

CSE 101 Homework 3
Due Thursday, March 3
Back-Tracking and Greedy Algorithms

Directions: For each of the first four problems, a "high level" greedy strategy is given. For some of the problems, the strategies give a correct (optimal) solution, and for others, it sometimes gives incorrect (suboptimal) solutions. For each, decide whether the greedy strategy guarantees optimal solutions. If it is, give a proof that it is correct, then describe what data structures and preprocessing you would use to give an efficient version, and give a time analysis.

If it is not correct, give a counter-example showing the strategy is incorrect. Then give a back-tracking algorithm for the problem, and give an upper bound on the time your algorithm would take.

For the last problem, you need to turn in two tables of values. You will need to write and run a program to generate the values, but all you should hand in are the numerical values (and the input sizes they correspond to).

Largest Independent Set for a tree The problem is to find the largest independent set for the special case when the input graph is a tree. (Edges are between nodes and their parents and children.) Remember, an independent set S of a graph is a set of nodes that does not contain both of the endpoints of any edge, i.e. for any edge $\{x, y\}$ either $x \notin S$ or $y \notin S$. So here, we must have a set of nodes of the tree S so that we cannot have both a node and its parent in the set.

Greedy strategy: Find any leaf x in the tree, i.e., any node with no children. Add x to S . Delete x and x 's parent from the tree. Repeat. (Note: when you repeat, the graph might become a *forest*, a collection of unconnected trees, rather than a single tree. In that case, x could be any leaf of any tree. If x does not have a parent, then only it is deleted.)

Library storage Consider the following problem. A library has n books that must be stored in alphabetical order on adjustable height shelves. Each book has a height and a thickness. The width of the shelf is fixed at W , and the sum of the thicknesses of books on a single shelf must be at most W . The next shelf will be placed on top, at a height equal to the maximum height of a book in the shelf. Give an algorithm that minimizes the total height of shelves used to store all the books. You are given the list of books in alphabetical order, $b_i = (h_i, t_i)$, where h_i is the height and t_i is the thickness.

Greedy strategy: Put books on the first shelf until the total width would exceed W . Then start the next shelf, solving the same problem recursively with the remaining books.

Halloween Route Consider the following problem. The input is a positive real number T and a labelled graph whose nodes represent houses in a

neighborhood. Each house is labelled with the number of pieces of candy that the residents give to trick-or-treaters. The edges represent paths between the houses. Each edge is labelled with the time it takes to walk down the path in your Halloween costume. Also given as part of the input is the total time T you can stand to be in your hot uncomfortable costume. Your goal is to find a route through the graph starting at your house (but it doesn't have to end there) that maximizes the amount of candy you get. You can go by the same house more than once, but you only get candy the first time.

Candidate greedy strategy: Until you run out of time do: from the current house, go to the neighbor that maximizes the ratio of candy available at the house and the distance to the house (either directly or indirectly, through other houses, from the current house). Treat visited houses as having 0 candy.

Oxen pairing Consider the following problem: We have n oxen, Ox_1, \dots, Ox_n , each with a strength rating S_i . We need to pair the oxen up into teams to pull a plow; if Ox_i and Ox_j are in a team, we must have $S_i + S_j \geq P$, where P is the weight of a plow. Each ox can only be in at most one team. Each team has exactly two oxen. We want to maximize the number of teams.

Candidate Greedy Strategy: Take the strongest and weakest oxen. If together they meet the strength requirement, make them a team. Recursively find the most teams among the remaining oxen.

Otherwise, delete the weakest ox. Recursively find the most teams among the remaining oxen.

Implementation of Independent Set Consider a greedy heuristic for independent set that selects the lowest degree node, puts it in the set, and deletes it and its neighbors and repeats. Implement this greedy heuristic, and a correct backtracking algorithm for Maximum Independent Set. Run both algorithms on random graphs where the graphs are constructed by adding edges between each pair of nodes independently with probability $1/2$. For various input sizes n , plot the average sizes of the greedy heuristic sets compared to the optimal independent sets found by the backtracking algorithm. You should try to get data for n as large as possible, and plot the results on a $\log n$ vs. size found (note: not log size found) graph. Can you give a conjecture as to how the independent set grows as a function of size? (Don't hand in code, but describe your algorithm, programming language, and architecture. Your grade for this problem will be partially based on how large n you can get your backtracking algorithm to run on.)