

CSE 361 Complexity of Sequential and Parallel Algorithms

Spring 2008; Exam I – Helpsheet

1. **Preliminaries.** We say $f(n) = O(g(n))$ if $f(n) \leq cg(n)$ for all $n \geq n_0$ for some constants c and n_0 . We say $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$. Also, $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

A partial list of functions in increasing order is: $O(1)$, $(\log n)^\epsilon$, $\log n$, $(\log n)^{1+\mu}$, n^ϵ , n , $n^{1+\mu}$, 2^{n^ϵ} , 2^n , $2^{n^{1+\mu}}$ where $0 < \epsilon < 1$ and $\mu > 0$ are constants.

Stirling's approximation: $n! \approx (n/e)^n \sqrt{2\pi n}$.

$$\sum_{i=1}^n i = n(n+1)/2. \quad \sum_{i=1}^n i^2 = n(n+1)(2n+1)/6. \quad \sum_{i=1}^n i^3 = n^2(n+1)^2/4.$$

2. **Master theorem.** Consider the recurrence relation: $T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$ are constants. **Case 1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. **Case 2:** If $n^{\log_b a} = \Theta(f(n))$, then $T(n) = \Theta(f(n) \log n)$. **Case 3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and $af(n/b) \leq cf(n)$ for some constant $c < 1$, then, $T(n) = \Theta(f(n))$.
3. **Randomized algorithms.** A Monte Carlo algorithm runs for a prespecified amount of time and its output is correct with high probability. By high probability we mean a probability of $\geq (1 - n^{-\alpha})$, for any constant α . A Las Vegas algorithm always outputs the correct answer and its run time is a random variable. We say the run time of a Las Vegas algorithm is $\tilde{O}(f(n))$ if the run time is $\leq cf(n)$ for all $n \geq n_0$ with probability $\geq (1 - n^{-\alpha})$, for some constants c and n_0 .
4. **Data Structures.** A dictionary supports three operations: INSERT, DELETE, and SEARCH. A (min) priority queue supports: INSERT, FindMin and DeleteMin. If one uses a 2-3 tree, each of these operations takes $O(\log n)$ time, n being the number of elements in the data structure. In the Union-Find paradigm, there are n singleton sets to begin with: $\{1\}, \{2\}, \dots, \{n\}$. We are interested in performing a sequence of union and find operations. If one uses WeightedUnion and CollapsingFind, a sequence of m operations can be completed in $O(m\alpha(m))$ time where $\alpha(\cdot)$ is the inverse Ackermanns function.
5. **Sorting.** Given a sequence of n numbers (or keys), the problem of sorting is to rearrange this sequence in either nondecreasing order or nonincreasing order. The mergesort algorithm has a worst case run time of $O(n \log n)$. The run time of quicksort is $O(n^2)$ in the worst case and $O(n \log n)$ on the average.

A sorting problem is called general sorting if the only information known about the keys is that the keys are from a linear order. Any general sorting algorithm makes use of comparison as the basic operation. It can be shown that any general sorting algorithm needs $\log(n!) = \Omega(n \log n)$ comparisons in the worst case.

We can sort n keys in $O(n)$ time if the keys are integers in the range $[1, n^c]$ (for any constant c).
6. **Selection.** Given a sequence of n keys and an integer $i(1 \leq i \leq n)$, the problem of selection is to identify the i -th smallest key from out of the n keys. Quickselect algorithm takes $O(n^2)$ time in the worst case and $O(n)$ time on the average. BFPRT algorithm takes $O(n)$ time in the worst case.
7. **Matrix Multiplication.** Strassen's algorithm takes $O(n^{\log_2 7})$ time.
8. **Convex Hull.** The convex hull of n points in the plane can be computed in $O(n \log n)$ time.