac{n}{2} - 1 \rfloor} a_{2k+1} \cdot x^k$

- FFT uses the $2n+1$-th roots of unity as it’s $x$’s and therefore computes the DFT $A*$ of the coefficient list $A$ which is:

  $a_k^* = \sum_{j=0}^{n-1} a_j \cdot e^{\frac{2\pi i k}{2n+1}}$ where $0 \leq k < 2n + 1$

- Because squaring the $j$-th roots of unity yields the $\frac{j}{2}$-th roots of unity (hence halves the size of the set of evaluations) FFT has the following familiar recurrence:

  $T(n) = 2T(\frac{n}{2}) + O(n) = O(n \log n)$
4 B - Trees

- Like binary trees, but each node may contain multiple keys/children.
- Achieves $O(\log_B n) = O(\lg n)$ runtime for Insert, Delete and Search.
- Each node stores keys in sorted order.
- Number of keys for any non root node, $\# keys$, is allowed to be $B - 1 \leq \# keys < 2B - 1$
- Root node is allowed to have anywhere between 1 and $2B - 2$ keys.
- Number of children is always $1 + \# keys$ (except for leaves which have 0 children). Consequently:
- Number of children for any internal node (non-root), $\# children$, is allowed to be $B \leq \# children < 2B$
- Number of children for root can be anywhere between 2 and $2B - 1$.

5 Amortization

Please refer to Recitation 3 notes for more detailed explanations of Union-Find. We have expanded the analysis for the “First Improvement, Smaller into Larger” section to be more helpful with amortization and uploaded a new PDF on stellar.

6 Augmentation (Range-Trees)

Not Covered in Review Session due to time constraint.

The following notes cover Randomized Algorithms (Analysis), Hashing, and Skip-Lists as discussed in the review session.
Randomization

- Method of analysis
- Doesn't guarantee great outcome but in expectation (or on "average"), it can show that an alg. performs well. Alternatively, whp shows that it performs in a range that is acceptable most of the time. However worst case still possible, but w/low prob.
- Relevant lectures/recs.
  - Lectures 6, 7, 8
  - Recitation 4

- Topics
  - Quicksort (Randomized)
  - Skip Lists
  - Matrix Multiplication (Verification)
  - Hashing

- Quick Review

  - Randomized Quicksort
    Pick a random pivot. Partition elts based on pivot. Recurse
    Deterministic worst case: $O(n^2)$, if we pick pivots in non-increasing order
    Randomized: $O(n \log n)$ (see recitation)
    Whp: $O(n \log n)$ (see ps4)

T/F An adversary can construct an input of length n to force RANDOMIZED-QUICKSORT to always run in $\Omega(n^2)$ time
False, expected runtime is $\Theta(n \log n)$. Can run in $\Theta(n^2)$ but not always
Skip Lists: Have \( \lg n \) sorted lists w/ successively more items in lower layers.

Search: \( O(\lg n) \) w/ hp

Move through level until next item is larger than you, use pointer to go down a level.

Complete analysis requires useful analysis: see notes for full details.

TF: Worst case running time for search in Skip Lists: \( \Omega(n) \).

Hashing:

Universal hashing:

Choose a random function \( h \) from family of functions \( \mathcal{H} \)

\( \mathcal{H} \) is universal if \( \forall k, k' \) where \( k \neq k' \)

\[ \Pr_{h \in \mathcal{H}} \left[ h(k) = h(k') \right] \leq \frac{1}{m} \]

where \( h \) maps to a set of size \( m \).

Favorite universal hash functions:

Dot product hash:

For a key \( k = \langle k_0, k_1, k_2, \ldots, k_r \rangle \)

For a hash function \( h \) pick a random vector \( a + \text{prime} \)

\[ a = \langle a_0, a_1, \ldots, a_r \rangle \]

\( m \) is a prime

\[ h_a(k) = a \cdot k \mod m \]

Another:

Pick prime \( p > m^r \), then \( a, b < p \)

\[ h_{ab}(k) = \left[ (a \cdot k + b) \mod p \right] \mod m \]

Perfect Hashing:

\( O(1) \) search - expected takes \( O(n^2 \lg n) \) to build table w/ hp

\( O(n) \) space

TF: When achieving perfect hashing w/ \( \Theta(n) \) space, both the 1st & 2nd levels must avoid collision. FALSE, only 2nd level.
More about skip lists

To insert, search for the item. Put it in bottom list. Flip a coin. Every time it comes up heads, put in the list a level higher. Analysis in class showed that w/hp it wouldn't rise too many levels, giving a limit to traversal depth. Also gives hp bounds on sideways traversal (i.e., has good # elts @ each level).

Frievald's Alg.
Choose a random binary vector $r_i^{[n]}$ such that $\Pr[r_i = 1] = 1/2$ independently.
If $A(B_r) = C_r$, then output YES.
0/1 NO

Correctness: If $AB = C - 100\%$.
$AB \neq C \geq 50\% \cdot 1/2$
Consider family of such that if a 3 maps

then H is a universal base family, and

So 31 ">= 302.

Also note, expected probability of small pim.

Always correct if small pim.