Instructions: (i) Submit a written assignment, or email to Khoa by the deadline. (ii) Submit one writeup per group; please discuss within your group – other resources including the Web are not allowed for consultation. Write your solutions neatly. (iii) When we refer to “Jeff Erickson’s chapter on Recursion” below, we mean the site http://compgeom.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/01-recursion.pdf

1. Consider the following decision problem: “Given a graph $G$ and an integer $k$, is the size of the maximum clique in $G$ exactly $k$?” (a) Show that this problem is NP-hard under Karp reductions. (b) If you believe this problem is in NP, then prove that it is in NP; if not, give an argument (informal is fine) suggesting why perhaps not.

2. Jeff Erickson’s chapter on Recursion, problem 11 (“Suppose we are given an array $A[1..n]$ with the special property ...”).

3. Jeff Erickson’s chapter on Recursion, problem 17 (“For this problem, a subtree of a binary tree ...”). See the page following this problem for an illustrative figure.

4. Separator theorems for graphs often lead to powerful divide-and-conquer algorithms. Recall that a graph is planar if it can be drawn in the plane with each edge being a line-segment joining its end-points, so that any two of these segments can meet - if at all - only at their end-points. The Separator Theorem for planar graphs, due to Lipton and Tarjan, says the following: given an $n$-vertex planar graph $G$, its vertices can be efficiently partitioned into three subsets $A$, $B$ and $C$ so that: (a) $n/3 \leq |A| \leq 2n/3$, (b) $n/3 \leq |B| \leq 2n/3$, (c) $|C| \leq O(\sqrt{n})$, and (d) no edge in $G$ joins any vertex in $A$ to any vertex in $B$. Thus, $C$ is a “separator” that decouples $A$ and $B$ from each other, $C$ is “small”, and neither $A$ nor $B$ is too large. This often helps us design the following type of generic divide-and-conquer algorithm for various planar-graph problems:

1. Find the separator $(A, B, C)$: this is known to be possible in $O(n)$ time.

2. Recursively solve the problem on $A$, and then on $B$.

3. Run for some $f(n)$ amount of time to process $C$ and the recursive solutions for $A$ and $B$, and to output the final correct value for $G$.

(i) Write a recurrence for the running-time $T(n)$ of this generic algorithm. (The number of edges $m$ in a planar graph is at most $3n - 6$, so we are not concerned with including $m$ as a parameter for $T$.) Remember that $|A|$ and $|B|$ can take values between $n/3$ and $2n/3$.

(ii) Solve, with proof, your recurrence for $f(n) = O(n \log \log n)$. (One of the proof techniques that we discussed, will be useful.)

(iii) Solve, with proof, your recurrence for $f(n) = O(n^2)$.

5. Read and understand Chebyshev’s inequality from Chapter 3 of Welzl’s notes. It will be useful for this problem. In class, we saw that selecting the $k$th smallest element from an array of $n$ elements can be done using an expected number of at most $4n$ comparisons. We do better now, but for simplicity we will focus on the case $k = n/2$. For the rest of this problem, “$o(1)$” will refer to any function of $n$ that goes to zero as $n \to \infty$.

So suppose we want the $(n/2)$th smallest element, say $m$, from an array $A$ of $n$ elements; assume all elements of $A$ are distinct. The basic intuition is this. In class, we picked one pivot at random. Instead, if we pick many more pivots (the multi-set $S$ below), then the median of these is likely to be quite close to the targeted element $m$. Our algorithm is as follows:
(i) Let \( s = \lceil 4n^{3/4} \rceil \); choose a (multi-)set \( S \) of \( s \) elements at random from \( A \) (say, with replacement).

(ii) Sort \( S \), and use this sorted order to (easily) find the \((s/2 - \sqrt{n})^{th}\) smallest element in \( S \), say \( a \), and the \((s/2 + \sqrt{n})^{th}\) smallest element in \( S \), say \( b \).

(iii) Let \( L = \{x \in A : a \leq x \leq b\} \). (We prove below that \( L \) contains the desired element \( m \) with high probability.) Construct \( L \) by comparing each element \( x \in A \) with \( a \) and \( b \).

(iv) Sort \( L \), and use this sorted order to find \( m \) (easily).

We now analyze this algorithm:

(a) Use Chebyshev’s inequality to prove that \( \Pr[m \in L] = 1 - o(1) \). (Write the undesirable event “\( m < a \)” in a form suitable for the application of Chebyshev’s inequality. Similarly for “\( m > b \)”.)

(b) Use Chebyshev’s inequality to prove that \( \Pr[|L| \leq n^{3/4}] = 1 - o(1) \).

(c) Give the complete details of step (iv) of the algorithm.

(d) Use the above parts to prove that the algorithm is correct with probability \( 1 - o(1) \), and that the number of comparisons is at most \( (2 + o(1))n \).

(e) Show how step (iii) can be implemented so that the expected number of comparisons becomes at most \( (1.5 + o(1))n \).