Homework 1 and Sample Solutions

1. Prove or disprove the following claims.

   (a) $2^{\lfloor \lg n \rfloor} = \Theta(2^{\lceil \lg n \rceil})$.

   Answer: *True.* $2^{\lfloor \lg n \rfloor} \geq 2^{\lceil \lg n \rceil}/2$.

   (b) $2^{2^{\lfloor \lg \lg n \rfloor}} = \Theta(2^{2^{\lceil \lg \lg n \rceil}})$.

   Answer: *False. Take* $n = 2^{2^k} + 1$. *Thus,* $\lg \lg n$ *is in between* k *and* $k+1$. *Thus* $\lfloor \lg \lg n \rfloor = k$, *and* $\lceil \lg \lg n \rceil = k+1$. *Therefore,* $2^{2^{\lfloor \lg \lg n \rfloor}} = 2^{2^k}$ *and* $2^{2^{\lceil \lg \lg n \rceil}} = 2^{2^{k+1}} = [2^{2^k}]^2$. *Therefore the statement is not true.*

2. List the following functions in increasing asymptotic order. Between each adjacent functions in your list, indicate whether they are asymptotically equivalent ($f(n) \in \Theta(g(n))$, you may use the notation that $f(n) \equiv g(n)$) or if one is strictly less than the other ($f(n) \in o(g(n))$ and use the notation that $f(n) \prec g(n)$).

$5n^3 + \log n$	2^n	$3^{n/2}$	$2^{n/3}$	$\sqrt{\lg n}$
$\ln n$	$2^{\sqrt{\lg n}}$	$\min\{n^2, 1045n\}$	$\sum_{i=1}^{n} i^{77}$	$n^{\ln 4}$
$\lfloor n^2/45 \rfloor$	$\lceil n^2/45 \rceil$	$n^2/45$	$\lg \sqrt{n}$	$\lg \lg n$
$\sum_{i=1}^{n} 1/i$	$\sum_{i=1}^{n} 1/i^2$	$\sum_{i=1}^{n} (i^2 + 5i)/(6i^4 + 7)$	$\ln(n!)$	$(\lg n)^{\sqrt{\lg n}}$

   Answer: $\sum_{i=1}^{n} 1/i^2 \equiv \sum_{i=1}^{n} (i^2 + 5i)/(6i^4 + 7) \prec \lg \lg n \prec \sqrt{\lg n} \prec \ln n \equiv \lg \sqrt{n} \equiv \sum_{i=1}^{n} 1/i \prec 2^{\sqrt{\lg n}} \prec (\lg n)^{\sqrt{\lg n}} \prec \min\{n^2, 1045n\} \prec \ln(n!) \prec n^{\ln 4} \prec \lfloor n^2/45 \rfloor \equiv n^2/45 \equiv \lceil n^2/45 \rceil \prec 5n^3 + \log n \prec \sum_{i=1}^{n} i^{77} \prec 2^{n/3} \prec 3^{n/2} \prec 2^n$.

   $\sum_{i=1}^{n} 1/i = H_n \approx \ln n + \gamma + 1/(12n) = \Theta(\ln n)$.
   $\sum_{i=1}^{n} 1/i^2 \leq \sum_{i=1}^{n} 1/[(i-1)i] = 2 - 1/n = \Theta(1)$.
   $\sum_{i=1}^{n} (i^2 + 5i)/(6i^4 + 7) = \sum_{i=1}^{n} i^2/(6i^4 + 7) + \sum_{i=1}^{n} 5i/(6i^4 + 7) \leq \sum_{i=1}^{n} i^2/6i^4 + \sum_{i=1}^{n} 5i/6i^4 = \Theta(1)$.
   $n^{77} \leq \sum_{i=1}^{n} i^{77} \leq n^{78}$.
   $(n/3)^n < n! < (n/2)^n \quad \forall n \geq 6 \qquad or \qquad n! \approx \sqrt{2\pi n}(n/e)^n$
   $\lg^{(\lg n)^{\sqrt{\lg n}}} = \sqrt{\lg n} \lg \lg n \quad \& \quad \lg \lg n = o(\sqrt{\lg n}) \implies \lg^{(\lg n)^{\sqrt{\lg n}}} = o(\lg n) \implies (\lg n)^{\sqrt{\lg n}} = o(n) \prec \Theta(n)$.

