5. Leftist Heaps

Target: Speed up merging in $O(N)$

Order Property – the same

Structure Property – binary tree, but unbalanced

The null path length, $Np(X)$. $NPL(NULL) = -1$

the length of shortest path from X to a node without two children

Theorem:

A leftist tree with r nodes on the right path must have at least $2^r - 1$ nodes

$T_p = O(\log N)$ (Merge、delete Min)

the average time for insertions with leftist or skew heaps

Merge (recursive version)

Step1: Merge ($H_1 \to$Right, H_2)

Step 2: Attach (H_2, $H_1\to$Right)

Step3: Swap ($H_1\to$Right, $H_1\to$Left) if necessary.

(iterative version)

step1: Sort the right paths without changing their children.

Step2: Swap children if necessary

DeleteMin:

Step1: Delete the root

step2: Merge the two subtrees.

the total time for inserting N keys into an empty binary heap? $O(n)$

6. Skew heaps

Target: Any M consecutive operations take at most $O(M\log N)$ time

Always swap the left and right children No npl

except that the largest of all the nodes on the right paths does not have its children swapped

no extra space is required

no tests are required to determine when to swap children

A node p is heavy if the number of descendants of p's right subtree is at least half of the number of descendants of p.

D_i = the root of the resulting tree

$\Phi(D_i)$ = number of heavy nodes

$H_i: l_i + h_i \ (i=1,2) \Rightarrow T_{worst} = l_1 + h_1 + l_2 + h_2$

Before merge: $\Phi_0 = h_1 + h_2 + h$

After merge: $\Phi_N \le l_1 + l_2 + h$

$T_{amortized} = T_{worst} + \Phi_N - \Phi_0 \le 2(l_1 + l_2)$

$l = O(\log N)$

$T_{amortized} = O(\log N)$

7. Bionomal Queues

FindMin: $O(\log N)$ (we remember whenever it is changed then this operation will take $O(1)$)

Merge $O(\log N)$

Delete Min: step 1: FindMIN in B_k $O(\log N)$

step 2: Remove B_k from H

step 3: Remove root from B_k

step 4: Merge (H', H'')

B_k consists of a root with k children which are $B_0 B_1 \cdots B_{k-1}$, B_k has exactly 2^k nodes.

The number of nodes at depth d is $\binom{k}{d}$

$T_{worst} = O(\log N)$

$T_{amortized} = 2$

A binomial queue of N elements can be built by N successive insertions in $O(N)$ time.

The worst case time for each insertion is $O(\log N)$

Performing N inserts on an initially empty binomial queue will take $O(N)$ worst-case time. Hence the average time is constant.

AVL: $T_p = O(h)$

$h = O(\ln n)$

Backtracking

Divide and Conquer

Dynamic Program

Splay: M consecutive tree operation at most $O(M\log N)$

worst-case bound $\ge$ amortized bound $\ge$ average-case bound

red-black N internal nodes

at most $2\ln(N+1)$ hight

$Depth(M,N) = O(\lceil \log_{\lceil M/2\rceil} N \rceil)$

$T_{Find}(M,N) = O(\log N)$

8. Backtracking

A Template

```
bool backtracking
{ Found = false;
  if (i>N)
      return true;
  for (each x_i ∈ S_i) {
      ok = Check((x_1,...,x_i), R);
      if(ok) {
          count x_i in
          Found = Backtracking(i+1);
          if(!Found)
              Undo(i);
      }
      if (Found) break;
  }
  return found
}
```

9. Divide and conquer

General recurrence: $T(N) = aT(N/b) + f(N)$

The maximum subsequence sum – the $O(N\log N)$ solution

Tree traversals — $O(N)$

Mergesort and quicksort — $O(N\log N)$

Leftist Heaps

merge in $O(N)$

Skew heap

$O(M\log N)$ M consecutive

$T_{amortized} = O(\log N)$

Three methods for solving recurrences:

$$T(N) = aT(N/b) + f(N)$$

substitution method

recursion-tree method

master method

Let $a \ge 1$ and $b > 1$ be constants, let $f(N)$ be a function, and let $T(N)$ be defined on the nonnegative integers by the recurrence $T(N) = aT(N/b) + f(N)$, then:

1. If $f(N) = O(N^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(N) = \Theta(N^{\log_b a})$,
2. If $f(N) = \Theta(N^{\log_b a})$, then $T(N) = \Theta(N^{\log_b a}\log N)$
3. If $f(N) = \Omega(N^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$, and if $af(N/b) < cf(N)$ for some constant $c < 1$ and all sufficiently large N, then $T(N) = \Theta(f(N))$