Exercise 10.1  Longest Path in a directed, acyclic graph.

While the problem of finding the shortest path in a directed graph can be solved efficiently, it is not so easy to find the longest path. However, the longest path can be found efficiently if the graph is acyclic. Develop an algorithm that finds the longest path in a directed acyclic graph \( G = (V, E) \) with \( n \) vertices and \( m \) edges as fast as possible.

Exercise 10.2  Fibonacci Heaps.

a) Can a Fibonacci heap degenerate to a structure that has exactly one node in the root list, with a linear list appended to it? Describe how this structure can occur, or prove that it is not possible.

b) Why is it impossible that all the heap operations (INSERT, EXTRACT-MIN, DECREASE-KEY) have an amortized running time of \( O(1) \) if we allow only key comparisons (but no computations)?

Exercise 10.3  Enumerating Palindromes.

A palindrome is a word whose meaning may be interpreted the same way in either forward or reverse direction, e.g. the word RACECAR. Formally, a palindrome is a sequence \( (a_1, \ldots, a_n) \) where either \( n = 1 \), or \( a_1 = a_n \) and \( (a_2, \ldots, a_{n-1}) \) is a palindrome (for \( n = 2 \) we only require \( a_1 = a_2 \)). Let \( A[1..n] \) be an array storing a string of length \( n \). A subarray \( A[i..j] \), \( 1 \leq i \leq j \leq n \), is called palindrome in \( A \) if \( (A[i], \ldots, A[j]) \) is a palindrome.

Example: The array \([L, A, R, A]\) contains the palindromes \( A, R, L \) and \( ARA \) (the palindrome \( A \) occurs twice). The array \([A, N, N, A]\) contains the palindromes \( A, N, NN \) and \( ANNA \) (the palindromes \( A \) and \( N \) occur twice).

a) Let \( A \) be an array containing a string of length \( n \). Describe a dynamic programming algorithm that outputs all pairs \((i, j)\) where \( (A[i], \ldots, A[j]) \) is a palindrome. Provide also the running time of your solution.

Example: For the input \([L, A, R, A]\), the output consists of the pairs \((1,1), (2,2), (3,3), (4,4), (2,4)\). For the input \([A, N, N, A]\), the output consists of the pairs \((1,1), (2,2), (3,3), (4,4), (2,3), (1,4)\). In the output, no special order of the pairs is required.

Please turn over.
Hint: Notice that the exercise can easily be solved without dynamic programming by trivial enumeration of all palindromes in time $O(n^3)$. We search for a more efficient algorithm.

b) Suppose that the algorithm of a) computed the DP table already. Describe in detail, how a longest palindrome in $A$ can be extracted from the DP table. Provide also the required running time.

Exercise 10.4 Programming Exercise: Longest Palindrome.

In this exercise we are going to implement the algorithm developed in Exercise 10.3 part a. Your program should only output the length $l$ of the longest palindrome, as well as all pairs $(i, j)$, such that $(A[i], \ldots, A[j])$ is a palindrome of length $l$. The pairs should be sorted in ascending order with respect to the starting position $i$.

Input The first line contains only the number $t$ of test instances. After that, we have one line per test instance containing the word $A$ with length $1 \leq n \leq 10000$.

Output For each test instance we output only one line. This line contains the length $l$ of the longest palindrome and the pairs $(i, j)$ for all palindroms of length $l$ in ascending order with respect to the starting position.

Example

Input:

```
2
OTTOSNEFFEN
SARASKAJAK
```

Output:

```
6 6 11
5 1 5 6 10
```