Solutions for the Practice Midterm Exam

These problems are sample problems for the midterm exam, so you may expect similar problems in the midterm. Do not hand in your solutions. The midterm exam is a closed book exam, but you may use one sheet of paper (written on both sides) with notes on it. Each problem is worth 20 points.

1. Use the Master Theorem to find the asymptotic solutions for the following recurrences:

   (a) \( T(n) = 7T\left(\frac{n}{2}\right) + n^2 \),
   (b) \( T(n) = T\left(\frac{n}{2}\right) + 1 \),
   (c) \( T(n) = 4T\left(\frac{n}{2}\right) + n^3 \).

Solution:

(a) We have \( a = 7, b = 2, \)
\[
f(n)/n^{\log_2 7} = n^2/n^{\log_2 7} = n^{2-\log_2 7} = O(n^{-0.8}),
\]
we get Case 1, and thus \( T(n) = \Theta(n^{\log_2 7}) \).

(b) We have \( a = 1, b = 2, \)
\[
f(n) = \Theta(1) = \Theta(n^{\log_2 1}) = \Theta(1),
\]
we get Case 2, and thus \( T(n) = \Theta(\log_2 n) \).

(c) We have \( a = 4, b = 2, \)
\[
f(n)/n^{\log_2 4} = n^3/n^2 = n,
\]
we get Case 3, and thus \( T(n) = \Theta(n^3) \). Note that the regularity condition is satisfied as \( 4(n/2)^3 = n^3/2 \leq cn^3 \) for \( c = 1/2 \).
2. Use indicator random variables to find the expected value of the number of fixed elements (elements left in the same position) in a random permutation of \( n \) elements.

**Solution:** Let

\[
X_i = \begin{cases} 
1 & \text{if the } i\text{th element is a fixed element} \\
0 & \text{otherwise}
\end{cases}
\]

Then by the linearity of expectation,

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

3. We have two input arrays, an array \( A \) with \( m \) elements and an array \( B \) with \( n \) elements, where \( m \leq n \). There may be duplicate elements. We want to decide if every element of \( B \) is an element of \( A \). Describe an algorithm to solve this problem in \( O(n \log m) \) worst-case time.

**Solution:** First we sort \( A \) by MERGESORT (in \( O(m \log m) \) time). Then for each element of \( B \) we do a binary search in the sorted list of \( A \) (in \( O(n \log m) \) time). The total worst-case running time is \( O((m + n) \log m) = O(n \log m) \).

4. We have an input array \( A \) with \( n \) (\( n \geq 2 \)) numbers.

(a) Describe a \( O(n) \) worst-case time algorithm to find two elements \( x, y \in A \) such that \( |x - y| \geq |u - v| \) for all \( u, v \in A \).

(b) Describe a \( O(n \log n) \) worst-case time algorithm to find two elements \( x, y \in A \) such that \( |x - y| \leq |u - v| \) for all \( u, v \in A \).

**Solution:** (a) For this we have to find the minimum and the maximum simultaneously which we can do in \( 3\lfloor n/2 \rfloor = O(n) \) time.

(b) For this we sort the numbers first, then \( x \) and \( y \) must be consecutive elements in the sorted order. We go through the sorted list and we find the smallest difference between two neighboring elements.

5. The Fibonacci numbers are defined by the recurrence:

\[
F_0 = 0, F_1 = 1, F_i = F_{i-1} + F_{i-2} \text{ for } i \geq 2.
\]
Give an $O(n)$-time dynamic programming algorithm to compute the $n$th Fibonacci number. Draw the subproblem graph. How many vertices and edges are in the graph?

**Solution:** Here is the algorithm:

```plaintext
FIBONACCI(n)
let fib[0..n] be a new array
fib[0] = 0, fib[1] = 1
for i = 2 to n
  fib[i] = fib[i - 1] + fib[i - 2]
return fib[n]
```

This directly implements the recurrence relation of the Fibonacci numbers. Each number in the sequence is computed from the previous two numbers in the sequence. The running time is clearly $O(n)$. The subproblem graph consists of $n + 1$ vertices, $v_0, v_1, \ldots, v_n$. For $i = 2, \ldots, n$, vertex $v_i$ has two leaving edges: to vertices $v_{i-1}$ and $v_{i-2}$. No edge leaves vertices $v_0$ and $v_1$. Thus, the subproblem graph has $2(n - 1)$ edges.