CS 341, Winter 2022 R. Peng, A. Storjohann

MIDTERM EXAM

Released: Thursday February 17, 4:30 PM.
DUE: Friday February 18, 4:30 PM.

IMPORTANT INSTRUCTIONS

There are 3 questions in this exam, each worth 10 marks, for a total of 30 marks.

For an algorithm you must include (unless specified otherwise): a description in English; pseudocode (unless the algorithm is really simple); correctness proof; runtime analysis. Always try for the fastest algorithm you can.

YOU MUST HANDWRITE AND SIGN THE FOLLOWING STATEMENT

(e.g., write it on paper, sign it, take a photo, and include that with your Crowdmark submission)

I promise, on my honour, that the work on this exam is my own. Apart from the course materials and private Piazza questions, I have not given or received any assistance on this exam.

STANDARD MATHEMATICAL CONVENTIONS

- Nonnegative integers means the set $\mathbb{Z}_{\geq 0} = \{0, 1, 2, \ldots\}$.
- Positive integers means the set $\mathbb{Z}_{>0} = \{1, 2, 3, \ldots\}$.
- Let $\emptyset = \{\}$ denote the empty set. Then $\min \emptyset = \infty$ and $\max \emptyset = -\infty$.

1. [10 marks] **Power of a Multi-Precision Integer.**
Let X be an n-bit multi-precision integer. The following recursive algorithm computes X^k for some positive integer k. It uses a bit-multiplication algorithm `Multiply`(Y, Z) that takes as input two multi-precision integers Y and Z and returns their product as a multi-precision integer in $O(\max(\lg X, \lg Y) \min(\lg X, \lg Y)^{\log_2 3-1})$ time (Karatsuba's algorithm).

```
Power(X, k)
  if k = 1 then return X
  if k > 1 then
    Let Y = Power(X, ⌊k/2⌋)
    Y ← Multiply(Y, Y)
    if k is odd then Y ← Multiply(Y, X)
    return Y
```

Set up a recurrence for the number of bit operations used by this algorithm to compute X^k, then solve it to give the asymptotic running time. You may use any method from the lectures, for example, the Master Theorem, recursion tree, or guess and prove by induction. Show your work.

2. [10 marks] **Volatile Knapsack.**
You are trying to fill a knapsack with volatile objects. There are n types of objects labelled from 1 up to n. Each type of object has a positive integer weight w_i, and a positive integer volatility level v_i, $1 \leq i \leq n$. There are an infinite number of objects of each type. There will be at least one type of object with weight 1.

You want a collection of items with total weight exactly W, but minimizing the max volatility of an object used. The target weight W is a nonnegative integer.

Example 1. *Suppose* $W = 10$ *and there are three types of objects:*

- $w_1 = 1$, $v_1 = 10$
- $w_2 = 3$, $v_2 = 7$
- $w_3 = 4$, $v_3 = 6$

Then we can obtain a volatility of 7 by using 2 objects of type 2, and 1 object of type 3. In particular, $2w_2 + w_3 = 10$ *and* $\max(v_2, v_3) = 7$.

Give an $O(Wn)$ time dynamic programming algorithm for finding this minimum max volatility. Algorithms running slower than this, but faster than $O((Wn)^9)$ will still receive partial marks. As with the assignments, your dynamic programming solution must consist of

- A description of the subproblems you will solve and the order in which you will solve them.
- A formula for solving one subproblem in terms of previously solved subproblems, and a justification of correctness.
- Pseudocode of your algorithm.
- An analysis of run-time using big-O notation.

3. [10 marks] **Seating Arrangements.**
There are $n \geq 1$ chairs in a row, arranged from left to right, labelled from 1 up to n. There are $m \geq 1$ people. Corresponding to person j ($1 \leq j \leq m$) is a leftmost chair $\ell[j]$ and rightmost chair $r[j]$ such that they can only be seated in a chair in the contiguous range $\ell[j], \ell[j]+1, \ldots, r[j]$.

Notes:

- Each chair can seat at most one person.
- Each person can sit in at most one chair.
- $1 \leq \ell[j] \leq r[j] \leq n$, so each person is willing to sit in at least one chair.
- There is no relationship between n and m. We could have $n = m$, $n < m$ or $n > m$.

Give an $O(nm)$ time (or faster) algorithm to seat the maximum number of people. An $O(nm)$ time solution will obtain full marks.