CSE 3500 Algorithms and Complexity
Fall 2016; Exam I; Help Sheet

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 $f(n) = \Omega(g(n))$. We say $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

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

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

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

2. **Master theorem.** Consider the recurrence relation: $T(n) = aT(n/b) + f(n)$, where $a \geq 1$ and $b > 1$ are constants. **Case1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. **Case2:** If $n^{\log_b a} = \Theta(f(n))$, then $T(n) = \Theta(f(n) \log n)$. **Case3:** 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 $\widetilde{O}(f(n))$ if the run time is $\leq c\alpha f(n)$ for all $n \geq n_0$ with probability $\geq (1 - n^{-\alpha})$ for some constants c and n_0.

4. **Dictionaries and Priority Queues:** A dictionary supports the operations: SEARCH (for an arbitrary element), INSERT (an arbitrary element), and DELETE (an arbitrary element). A (max) priority queue supports: INSERT (an arbitrary element), SEARCH (for the maximum element), and DELETE (the maximum element).

5. **Heaps and Heapsort:** A (max) heap is a complete binary tree where a key is stored at each node. The key at any node will be greater than the keys of its children.

   A (max) heap supports the following operations: SEARCH (for the maximum), INSERT (an arbitrary element), and DELETE (the maximum). Each operation can be completed in $O(\log n)$ time, n being the number of elements in the heap. A heap can be used to sort elements. Heapsort on n elements takes $O(n \log n)$ time.

6. **A 2-3 Tree** can be used to support a dictionary as well as a priority queue. Each operation of interest will take $O(\log n)$ time in the worst case.

7. Binary search on a sorted array of size n takes $O(\log n)$ time. Mergesort sorts n arbitrary keys in $O(n \log n)$ time. Quicksort takes $\Omega(n^2)$ time in the worst case to sort n keys. Its average run time is $O(n \log n)$.

8. We have shown that any comparison sorting algorithm will need at least $\log n!$ comparisons to sort n elements.