Solutions for Homework 2

1. Use the Master Theorem to find the asymptotic solution for the following recurrence: $T(n) = 5T\left(\frac{n}{2}\right) + n^3$.

Solution: We have $a = 5$, $b = 2$, $f(n) = n^3/n^{\log_2 5}$, we get Case 3, and thus $T(n) = \Theta(n^3)$. Note that the regularity condition is satisfied as $5(n/2)^3 = 5n^3/8 \leq cn^3$ for $c = 5/8$. (20 points)

2. Suppose that we are given a sorted array of distinct integers $A[1, \ldots, n]$ and we want to decide whether there is an index $i$ for which $A[i] = i$.

(a) Describe a divide-and-conquer algorithm that solves this problem.

(b) Use the Master Theorem to estimate the running time of the algorithm you described in part (a). Your algorithm should run in $O(\log n)$ time.

Solution:

(a) Algorithm: If the array has just one integer, then we check whether $A[1] = 1$ with one comparison. Otherwise divide the list into two parts, the first half and the second half, as equally as possible. Consider the largest element $A[m]$ of the left half. We compare $A[m]$ with $m$.

- If $A[m] = m$, then the answer is yes and we are done.
- If $A[m] > m$, then we can throw away the right half and continue recursively in the left half. Indeed, then for every integer $k \geq 0$ using the fact that the integers are distinct and sorted $A[m + k] \geq A[m] + k > m + k$. 


• If $A[m] < m$, then we can throw away the left half and continue recursively in the right half. Indeed, then for every integer $k \geq 0$ using the fact that the integers are distinct and sorted

$$A[m - k] \leq A[m] - k < m - k.$$ 

Thus for the number of comparisons we get the following recursion:

$$T(n) = T\left(\frac{n}{2}\right) + 1, \quad T(1) = 1.$$

(b) By the Master Theorem, since we have $a = 1$, $b = 2$,

$$f(n) = 1 = \Theta(n^{\log_2 1}) = \Theta(n^0) = \Theta(1),$$

we get Case 2, and thus $T(n) = \Theta(\log n)$, as desired. (20 points)

3. Use indicator random variables to find the expected number of balls that fall into the first bin when $m$ balls are distributed into $n$ bins uniformly at random.

**Solution:** Let

$$X_i = \begin{cases} 
1 & \text{if the } i\text{th ball falls into the first bin} \\
0 & \text{otherwise} 
\end{cases}$$

Then by the linearity of expectation

$$E(X) = E(\sum_{i=1}^{m} X_i) = \sum_{i=1}^{m} E(X_i) = \sum_{i=1}^{m} \frac{1}{n} = \frac{m}{n}.$$ 

(20 points)

4. A sorting algorithm is called ***stable*** if equal elements are in the same relative order in the sorted sequence as in the original sequence. Describe an easy way to make any unstable sorting algorithm stable.

**Solution:** Any unstable sorting algorithm can be made stable by the following trick: Instead of sorting the given sequence $(a_1, a_2, \ldots, a_n)$, sort the sequence

$$((a_1, 1), (a_2, 2), \ldots, (a_n, n))$$

of pairs of elements and their indices in the original sequence. When comparing two pairs, we have $(a_i, i) < (a_j, j)$ if and only if $a_i < a_j$, or $a_i = a_j$ and $i < j$. (20 points)
5. In our MERGE-SORT algorithm we merged two sorted lists into one sorted list in $O(n)$ time. Describe an $O(n \log k)$-time algorithm using a min-priority queue to merge $k$ sorted lists into one sorted list, where $n$ is the total number of elements in all the input lists.

**Solution:** First we remove the smallest element from each sorted list and we build a min-priority queue (using a min-heap) out of these elements in $O(k)$ time. Then we repeat the following steps: we extract the minimum from the min-priority queue (in $O(\log k)$ time) and this will be the next element in the sorted order. From the original sorted list where this element came from we remove the next smallest element (if it exists) and insert it to the min-priority queue (in $O(\log k)$ time). We are done when the queue becomes empty and at this point we have all the numbers in our sorted list. The total running time is $O(k + n \log k) = O(n \log k)$. (20 points)