Unofficial, bootleg, underground, blackmarket and generally hush-hush final review worksheet

by Rebel Elements within the CSE 326 staff

Thursday, March 2, 2000

Welcome to the Rebel Alliance's secret review worksheet for the final exam in CSE 326. Our sources inform us that

- The final exam will be comprehensive, and will cover topics from the entire quarter.
- The final will focus on topics after the midterm.
- This secret communication (worksheet) will cover only topics after the midterm.
- For review of the midterm topics, please see the midterm itself, and the worksheets. All are available on the web.
- Our sources have taken cyanide tablets after relaying this information, in order to avoid compromises to the Rebel Alliance.

To minimize the risk of discovery and telepathic strangulation by Darth Wolfman, destroy this worksheet as soon as you have used it. Our goal is to have everyone ace the final exam, without Darth Wolfman knowing how. We're counting on you. Help me, students, you're my only hope.

Unless marked "harder" or something like that, all questions are fair game for the midterm. All topics tested are fair game, even those in the "harder" questions.

1 Deactivating the Death Nebula

We felt a new hope after destroying the evil Death Star, mainly because it was so easy to just drop a bomb down a shaft and blow up the whole thing. But, this time, the Empire has designed a new instrument of evil destruction, called the "Death Nebula". The Death Nebula is not so easy to destroy.

In fact, destroying the Death Nebula will require a complicated series of disabling systems. Each system has checking systems. The design of the Death Nebula means that if we disable one security system before disabling its checking systems, we will be discovered. Systems may check multiple other systems.

The Rebel plan to defeat the Death Nebula. We need to find an order in which to disable the security systems so that the alarms don't go off. Using the Force, we have discovered that such an order does exist.

But, in addition to finding some order, we'd also like to disable some systems before others, where possible. Some systems are involved in more evil uses of the Dark Side than others. Our operatives know which systems check which other systems, and also what amount of evil is inherent in each system.

If S is a system, we are given Evil(S), its evil. We are also given Checkers(S), the set of other systems that check S (so we need to disable everything in Checkers(S) before disabling S, if we want to avoid certain death in the Death Nebula).

Luke Graphwalker. A long, long time ago, in a galaxy far, far away, one of our Jedi Programmer, who studied at the University of Washington-Prime, suggested that we could perform a topological sort. In the topological sort, we maintain lists of nodes with 0 in-edges, and choose one of these. The algorithm proceeds in the usual manner (as in a recently decoded message send by none other than Darth Wolfman himself).

Unfortunately, our Jedi Programmer was unable to find an appropriate data structure to keep the 0 in-edges and an associated algorithm, to make sure we disable more evil systems before others, where possible.

Your mission. Consider the following ADTs, and say which of these might be appropriate. What data structures should be used for any ADTs that are appropriate? Justify your answers; the lives of Jedi Programmers and the fate of the galaxy hang in the balance.

- A queue.
- A stack.
- A priority queue.
- A multi-dimensional dictionary.
- A lightsaber.

Work quickly, and may the Force be with you.

2 Sorting by 4-way Comparisons

A Jedi Programmer has suggested that we can break the lower bound on the worst case performance of a sorting by comparison algorithm. It[1] says that a 4-way Comparison allows a sorting algorithm to compare 4 different numbers *simultaneously* an in one operation determine which is biggest, next biggest, next biggest, and smallest.

[1]This Jedi Programmer is from Anxon-5

Some of us believe that this new technique may allow us to create sorting-by-comparison algorithms that have a worst case behavior in $o(n \log n)$. Others believe that this Jedi Programmer has joined the dark side, and that its idea is unsound. Who is right, and how can we be sure of this?

3 Advice

What is the best representation of a graph for a typical topological sort?

4 (harder) Find the Median

The Rebel Alliance has found that sending a very bad Jedi Programmer to complete a mission generally results in failure. Less obviously, sending a very good Programmer generally results in that programmer being overly-confident, and ultimately too careless.

Therefore, we plan to rank all Jedi Programmers, and send the one with the middle-most ability. Clearly, we need a fast algorithm to find the median - we can't afford to delay missions.

Propose an algorithm that finds the median in an expected time of $O(n)$, but may take $O(n \log n)$ in the worst case (formal analysis of expected time not necessary). Master Yoda suggests, "Fast algorithm to find, think of you QuickSort partition must".

P.S.: Beyond the scope of this mission, one of our operatives working undercover, who has infiltrated Darth Wolfman's organization and is known only as "Z", has suggested that an algorithm with worst case $O(n)$ behavior also exists. However, it is too hard to think of - it is only known since it mysteriously appeared in books in 1973.

5 A Job for an Injured Programmer

One of our star programmers was tragically injured in a light-pen fight. Although she retains her awesome programming skills, she is unable to use recursion. Unfortunately, her initial plan was to code Find in an Up-Tree using a recursive implementation (she is finding a Minimum Spanning Tree to lay wire between Rebel bases in the galaxy).

How can she implement the Find operation without using recursion? Given that the Force is strong within her, how do you think she will implement this to use only a constant amount of extra storage while traversing the path through the Up-Tree?

6 Demographics for Jedi Programmers

We are always in need of promising Jedi apprentices. We downloaded a galaxy census containing demographic information on all members of the galaxy. In particular, we are interested in age ranges and Galaxy-Wide Fencing Ability Scores.

Sometimes students who are very good at fencing are never able to truly make the adjustment to light-pens, and are not good Jedi Programmers. Students who are poor at fencing demonstrate a low aptitude for light-pen work.

As the Jedi Programming Council considers different ranges of ages and Fencing Abilities, they would like to be able to query the census for good candidates to undergo Jedi training.

What is an appropriate ADT for this problem? What is an appropriate data structure?

7 Predicting the Enemy

Our sources inform us that Darth Wolfman is fed up with the printers, and needs to have a priority queue of printer jobs, which he can merge when printers fail. However, we know that Darth Wolfman is *extremely* right-wing in his politics. Therefore, we know he would never use a Leftist Heap for this, or the closely-related Skew-Heap.

We need to know what data structure(s) he might use, so we can be prepared. Please help.

8 Fast hash tables with good worst case performance

Darth Wolfman has won the galaxy-wide programming contest six millennia in a row. We need to break his winning streak. Our current challenge has to do with hash tables.

Hash tables are very fast in the expected case, but for some data the hashing function may not be so good. In this case, things bunch up in the hash table, making performance slower than a Womp Rat in the 5th Desert Moon of Therbidius Prime[2].

1. If we can change our hashing function, how should we change it in order to try and fix the problem.

2. If we cannot change our hashing function, what kind of a hashing table can we use to guarantee $O(\log n)$ behavior in the worst case? How can we make things as fast as possible in the common case, where things aren't bunching up too badly?

[2] It's very slow; you don't want to know.