# CS 473: Algorithms, Fall 2010
# HW 3 (due Tuesday, September 21st)

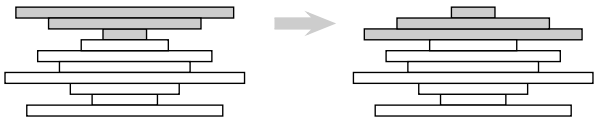
This homework contains four problems. **Read the instructions for submitting homework on the course webpage**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper; the sheets for each problem should be stapled together. Write your name and netid on each sheet.

**Collaboration Policy:** For this home work, Problems 1-3 can be worked in groups of up to 3 students each.

**Problem 0 should be answered in Compass as part of the assessment HW3-Online and should be done individually.**

0. (25 pts) HW3-Online.

1. (25 pts) Suppose we have a stack of $n$ pancakes of different sizes. We want to sort the pancakes so that the smaller pancakes are on top of the larger pancakes. The only operation we can perform is a *flip* - insert a spatula under the top $k$ pancakes, for some $k$ between 1 and $n$, and flip them all over.



Flipping the top three pancakes

   - (15 pts) Describe an algorithm to sort an arbitrary stack of $n$ pancakes and give a bound on the number of flips that the algorithm makes. Assume that the pancake information is given to you in the form of an $n$ element array $A$. $A[i]$ is a number between 1 and $n$ and $A[i] = j$ means that the $j$'th smallest pancake is in position $i$ from the bottom; in other words $A[1]$ is the size of the bottom most pancake (relative to the others) and $A[n]$ is the size of the top pancake. Assume you have the operation Flip($k$) which will flip the top $k$ pancakes. Note that you are only interested in minimizing the number of flips.
   - (10 pts) Suppose one side of each pancake is burned. Describe an algorithm that sorts the pancakes with the additional condition that the burned side of each pancake is on the bottom. Again, give a bound on the number of flips. In addition to $A$, assume that you have an array $B$ that gives information on which side of the pancakes are burned; $B[i] = 0$ means that the bottom side of the pancake at the $i$'th position is burned and $B[i] = 1$ means the top side is burned. For simplicity, assume that whenever Flip($k$) is done on $A$, the array $B$ is automatically updated to reflect the information on the current pancakes in $A$.

2. (25 pts) Let $A$ be an array of $n$ *distinct* elements such that each element in $A$ is at most $k$ positions away from its position in the sorted order; here $k$ is some integer such that $2 \leq k \leq n-1$. Thus the rank of an element at location $A[i]$ is at least $i-k$ and at most $i+k$; here the rank of an element is its position in the sorted array. In other words the array $A$ is close to being sorted if $k$ is small. Can we take advantage of this and sort faster? Design an $O(n \log k)$ time algorithm to sort elements of $A$. An $O(nk)$ algorithm will get you at most 13 points.

3. (25 pts) Let $A$ be an array of $n$ integers. There is no direct way to access the numbers in $A$; the only way to compare them is via a procedure **AreEqual**$(i, j)$ that given two indices $i$ and $j$ between 1 and $n$ will return "Yes" if $A[i] = A]j]$ and "No" otherwise. Describe an algorithm that checks whether there is a number $x$ that is in more than $n/2$ locations in $A$. Your algorithm should run in $O(n \log n)$ time. Extra credit of 10 pts for an algorithm that runs in $O(n)$ time.