1a. (10 pts) Explain why an $O(n^4)$ algorithm would be preferable over an $O(2^n)$ algorithm when one does not have any idea about the expected input problem instance-size $n$.

1b. (10 pts) How will the decision be affected when there is some idea on how large the problem size ($n$) would be as input to the chosen algorithm?
2. (20 pts) The **MAXSUM** problem over a sequence of positive and negative numbers is to find a subsequence that produces the largest sum. For instance, over a sequence \((3, -1, 9, -5, 2)\), the answer is 11 for the subsequence \((3, -1, 9)\). The following iterative algorithm calculates the maximum subsequence sum.

Algorithm MaxSum1(an array of numbers a, of length n)

MaxSum=0;
For (i=0; i<n; i=i+1) {
    For (j=i; j<n; j=j+1) {
        thisSum=0;
        For (k=i; k<=j; k=k+1) {
            thisSum=thisSum+a[k];
        }
        If (thisSum>MaxSum) MaxSum=thisSum;
    }
} return MaxSum;
End Algorithm.

Analyze the time-complexity of the algorithm MaxSum1.
3. (20 pts) For the MAXSUM problem a recurrence equation can calculate the result:

\[
\text{MaxSum}[i, j] = \begin{cases} 
\max\{\text{MaxSum}[i - 1, j] + a[i], \text{MaxSum}[i, j - 1] + a[j]\} & \text{for } i < j \\
a[i] & \text{for } i = j
\end{cases}
\]

where MaxSum[i, j] is the maximum subsequence sum over the subsequence a[i..j]. Of course, the result would be obtained for the whole sequence in MaxSum[1, n].

Design a dynamic programming algorithm for this optimization problem. Just outlining the algorithm with an example run (over the above problem instance in the question 2) will suffice as an answer. Hint: compute the \(i \times j\)-matrix diagonally, first for \(j=i\), then for \(j=i+1\), then for \(j=i+2\), and so on.
4. (20 pts) A sparse directed graph $G = (V, E)$ is represented as an adjacency list, where $V$ is a set of $n$ nodes, and $E$ is the set of $e$ edges, each of which is an ordered pair of nodes.

Carefully analyze the time-complexity of the following algorithm-fragment. Express your answer in terms of $|e|$ and $|n|$.

```plaintext
For each node N1 in V do {
    Print N1;
    For each node N2 adjacent to N1 do { Print N2; }
}
```
5. (20 pts) Write a recursive divide-and-conquer algorithm for computing the sum over a sequence of numbers. Analyze its time-complexity.