

COMP3711: Design and Analysis of Algorithms

Tutorial 6

HKUST

Question 1

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

- (a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.
- (b) Suppose that the available coins are in the denominations that are powers of c . i.e. the denominations are $c^0, c^1, \dots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.
- (c) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n .

Question 2

In the old days, files were stored on tapes rather than disks. Reading a file from tape isn't like reading a file from disk; first we have to fast-forward past all the other files, and that takes a significant amount of time. Suppose we have a set of n files that we want to store on a tape, where file i has length $L[i]$. Given the array $L[1..n]$, your job is to design an algorithm to find the optimal order to store these files on a tape to minimize the cost. Note that the cost of reading file i is total length of all files stored before it, including file i itself. Your algorithm should run in $O(n \log n)$ time.

Question 2

- (a) Suppose each file is accessed with equal probability, and you want to minimize the expected cost. For example, if $L[1] = 3, L[2] = 6, L[3] = 2$, you would want to use the order $(3, 1, 2)$. This way, the expected cost is $2/3 + (2 + 3)/3 + (2 + 3 + 6)/3 = 6$, which is optimal. You need to prove the optimality of your algorithm.
- (b) Suppose the files are not accessed uniformly; file i will have probability $p[i]$ to be accessed. Given the array $L[1..n]$ and $p[1..n]$, how would you find an ordering that minimizes the expected cost? For example, if $L[1] = 3, L[2] = 6, L[3] = 2$ and $p[1] = 1/6, p[2] = 1/2, p[3] = 1/3$, then the optimal ordering would be $(3, 2, 1)$ with an expected cost of $2/3 + (2 + 6)/2 + (2 + 6 + 3)/6 = 6.5$. Remember to prove the optimality of your algorithm.