1 Notes on grading.

- You can write “I don’t know” for any question and receive 25% credit. You can take this option for any numbered problem, but not for part of a problem. For example, you can answer 3.1 and write “I don’t know” for 3.2, but you can’t write part of the solution for 3.2 and then write “I don’t know” for the rest.

- You get a 10% bonus for typing your homework. You are encouraged to use \LaTeX. You must type your entire homework to receive the bonus. The 10% bonus does not apply to problems answered with “I don’t know.”

2 Rotating images

A gray-scale image of size \((n \times n)\) is a \((n \times n)\) matrix of integers. Given that we have a very quick algorithm \texttt{rectangularCopy()} to copy a rectangular chunk of pixel of size from one location to another:

- 1. Design a divide-and-conquer algorithm to rotate an image \(90^\circ\) clockwise. Assume that \(n\) is a power of 2. (You can use figures to help illustration).

- 2. If \(n\) is an arbitrary integer, how to modify the algorithm in question 1?

- 3. If the running time of \texttt{rectangularCopy()} on \((a \times a)\) matrix is \(O(a^2)\), what’s the running time of your algorithm?

- 4. If the running time of \texttt{rectangularCopy()} on \((a \times a)\) matrix is \(O(a)\), what’s the running time of your algorithm?
3 Applications of findRankKElt

1. Show how to find all the \( k_1 \)- through \( k_2 \)-th-smallest elements of an array \( A \) of \( n \) distinct elements in \( O(n) \) time.

2. Show how to determine in linear time whether any element of an array \( A \) of \( n \) elements occurs at least \( n/2 \) times.

3. Given \( n \) distinct elements \( X = \{x_1, \ldots, x_n\} \) with weights \( w_1, \ldots, w_n \) such that \( \sum_i w_i = 1 \), the weighted median of \( X \) is the element \( x_k \) such that

\[
\sum_{i: x_i < x_k} w_i < \frac{1}{2}
\]

and

\[
\sum_{i: x_i > x_k} w_i \leq \frac{1}{2}
\]

Show how to compute the weighted median in linear time.

4 Finding the lonely element

Given a sorted array in which each number occurs twice except one number. Find the unique number using an order of less than \( O(n) \) time. Hint: \( O(\log n) \) time.

Example: \{1,1,2,3,3,5,5,8,8,13,13\}.

5 A very frequent element

An array \( A[1..n] \) is said to have a dominant element if more than half of the entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a dominant element, and, if so, find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is \( A[i] > A[j] \)?". (Think of the array elements as GIF files, say.) However, you can answer questions of the form "is \( A[i] = A[j] \)?" in constant time.

(a) Show how to solve the problem in \( O(n \log n) \) time.

(b) Can you give a linear time algorithm?
6 Tetromino cover

Is it possible to cover a chessboard with 15 T-tetrominoes and a square one?

7 Infinite array

You are given an infinite array $A[..]$ in which the first $n$ cells contain integers in sorted order and the rest of the cells are filled with $\infty$. You are not given the value of $n$. Describe an algorithm that takes an integer $x$ as input and finds a position in the array containing $x$, if such a position exists, in $O(\log n)$ time. (Assume that the implementation of the array data type in your programming language returns the error message $\infty$ whenever elements $A[i]$ with $i > n$ are accessed.)