Deterministic Methods

1. Incremental Method: Add the objects in the given set, one at a time, in some *predetermined* order.

2. Divide and Conquer: Divide the object set in some *predetermined* fashion and recur.

3. Search through a sequence of partitions computed in some *predetermined* fashion.

4. Scanning.

3 Randomized Methods

1. Randomized Incremental Method: Add the objects in the given set, one at a time, in a *random* order.

2. Divide and Conquer: Divide the object set in a *random* fashion and recur.

3. Search through a sequence of partitions computed in a *random* fashion.

Why/Why-Not Randomization

1. Generalizable to Higher Dimensions.
2. Simple Algorithms
3. Easy to Implement
4. Efficient Algorithms.
5. Need not always work!

Model of Randomness

- Algorithm makes random choices.
- No assumption on the Input.
- Expected Bounds refer to random choices in the input.
- Random choices are hidden from the outside world.
- All permutations are equally likely - i.e. each insertion order is equally probable.

Chapter 1: Quick-sort and search

1-Dimensional Geometry

Given a set N of n-points on the real line $\Re$.

Sorting Problem: Find the partition $H(N)$ of $\Re$ formed by the given set of points N. The partition is specified by the points in N, the resulting intervals, and the adjacencies among the points and the intervals.

Search Problem: Associate a search structure $\tilde{H}(N)$ with $H(N)$ so that, given any point $q \in \Re$, one can locate the interval in $H(N)$ containing q quickly, i.e., in logarithmic time.

Contents

1. Quick-Sort's Randomized Incremental version.
2. Search Structure: Randomized Binary Trees
3. Skip Lists
4. Dynamization.

The familiar Quicksort

randomized divide-and-conquer paradigm

Pick a random point $S \in N$.

It splits the real line $\Re$ into two halves.

Let $N_1, N_2 \subset N$ be the set of points contained in these two halves.

Sort N_1 and N_2, recursively.

Intuitively,

- *expected size* of N_1 and N_2, is $n/2$.
- *expected depth* of the recursion to be $O(\log n)$.
- *expected running time* to be $O(n \log n)$.

Pseudo-Random Numbers will suffice!

Randomized Incremental Version of Quick-sort

Key:: Unwind the recursion in the definition of quick-sort.

The required partition of the line is computed incrementally by adding the points in N, one at a time, in random order.

1. $N^i =$ set of first i added points.

2. $H(N^i) =$ partition of R formed by N^i.

3. Starting with $H(N^0)$, we compute a sequence $H(N^1)$, $H(N^2)$, ...

4. $H(N^n) = H(N)$.

5. Conflict List at the i-th stage for each interval $I \in H(N^i)$.
   $L(I) =$ Unordered points of $N \setminus N^i$ in I.

6. or, For each point in $N \setminus N^i$, pointer to its conflicting interval in $H(N^i)$.

7. Addition of a random chosen point $S = S_{i+1} \in N \setminus N^i$ consists of splitting the interval $I \in H(N^i)$ containing S, together with its conflicting list.

8. Let $I_1(S)$ and $I_2(S)$ be the intervals in $H(N^{i+1})$ adjacent to S. Let $l(I_1(S))$ and $l(I_2(S))$ be their conflict sizes.

Fact: The cost of adding S is proportional to $l(I_1(S)) + l(I_2(S))$, ignoring an additive $O(1)$ term.

What is the expected running time?

- Randomization in Algorithm only

-No assumption about the input.

-Points are added in a random order.

-All possible permutations are equally likely.

Useful Analysis Tricks

-What is the cost of $(i+1)$-st addition?

-*Go Backwards*: What is the cost of $i+1$-st deletion?

-Formally, given $H(N^{i+1})$ how do we obtain $H(N^i)$.

-For a fixed choice of N^{i+1} what is the cost?

-We aim to bound the *conditional expected cost* of the $i+1$-st addition, for every fixed choice of N^{i+1}, not N^i.

-It turns out that this will depend only on i and not on N^{i+1}.

-Hence, it will be unconditional!

Analysis

1. Fix N^{i+1} arbitrarily.

2. Due to randomness, each point in N^{i+1} is equally likely to occur as $S = S_{i+1}$.

3. Imagine, N^i results from N^{i+1} by deleting a random point S.

4. Expected cost of the (i+1)st addition, conditional on a fixed N^{i+1}, is
$\frac{1}{i+1} \sum_{S \in N^{i+1}} [l(I_1(S)) + l(I_2(S)) + 1]$
$\leq \frac{2}{i+1} \sum_{J \in H(N^{i+1})} [l(J) + 1]$
$= O(\frac{n}{i+1})$

5. Observe that this bound does not depend upon N^{i+1}, it unconditionally bounds the expected cost of $i+1$st addition.

6. Expected cost of the whole algorithm is $n \sum_i 1/(i+1) = O(n \log n)$.

Randomized Binary Tree - Search Structure

What is $\tilde{H}(N)$?

Just remember what we are doing - or maintain a history.

1. $\tilde{H}(N)$ is the history of computed partitions $H(N^0), \cdots, H(N^n)$.

2. Its not feasible to remember whole of $H(N^i)$, for all i's.

3. The difference between $H(N^i)$ and $H(N^{i+1})$ is small, so we just remember the difference between successive partitions.

4. **history(i)** be the history of first i-additions.

5. $\tilde{H}(N)$ =history(n).

6. history(0) consists of just one node, corresponding to $\Re$. Inductively, assume $history(i)$ has been defined. Let S_{i+1} be the randomly chosen point, belonging to interval I of $H(N^i)$. Interval I is killed and two new intervals I_1 and I_2 are created corresponding to the split of I.

7. history($i+1$) is obtained from history(i) by labeling the leaf of history(i), corresponding to interval I, with S_{i+1} and assigning two children, one each for I_1 and I_2.

Analysis of Randomized Binary Tree

Expected cost of computing $\tilde{H}(N)$ is $O(n \log n)$. The analysis is similar to the quick-sort analysis, so we skip.

What is the expected search/query cost?

1. Fix a query point $p \in \Re$.

2. *Backward Analysis*: Imagine tracing p backward in the history. Starting with $H(N) = H(N^n)$, imagine deleting points $S_n, \cdots, S_0$, one at a time, in the reverse order.

3. Define a 0-1 random variable V_j. $V_j = 1$ if and only if the interval in $H(N^j)$ containing p is adjacent to the deleted point S_j. (This interval is destroyed during the imaginary deletion o S_j)

4. Length of the search path in history(n)=$\sum_1^n V_j$.

5. Each point in N^j is equally likely to occur as S_j.

6. Probability($V_j = 1$)$\leq 2/j$; since at most two points are adjacent to the interval in $H(N^j)$ containing p.

7. Expected Value of $V_j = 2/j$.

8. By Linearity of Expectation, Expected Value of $V = O(\sum 1/j) = O(\log n)$.

9. Thus for a fixed query point p, the expected length of the search path is $O(\log n)$.