# CS 125 Algorithms & Complexity — Fall 2016
# Problem Set 9

Due: 11:59pm, Friday, November 11th

See homework submission instructions at `http://seas.harvard.edu/~cs125/fall16/schedule.htm`

## Problem 1

Consider the Tiling from class, in which the input is a finite collection of distinct tile types whose edges are colored, and the question is whether we can tile the entire plane using those squares (such that squares sharing an edge in the tiling must have the same color on that edge). In class we showed that TilingFirstQuadrant is undecidable, which was a variant of the problem in which we only had to tile the upper right quadrant, and we are also told which square should be placed in the bottom left corner. (See Lecture 17, Section 17.4)

(a) (3 points) Consider the problem Tiling$_k$, which is exactly like Tiling except that the square types in the input can only have edge colors in the set $\{1, \ldots, k\}$. Prove or disprove: there exists a fixed $k$ such that Tiling$_k$ is undecidable.
If it is decidable, what is the best running time you can achieve by an algorithm to decide it, say in the word RAM model?

(b) (7 points) Consider now the problem Tiling$_k$Completion, where a finite partial tiling of the plane is given as input together with a finite set of tile types that may be used, and the problem is to decide whether the partial tiling can be completed to a full tiling of the plane. Again, all tile types have edge colors in $\{1, \ldots, k\}$. Prove that there is a fixed constant $k$ such that Tiling$_k$Completion is undecidable. **Hint:** reduce from a suitable variant of the Halting Problem.

## Problem 2

In class we analyzed QuickSelect using the notion of "good" recursive calls: a recursive call was good if it reduced the number of working elements by a factor of at least 3/4. Instead, we could adopt an analysis for QuickSelect similar to that for QuickSort. Suppose we call QuickSelect$(A, k)$, to find the $k$th smallest element in an array $A$ of size $n$. Assume the elements of $A$ are distinct. Note the running time of QuickSelect is proportional to the number of comparisons. Thus if we let $X_{i,j}$ be a random variable which is 1 if $i$th smallest item and $j$th smallest item are ever compared throughout the execution of QuickSelect$(A, k)$, then the running time is proportional to $\sum_{i<j} X_{i,j}$.

(a) (4 points) For $i < j$, give an exact expression for $\mathbb{E}\, X_{i,j}$ in terms of $i, j, k, n$. You may need to employ case analysis.

(b) (6 points) Using (a), show that $\mathbb{E} \sum_{i<j} X_{i,j} = O(n)$.

## Problem 3

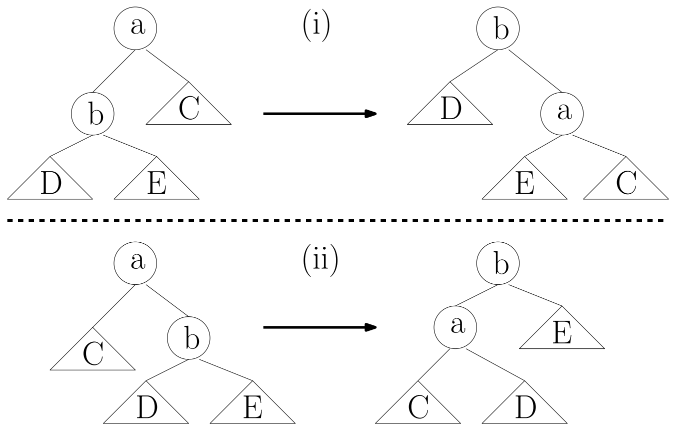

Figure 1: BST rotation: in (i), $b$ is rotated upward as a left child. In (ii), $b$ is rotated upward as a right child. A circle denotes a single vertex, and a triangle denotes a subtree.

Consider the following implementation of a binary search tree (BST). When searching, we search based on key as in a normal BST. When inserting an element with key $k$, we assign the element a uniformly random ID in $[0, 1]$. We first insert the element into the BST based on its key as normal. We then *rotate* the node it lands in upward to preserve the invariant that, when looking at node IDs instead of node keys, the tree should be a min heap (see Figure 1 for a depiction of rotation).

Suppose the keys in the BST at some point are $k_1 < k_2 < \ldots < k_n$. When searching for key $k_r$, note that we touch the node with key $k_i$ if and only if it is an ancestor of the node with $k_r$ in the BST. Using this fact, show that the expected time to perform a query is $O(\log n)$. Also show that the expected time to insert a new key into the data structure is also $O(\log n)$.

## Problem 4

We have $n$ wooden blocks and $n$ baskets. For each block, we throw it into a uniformly random basket, and we do this independently for each block. Show that with probability at least 99%, no basket receives more than $O(\log n / \log\log n)$ blocks. You can use the following two facts without proof:

(1) for all $1 \leq k \leq n$, $(n/k)^k \leq \binom{n}{k} \leq (en/k)^k$, and

(2) the "union bound": for any set of probabilistic events $\{\mathcal{E}_i\}_{i=1}^t$,

$$\mathbb{P}(\mathcal{E}_1 \text{ or } \mathcal{E}_2 \text{ or } \ldots \text{ or } \mathcal{E}_t) \leq \sum_{i=1}^{t} \mathbb{P}(\mathcal{E}_i).$$