13.5
How to come up with dynamic programming algorithm: summary

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

Dynamic Programming

1. Find a "smart" recursion for the problem in which the number of distinct subproblems is small; polynomial in the original problem size.
2. Estimate the number of subproblems, the time to evaluate each subproblem and the space needed to store the value.
3. This gives an upper bound on the total running time if we use automatic/explicit memoization.
4. Come up with an explicit memoization algorithm for the problem.
5. Eliminate recursion and find an iterative algorithm.
6. ...need to find the right way or order the subproblems evaluation. Th is leads to an a dynamic programming algorithm.
7. Optimize the resulting algorithm further
8. ...
9. Get rich!

THE END

...

(for now)