hw4_pr1.py > ...

```python
# Subproblem
# Determine the maximum number of rounds the player can win.

# Recurrence
# dp[i] = 1 + max(dp[i+1], dp[i+2])

# Topological Order
# Starting from the last round and moving towards the first round.

# Base
# The base case is when the last round is reached and there are fewer than 5 cards
# remaining in the deck. In this case, the maximum number of rounds the player can
# win is 0, as a new round cannot be played.

# Original Problem
# Find the maximum number of rounds the player can win in the game 21,
# given a deck of n cards with a known order.

# Time
# O(n)
```

hw4_pr2.py > ...

```python
# Subproblem
# For a given house i, find the minimum cost to paint the first i houses.

# Recurrence
# dp[i][0] = costs[i][0] + min(dp[i-1][1], dp[i-1][2])
# dp[i][1] = costs[i][1] + min(dp[i-1][0], dp[i-1][2])
# dp[i][2] = costs[i][2] + min(dp[i-1][0], dp[i-1][1])

# Topological Order
# From left to right

# Base
# The base case is when there is only one house.
# In this case, the minimum cost is the cost of painting that house with each color.

# Original Probelm
# minCostTD(costs)

# Time
# O(n)

```

hw4_pr3.py > ...

```
# Subproblem
# find the size of the largest square that can be formed at that position,
# considering only α's

# Recurrence
# dp[i][j] = min(dp[i-1][j-1], dp[i-1][j], dp[i][j-1]) + 1, if matrix[i][j] == 'α'
# dp[i][j] = 0, if matrix[i][j] == 'ω'

# Topological Order
# The topological order for solving the subproblems is from left to right and top
# to bottom, iterating through the positions in the matrix row by row

# Base
# The base case is when the position (i, j) is at the first row or the first column.
# In this case, the size of the largest square at that position will be 1 if it
# contains α and 0 if it contains ω.

# Original Probelm
# largestAlphaArea(costs)

# Time
# O(n * m)
```