

CS 473 (ug): Combinatorial Algorithms, Fall 2005

Homework 0

This homework is not to be turned in, but you should do it to check your understanding of the pre-requisite material.

This homework tests your familiarity with prerequisite material—big-Oh notation, elementary algorithms and data structures, recurrences, discrete probability, and induction—to help you identify gaps in your knowledge. **You are responsible for filling those gaps on your own.** For review, you can check chapters 2 and 3 of the 473 textbook, your discrete mathematics and data structures textbooks, or the first few chapters of CLRS.

1. Solve the following recurrences. State tight asymptotic bounds for each function in the form $\Theta(f(n))$ for some recognizable function $f(n)$. Assume reasonable but nontrivial base cases.

(a) $A(n) = 2A(n/4) + \sqrt{n}$

(b) $B(n) = \max_{n/3 < k < 2n/3} (B(k) + B(n-k) + n)$

(c) $C(n) = 3C(n/3) + n/\lg n$

(d) $D(n) = 3D(n-1) - 3D(n-2) + D(n-3)$

(e) $E(n) = \frac{E(n-1)}{3E(n-2)}$ [Hint: This is easy!]

(f) $F(n) = F(n-2) + 2/n$

(g) $G(n) = G(\log n) + \log n$

(h) $H(n) = H(n/2) + 1$

(i) $I(n) = I(n/2) + I(n/4) + I(n/8) + I(n/12) + I(n/24) + n$

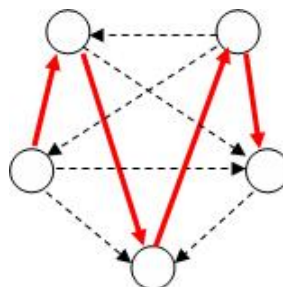
(j) $J(n) = 2J(\sqrt{n}) + n^2$

2. Sort the following functions from asymptotically smallest to asymptotically largest, indicating ties if any.

$$\begin{array}{cccccc} \sum_{i=1}^n i & n^{\lg \lg n} & 3^n & n^{1/n} & (\lg n)^{\lg^2 n} \\ \lg(n!) & \sum_{i=1}^n (i^2 - (i-1)^2) & 2^{\lg(2^n)} & \sum_{i=1}^n \frac{1}{i} & \lg^* 2^{2^n} \end{array}$$

Where $\lg^* n$ is the iterated logarithm (to the base 2) of n . For more information about iterated logarithms, see [the Wikipedia article on iterated logarithms](#).

3. A *tournament* is a directed graph, such that every pair of vertices has one edge between them. (That is, for any vertices u, v there is either an edge from u to v or an edge from v to u .) A Hamiltonian Path in a graph is a path that visits every vertex exactly once. Show that every tournament has a Hamiltonian Path. The figure below shows a 5-vertex tournament, with the edges corresponding to one Hamiltonian Path highlighted.)



4. Consider this algorithm to sort an array A of n distinct numbers:

```
Pick two indices  $i, j$  uniformly at random from  $\{1, 2, \dots, n\}$ 
If  $A[\min(i, j)] > A[\max(i, j)]$ 
    swap the elements in positions  $i$  and  $j$ 
Repeat
```

The algorithm automagically stops once the array is sorted.

- Prove that after $O(n^2)$ **swaps**, the algorithm must halt.
- Show that if the array is unsorted, the algorithm will perform a swap in a single iteration with probability at least $\frac{2}{n^2}$.
- Hence, find an upper bound on the expected number of iterations to perform a swap.
- Prove that the expected running time of the algorithm is $O(n^4)$.
- Make a simple modification to the algorithm that eliminates the need for the automagical stop *without* increasing the overall running time. (That is, the modified algorithm should still have running time $O(n^4)$.)