3. Solving recurrences. Find the asymptotic order of the following recurrence, represented in big-Θ notation.

   (a) $A(n) = 4A(\lfloor n/2 \rfloor + 5) + n^2$

   (b) $B(n) = B(n-4) + 1/n + 5/(n^2 + 6) + 7n^2/(3n^3 + 8)$

   (c) $C(n) = n + 2\sqrt{n}C(\sqrt{n})$ Hint: take $H(n) = C(n) + n$.

Answer:

(a) $A(n) = \Theta(n^2 \log n)$.

(b) $B(n) = \Theta(\ln n)$(Harmonic series).

(c) $C(n) = \Theta(n \log n)$.
Let $H(n) = C(n) + n$, we get : $H(n) = 2\sqrt{n}H(\sqrt{n})$
Let $H(n^{\frac{1}{2^{k-2}}})/H(n^{\frac{1}{2^{k-1}}}) = 2n^{\frac{1}{2^{k-1}}} < 8 \Rightarrow$ H(n) terminates at step k $\Rightarrow k = log_2 log_2^n$.
By multiplying $H(n)/H(\sqrt{n}), \ldots, H(n^{\frac{1}{2^{k-1}}})/H(n^{\frac{1}{2^k}})$, we get $H(n)/H(n^{\frac{1}{2^k}}) = 2^{log_2 log_2^n} n^{\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots}$.
Therefore, $H(n) = \Theta(n \log n) \qquad \Rightarrow \qquad C(n) = \Theta(n \log n)$.

4. Counting Inversions: ([KT] Chapter 5.1)

5. Karatsuba's Algorithm for Integer Multiplication: ([DPV] Chapter 2.1)

6. Finding the k-th Smallest Element (Quickselect): ([CLRS] Chapter 9.2)

7. Closest Pair of Points: ([KT] Chapter 5.4)

8. Suppose you are given a function $f(A, i)$ which sorts the subarray $A[i+1, i+2, \cdots, i+\sqrt{n}]$ in place (meaning the elements are re-arranged in the subarray) for any given $0 \le i \le n - \sqrt{n}$.

(a) Design an algorithm which only calls this function f to sort a given array $A[1..n]$. How many times do you call this function? Given the asymptotic answer in $O(\cdot)$ notation. Your algorithm is not allowed to directly compare elements in A.

Answer: First observe that if we run $f(0)$ which sorts $A[1, \sqrt{n}]$ and then $f(\sqrt{n}-1)$ to sort $A[\sqrt{n}, 2\sqrt{n}-1]$, the largest value in the first $2\sqrt{n}-1$ positions of the input array is now at position $2\sqrt{n}-1$. Similarly, if we run $f(0)$ which sorts $A[1, \sqrt{n}]$ and then $f(\sqrt{n}-i)$ to sort $A[\sqrt{n}-i+1, 2\sqrt{n}-i]$, the largest i value in the first $2\sqrt{n}-i$ positions of the input array is now moved to the rightmost.

Now we take $i = \sqrt{n}/2$. Thus by calling functions

$$f(0), f(\sqrt{n}/2), f(2 \cdot \sqrt{n}/2), f(3 \cdot \sqrt{n}/2), \cdots, f((2\sqrt{n}-2)\sqrt{n}/2),$$

we have the top $\sqrt{n}/2$ values placed in the correct position. We've used $O(\sqrt{n})$ calls.

This leaves an array of $n - \sqrt{n}/2$ numbers to sort. We apply the same strategy to find the next $\sqrt{n}/2$ highest values. Overall we use $O(n)$ function calls to sort the entire array.

(b) Prove that the algorithm you design in (a) is optimal up to a constant factor. That is, argue that no other algorithm can be asymptotically better than your algorithm in terms of the number of times to call the function f.

Answer: The total number of inverted pairs can be as high as $\Omega(n^2)$ (e.g., for a decreasing sequence). Each function call can only fix $O(n)$ inverted pairs. Thus we need $\Omega(n)$ function calls at least.

9. Given n half planes $\{H_1, H_2, \cdots, H_n\}$, we ask for an efficient algorithm to compute their intersection. Specifically, a half plane H_i is defined by an inequality $a_i x + b_i y \le c_i$ for three integers a_i, b_i, c_i (at least one of a_i, b_i is not zero for H_i to be well defined).

