Analysis of Algorithms: 354 — YOUR NAME HERE
CSU Channel Islands — Dec 8, 2020
Ryan McIntyre — Final Exam

Final Exam

This final is due **Thursday, Dec 10** at **11:59PM**. Note that you must take an interview exam in addition to this written exam; failure to interview will result in a failing grade on the final exam, of which this is only half.

Problem 3-1. [15 points] **Proof of Correctness**

Consider a problem in which you have a list J of jobs, each of which takes the same amount of time to complete. You need to schedule these jobs in array S, indexed from 0 to n. Each job occupies a single index in the array, because they each take the same amount of time to complete. Each job $j \in J$ has:

- **deadline** $j.deadline$ from 0 to n, denoting the latest index in S at which it can be scheduled.
- **profit** $j.profit$, denoting the profit or gain from completing job j.

The goal is to maximize the sum of profits of the jobs scheduled in S. $|J|$ can be greater than $|S|$, so it might not be possible to schedule all jobs. There is no penalty for leaving a job undone.

Consider the following algorithm:

Job Scheduling

IN : List J of jobs, max index n

$S \leftarrow$ an array indexed 0 to n, with $null$ at each index
Sort J in non-increasing order of profits
for i *from* 0 *to* n
    Find the largest t such that $S[t] = null$ and $t \leq J[i].deadline$ (if one exists)
    if *an index* t *was found*
        $S[t] \leftarrow J[i]$

OUT: S maximizes the profit of scheduled jobs

Note that $\leftarrow$ is used to denote assignment; $x \leftarrow y$ means that x is assigned the value of y.

Prove that the algorithm is correct.

HINT: A schedule S is **promising** if it can be extended to an optimal schedule by scheduling jobs that have not yet been considered. If S is promising after all jobs have been considered, then S is an optimal schedule.

Problem 3-2. [15 points] **Time Complexity**

Derive the time complexity of the following algorithm. Show your work.

MULT

IN : x and y, two n-bit binary numbers

if $b = 1$
    if $x = 1 \land y = 1$
        return 1
    else
        return 0
$(x_1, x_0) \leftarrow$ (first $\lceil n/2 \rceil$ bits , last $\lfloor n/2 \rfloor$ bits) of x
$(y_1, y_0) \leftarrow$ (first $\lceil n/2 \rceil$ bits , last $\lfloor n/2 \rfloor$ bits) of y
$z_1 \leftarrow \text{MULT}(x_1 + x_0, y_1 + y_0)$
$z_2 \leftarrow \text{MULT}(x_1, y_1)$
$z_3 \leftarrow \text{MULT}(x_0, y_0)$
return $z_2 \cdot 2^n + (z_1 - z_2 - z_3) \cdot 2^{\lceil n/2 \rceil} + z_3$

OUT: The product of x and y

Some details:

- All additions and subtractions take $O(n)$ time.
- In the final return line, the multiplications by $2^{\text{some power}}$ are actually bit shifts, and take $O(n)$ time.
- The base case takes $O(1)$ time.

Problem 3-3. [20 points] **Algorithmic Design**

Given a sequence $R = \{r_1, r_2, ..., r_n\}$ of real numbers, let S_{ij} denote the sum $r_i + r_{i+1} + ... + r_{j-1} + r_j$; in other words, S_{ij} is the sum of all elements in R from index i to index j.

Write an algorithm which takes a sequence of real numbers R as an input and efficienltly finds a maximum sum S_{ij}.

For instance, given the sequence $R = \{1, -2, 1, 2, 4, -9, 6\}$, your algorithm should output the number 7, which results from adding 1, 2 and 4; this is the largest sum which can be acquired by adding consecutive elements of R.

Note that while this is easily doable on $O(n^2)$ time by simply finding the sum of every consecutive subsequence, it is doable in $O(n)$ time if the subproblems are chosen more carefully.

Your pseudocode should be concisely and clearly written; convoluted, messy, or unnecessarily complex solutions will lose points.