

CS 473: Algorithms, Fall 2016

HW 0 (due Tuesday, August 30th at 8pm)

This homework contains three problems. **Read the instructions for submitting homework on the course webpage.**

Collaboration Policy: For this home work, each student should work *independently* and write up their own solutions and submit them.

1. Solve the following recurrences in the sense of giving an asymptotically tight bound of the form $\Theta(f(n))$ where $f(n)$ is a standard and well-known function. No proof necessary for the first four parts; simply state the bound.

- (a) $A(n) = n^{1/3}A(n^{2/3}) + n$, $A(n) = 1$ for $1 \leq n \leq 8$.
- (b) $B(n) = B(n/2) + \sqrt{n}$, $B(1) = 1$.
- (c) $C(n) = 3C(n-1) + 1$, $C(1) = 0$.
- (d) $D(n) = 3D(n/3) + 4D(n/4) + n^3$, $D(n) = 1$ for $n \leq 4$.
- (e) *Prove* by induction that the $T(n)$ defined by the recurrence

$$T(n) = n + \frac{1}{n} \sum_{i=1}^{n-1} (T(i) + T(n-i-1))$$

with $T(n) = 1$ for $n \leq 2$ satisfies the bound $T(n) = O(n \log n)$.

2. Let $G = (V, E)$ be directed graph with non-negative edge lengths where $\ell(e)$ denotes the length of edge $e \in E$ that represents travel distances in a network. The nodes of the graph are partitioned into red, blue, white and black sets denoted by R, B, W, K respectively. Describe an algorithm that given two distinct nodes $s, t \in V$ finds a shortest walk that contains at least one red, one blue and one white node. You should give a *reduction* from this problem to solving a shortest path problem in a directed graph; that is, assume that you have access to an algorithm that given a directed edge-weighted graph H and two nodes u, v computes a u - v shortest path (for instance Dijkstra's algorithm accomplishes this). Do *not* describe a customized algorithm. You need to prove the correctness of your reduction.
3. Consider the standard balls and bins process. A collection of m identical balls are thrown into n bins: each ball is thrown independently into a bin chosen uniformly at random.
 - (a) What is the (precise) probability that a particular bin i contains exactly k balls at the end of the experiment? Let X be the (random) number of bins that contain exactly k balls. What is the expected value of X ?
 - (b) What is the variance of X ?
 - (c) Consider the same experiment but with $m = n$. Now balls and bins are numbered from 1 to n . We say that there is a match in bin i if at the end of the experiment it contains the ball numbered i (it could contain other balls as well). What is the probability that there are exactly k matches?

Explain your calculations when you derive the bounds.

The remaining problems are for self study. Do *NOT* submit for grading.

- Sort the following functions from asymptotically smallest to asymptotically largest. Your answer should be a sorted list. Note that there may be ties.

$$\begin{array}{cccc} \ln \ln n & (\sqrt{3})^{\lg n} & n^{1+1/\lg n} & e^{\ln \ln n} \\ \sqrt{n} & H_n & 4^{H_n} & n^{1.2} \\ \lg^{\lg n} n & n^n & n! & (1 + 1/n)^{3n \ln n} \end{array}$$

To jog your memory: $\lg n = \log_2 n$ and $\ln n = \log_e n$ and $H_n = \sum_{i=1}^n 1/i \simeq \ln n + 0.577215 \dots$ is the n 'th harmonic number.

- Let $G = (V, E)$ be an undirected and connected graph. Fix a vertex $u \in V$. Let \mathcal{T}_1 be the set of all spanning trees that can be obtained by doing a depth first search (DFS) in G starting with u . Note that there are multiple DFS trees possible because of the choice available in exploring the neighbors of a node. And let \mathcal{T}_2 be the set of all spanning trees that can be obtained by doing a breadth first search (BFS) starting with u . Prove that there are trees $T_1 \in \mathcal{T}_1$ and $T_2 \in \mathcal{T}_2$ such that $T_1 = T_2$ (in the sense that they have the same set of edges) if and only if G is a tree. *Hint: Understand how DFS and BFS explore the graph if it is a simple cycle.*
- A standard priority queue data structure stores keys that have associated priorities and supports two basic operations: insertion of a key with a given priority, and extracting a key with the highest priority. Priority queues can be implemented at the cost of $O(\log n)$ per operation where n is the number of keys stored in the data structure. In some applications the number of distinct priorities, k , is often very small compared to the number of keys and the parameter k is often known in advance. Design a modified priority queue data structure that supports insertions and extract in $O(\log k)$ time per operation. Your data structure knows k in advance and should report an error if more than k distinct priorities are inserted into it at any point. You should assume that when extracting a key with highest priority it does not matter which of the keys with that priority is returned. *Hint: combine a standard priority queue with an auxiliary standard data structure.*
- See Homework 0 from Spring 2016 taught by Jeff Erickson. <https://courses.engr.illinois.edu/cs473/sp2016/hw/hw0.pdf>.