(a) Prove that the intersection of $\{H_1, H_2, \cdots, H_n\}$ is convex with at most n boundary edges. Here a set S is convex if $\forall x, y \in S$, the points on the line segment xy are also in S.

Answer: This can be proved by induction on n. When $n = 1$, the half plane intersection is itself which is convex. Suppose the intersection of k half planes is a convex polygon S_k with at most k boundary edges; we take the intersection with H_{k+1}. There are a few cases. If H_{k+1} does not intersect S_k, then the claim holds. Otherwise, the line on the boundary of H_k cuts the polygon S_k into two parts and only one part remains. This will increase at most one more boundary edge (which stays on the boundary of H_k) – cutting a corner of H_k out. Thus the claim holds by induction. The convexity follows by the fact that the intersection of two convex polygons is convex.

(b) Develop a divide-and-conquer algorithm with running time $O(n \log n)$.

Answer: We divide the input half planes into two sets A, B of $n/2$ half planes each. Recursively compute their intersections $S(A)$ and $S(B)$ respectively. And then take the intersection of $S(A)$ and $S(B)$. Observe that the boundary of $S(A)$ can be cut into an upper hull and a lower hull with monotonic slope at the leftmost/rightmost vertex. The intersection of two upper hulls, one from $S(A)$ and one from $S(B)$, can be done in linear time. Same for the two lower hulls. By master theorem, we have $O(n \log n)$.

10. A Toeplitz matrix is an $n \times n$ matrix $A = (a_{ij})$ such that $a_{ij} = a_{i-1,j-1}$ for $i = 2, 3, \cdots, n$ and $j = 2, 3, \cdots, n$.

(a) Is the sum of two Toeplitz matrices necessarily Toeplitz? What about the product?

Answer: Yes for the sum, which is trivial to prove.

For the product of two Toeplitz matrices A and B. We check their product $C = AB$. For $i = 2, 3, \cdots, n$ and $j = 2, 3, \cdots, n$,

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = a_{i1}b_{1j} + a_{i-1,1}b_{1,j-1} + \cdots + a_{i-1,n-1}b_{n-1,j-1}$$

Also,

$$c_{i-1,j-1} = a_{i-1,1}b_{1,j-1} + a_{i-1,2}b_{2,j-1} + \cdots + a_{i-1,n-1}b_{n-1,j-1} + a_{i-1,n}b_{n,j-1}$$

Thus $c_{ij} - c_{i-1,j-1} = a_{i1}b_{1j} - a_{i-1,n}b_{n,j-1}$. This matrix C is not necessarily Toeplitz.

(b) Describe how to represent a Toeplitz matrix so that two $n \times n$ Toeplitz matrices can be added in $O(n)$ time.

Answer: The value of an element in a Toeplitz matrix propagates along the lower-right diagonal direction. Thus we only need to remember those elements that do not have an upper left element – the first row and the first coulmn. We represent a Toeplitz matrix A by a vector of length $2n - 1$

$$R = (a_{n1}, a_{n-1,1}, \cdots, a_{11}, a_{12}, \cdots, a_{1n}).$$

Essentially, we trace the elements in A from the bottom element of the first column upwards until we reach a_{11} and then follow the first row.

The sum of two Toeplitz matrices A and B can be implemented by taking the sum of their vector representation.

(c) Give an $O(n \log n)$ algorithm for multipling an $n \times n$ Toeplitz matrix by a vector of length n. Use your representation in the previous part.

Answer: Take $y = Ax$ where x is a column vector $(x_1, x_2, \cdots, x_n)^T$. If we represent A by a vector R of length $2n-1$,

$$R = (r_1, r_2, \cdots, r_{2n-1})$$

We can check that $y_i = x_1 r_{n-i+1} + x_2 r_{n-i+2} + \cdots + x_n r_{2n-i}$. We use FFT for this problem. Basically, we construct two polynomials

$$p(z) = r_1 z^0 + r_2 z^1 + \cdots + r_{2n-1} z^{2n-2}$$

and

$$q(z) = x_1 z^{n-1} + x_2 z^{n-2} + \cdots + x_n z^0.$$

We can check that y_i is the coefficient of the polynomial $p(z) \cdot q(z)$ with degree $2n-i-1$. Multiplying two polynomials can be done in time $O(n \log n)$.