1. The $n$-th Catalan number $C_n$ is defined by the following recurrence relation

$$C_0 = 1, C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}, n \geq 0.$$ 

Give a dynamic programming algorithm based on this recurrence to compute the $n$-th Catalan number. Draw the subproblem graph. How many vertices and edges are in the graph? What is the running time of your algorithm?

**Solution:** Here is the algorithm:

```
catalan(n)
let C[0..n] be a new array
C[0] = 1
for i = 1 to n
    C[i] = \sum_{j=0}^{i-1} C[j]C[i - 1 - j]
return C[n]
```

Each number in the sequence is computed from all the previous numbers in the sequence. The running time is clearly $O(n^2)$. The subproblem graph consists of $n + 1$ vertices, $v_0, v_1, \ldots, v_n$. For $i = 1, \ldots, n$, vertex $v_i$ has edges to all vertices $v_j$ with $j < i$. No edge leaves vertex $v_0$. Thus, the subproblem graph has $n(n + 1)/2$ edges. (20 points)

2. Not every greedy strategy works. Show that if in the activity-selection problem we always select the activity of least duration (another greedy strategy) from among those that are compatible with the previously selected activities we might not get an optimal solution.

**Solution:** Assume we have the following three activities: $a_1$ with $s_1 = 0, f_1 = 3$, $a_2$ with $s_2 = 2$ and $f_2 = 4$ and $a_3$ with $s_3 = 3$ and $f_3 = 6$. The
suggested greedy approach selects just \( \{a_2\} \), but the optimal solution selects \( \{a_1, a_3\} \). (20 points)

3. Suppose we perform a sequence of \( n \) operations on a data structure in which the \( i \)-th operation costs \( i \) if \( i \) is an exact power of 2, and 1 otherwise. Use the accounting method to determine the amortized cost per operation and use this to get an upper bound on the actual cost of the sequence of \( n \) operations.

**Solution:** Let \( c_i \) = actual cost of the \( i \)-th operation, i.e.

\[
c_i = \begin{cases} 
i & \text{if } i \text{ is an exact power of 2}, \\
1 & \text{otherwise}.
\end{cases}
\]

Then let \( \overline{c_i} \) = amortized cost of the \( i \)-th operation = 3 for every \( 1 \leq i \leq n \). Thus we have \( \sum_{i=1}^{n} \overline{c_i} = 3n \). Furthermore, we have

\[
\sum_{i=1}^{n} c_i \leq n + \sum_{j=0}^{\lfloor \log n \rfloor} 2^j \leq n + (2n - 1) < 3n,
\]

thus indeed \( \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \overline{c_i} = 3n \). (20 points)

4. Consider the same set-up as in the previous problem but now use the potential method to determine the amortized cost per operation and use this to get an upper bound on the actual cost of the sequence of \( n \) operations.

**Solution:** We define the potential function \( \Phi \) as twice the distance of \( i \) from the largest power of 2 that is at most \( i \), i.e.

\[
\Phi(D_i) = 2 \left( i - 2^{\lfloor \log_2 i \rfloor} \right).
\]

Then we have \( \Phi(D_i) \geq 0 = \Phi(D_0) \), and thus \( \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \overline{c_i} \). We just have to estimate \( \overline{c_i} \). If \( i \neq 2^k \), then

\[
\overline{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2 = 3.
\]

If \( i = 2^k \), then

\[
\overline{c_i} = c_i + \Phi(D_i) - \Phi(D_{i-1}) = i + 0 - 2(2^k - 1 - 2^{k-1}) = 2^k - 2^k + 2 \leq 3.
\]

Thus \( \sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \overline{c_i} \leq 3n \). (20 points)
5. Describe an algorithm for finding a spanning tree with minimal weight containing a specified acyclic (cycle-free) set of edges $A$ in a connected weighted undirected graph.

Solution: We just run Kruskal’s algorithm, but instead of starting with an empty set of edges, we start with the edge set $A$ given. (20 points)