CSE 5500 Advanced Sequential and Parallel Algorithms. Spring 2011

Exam I Solutions

1. Pick a random element of $B$ and check if this element is in $A$. Checking can be done using binary search in $O(\log n)$ time. Call these two steps a phase of the algorithm. Repeat this phase as many times as it takes to identify a common element.

The probability of success in any phase is $\geq \frac{1}{\sqrt{n}}$ since we know that there are $\sqrt{n}$ common elements between $A$ and $B$. Probability of failure in one phase is $\leq 1 - \frac{1}{\sqrt{n}}$. Therefore, probability of failing in $k$ successive phases is $\leq \left(1 - \frac{1}{\sqrt{n}}\right)^k \leq \exp(-k/\sqrt{n})$. This probability will be $\leq n^{-\alpha}$ if $k \geq \alpha \sqrt{n} \log n$. In other words, the runtime of the algorithm is $\tilde{O}(\sqrt{n} \log^2 n)$.

2. Keep two 2-3 trees $N$ and $S$. In $N$ store all the records with the name as the key for each record and in $S$ store all the records with the social security number as the key for each record. To process $\text{Find Name}(SSN)$, we search for a record whose key is $SSN$ in the tree $S$. The name in this record will be output. The run time is $O(\log n)$. We process $\text{Find SSN}(Name)$ in a similar manner.

3. Use a 2-3 tree and a counter. The value of the counter is set to zero when the 2-3 tree is initialized.

$\text{INC\_ALL}(y)$: Increment the counter by $y$. The elements in the 2-3 tree are left untouched.

$\text{INSERT}(X)$, $\text{FIND\_MIN()}$, $\text{DEL\_MIN()}$ are the standard 2-3 tree algorithms with the following changes. When an element $X$ is inserted, the value of the counter is subtracted from $X$ and the resulting value is inserted into the 2-3 tree. When an element is returned from the 2-3 tree, the value of the counter is added to the element and the resulting value is returned. All the operations take $O(\log n)$ time.

4. Recurrence relations for the run times of the two algorithms are:

$T_A(n) = 5T_A(\frac{n}{3}) + \Theta(n^2)$ & $T_B(n) = 10T_B(\frac{n}{4}) + \Theta(n^{1.8})$.

These solve to $T_A(n) = \Theta(n^2)$ and $T_B(n) = \Theta(n^{1.8})$ (using the Master Theorem) Hence, $B$ is preferred.

5. Sort $a[]$ in time $O(n \log n)$. Then run the following algorithm:

$\text{ThreeSum}(x)$

(1) for $(i = 1; i \leq n; i + +)$, $(j = 1; j \leq n; j + +)$ do
(2)   $k = \text{BinarySearch}\ (x - a[i] - a[j], 1, n)$;
(3)   if $k$ is nonzero, return $(k, i, j)$;
(4) print “not found” and exit;

Since binary search takes $O(\log n)$ time and there are $n^2$ invocations of binary search, the run time of the above algorithm is $O(n^2 \log n)$. 
6. Replace each element in each set with a tuple as follows. If $S_i = \{k_{i1}^i, k_{i2}^i, \ldots, k_{in_i}^i\}$ where $n_i = |S_i|$, then generate a sequence $X_i = (i, k_{i1}^i), (i, k_{i2}^i), \ldots, (i, k_{in_i}^i)$, for $1 \leq i \leq k$. Now sort the sequence $X = X_1, X_2, \ldots, X_k$. Since there are $n$ elements in $X$ and the elements of $X$ are integers in the range $[1, n^{11}]$, this sorting can be done in $O(n)$ time. From the above sorted sequence we can obtain each of the input sets in sorted order.