CSE 101 Homework 4

Spring 2016

This homework is due Friday May 13th, 11pm on gradescope. Remember to justify your work even if the problem does not explicitly say so. Writing your solutions in LaTeX is recommended though not required.

Question 1 (Hadamard Matrices, 25 points). Textbook problem 2.28. This is a good algorithm to know as these matrices do show up and it’s important that you can multiply by them very quickly. Also the idea for this is similar to the idea for the Fast Fourier Transform.

Question 2 (Online Caching, 30 points). In practice the furthest in the future algorithm is not usually used for making caching decisions, since while it’s optimal, it requires being able to look ahead and know future memory accesses in advance. In practice, one often needs an online algorithm, that is one that only knows about previous accesses when it makes the decision about what to eject at any given time. In practice, a slightly different greedy approach is often used. When a memory cell needs to be dropped from cache, you choose the cell that was used least recently. This is the least recently used (or LRU) algorithm.

(a) Show that there are some access sequences so that the LRU algorithm with a cache of size $k$ has $k$ times as many cache misses as the FITF algorithm. [10 points]

(b) Consider any sequence of accesses during which at most $k$ distinct memory locations are addressed. Show that the LRU algorithm with cache size $k$ makes at most $k$ cache misses over such an interval. [10 points]

(c) Show that for any sequence of memory accesses that the number of cache misses obtained by the LRU algorithm with a cache of size $2k$ is at most twice the number of cache misses needed by the best offline algorithm using a cache of size $k$. Hint: break the access sequence into time intervals in which exactly $2k$ distinct memory cells are accessed. [10 points]

Question 3 (Other Minimum Spanning Tree Algorithms, 45 points). Let $G$ be a graph whose edges are given distinct weights. For each of the following proposed greedy algorithms for computing a minimum spanning tree in $G$, either prove that the algorithm works, or provide an example of such a $G$ for which the algorithm obtains a wrong answer.

(a) Remove the heaviest edge whose removal does not disconnect $G$. Repeat until $G$ is a tree. [15 points]

(b) Pick two vertices $u, v$ in $G$. Add the shortest path between $u$ and $v$ to the tree, and glue all vertices on the path together. Repeat until $G$ has a single vertex. [15 points]

(c) For each vertex $v$, add the lightest edge incident to $v$ to the tree. Glue vertices along all such edges. Repeat until $G$ is a single vertex. [15 points]

Question 4 (Extra credit, 1 point). Approximately how much time did you spend working on this homework?