Problem

Order the following functions in increasing order of order-of-magnitude growth. If two functions have the same o.m. growth, indicate that.

\[2^n, 2\sqrt{n}, 2n/2, \sqrt[3]{2n}, n, n^3, (n/2)^3, \sqrt{n}, (\sqrt{n})^3\].

Problem

Write a function \texttt{LargestWindow(A,M)} that takes as arguments an sorted integer array \texttt{A} of size \texttt{N} and an integer \texttt{M} and returns starting and ending index of the largest segment of \texttt{A} in which the top and bottom element differ by at most \texttt{M}. For example if \texttt{A} = \[1,15,21,25,29,35\] and \texttt{M} = 6 then \texttt{LargestWindow(A,M)} should return \langle 3, 5 \rangle since 3,4,5 is a segment of length 3, and \texttt{A[5]} - \texttt{A[3]} = 5 \leq M. Your function should have running time \texttt{O(N)}. (Hint: Use a two-fingered method.)

Problem

A. Let \texttt{A} be an \texttt{N} \times \texttt{N} array with 1-based indexing and let \texttt{I,J} be integers between 1 and \texttt{N}. The function \texttt{Minor(A,R,C)} returns a copy of \texttt{A} with the \texttt{R}th row and \texttt{C}th column deleted. For example

\[
\begin{bmatrix}
6 & 2 & 1 & 0 \\
2 & 5 & 7 & 9 \\
3 & 8 & 1 & 4 \\
7 & 2 & 3 & 8 \\
\end{bmatrix}
\]

then \texttt{Minor(A,2,4) =}

\[
\begin{bmatrix}
6 & 2 & 1 \\
3 & 8 & 1 \\
7 & 2 & 3 \\
\end{bmatrix}
\]

The code for \texttt{Minor} can be written as follows:

\[
\begin{bcode}
\text{Minor(A[1..N,1..N] ,R,C) \{ }
\text{B = a new (N-1) x (N-1) array}
\text{IB = 1}
\text{for (IA = 1 to N)}
\text{\quad if (IA != R) \{ }
\text{\quad JB = 1;}
\text{\quad for (JA = 1 to N)}
\text{\quad\quad if (JA != C) \{ }
\text{\quad\quad B[IB,JB] = A[IA,JA]}
\text{\quad\quad JB++;}
\text{\quad\}}
\text{\quad IB++;}
\text{\}}
\text{return B;}
\text{\}}
\end{bcode}
\]

What is the order of magnitude running time of \texttt{Minor} as a function of \texttt{N}?

B. The function \texttt{Determinant(A)} takes as argument an \texttt{N} \times \texttt{N} array \texttt{A} and is defined as follows:
function Determinant(A[1..N,1..N]) {
    if (N == 1) return A[1,1];
    sum = 0;
    sign = 1;
    for (C = 1 to N) {
        B = Minor(A,1,C);
        sum = sum + sign*Determinant(B);
        sign = -sign;
    }
    return sum;
}

Write the recurrence equation for the running time of Determinant as a function of N.

Problem

Heaps are standardly implemented as arrays in which the children of node H[i] are H[2i] and H[2i+1] (using 1-based indexing). In principle, the same technique can be used to implement binary search trees. Assume for this problem that the values stored in the tree are all positive integers, so −1 can be used as a null value.

A. Show the array that would correspond to the tree below.

B. Assume that A is a binary search tree implemented as an array, as described above. Write an iterative algorithm Search(A,X) that searches for value X in A. It should return the index of X in A if X is there, or −1 if it is not. The algorithm should run in time proportional to the height of the tree (not to the size of the array.)

C. Why is this implementation not used in practice?