Week 5. Problem set (solutions by Danil Elgin)

In a fictional multiplayer video game "Dragons and Pirates", each player can select one of tree character classes: a KNIGHT, a PIRATE, or a COOK. A team of 5 players wants to optimize their chances at the game by choosing the right combination of classes for each player in the team. When multiple players take the same class, their powers overlap, leading to an uneven increase in performance. For example, in the game multiple Knights cannot attack an enemy simultaneously, so overall performance metric for three Knights is less than three times the performance of a single Knight.

The table below gives the estimated performance for each character class for each possible number of players of the same class. Total performance of a team is the sum of performances for each class.

1. Describe a general algorithm that would find an optimal class assignments for any number P of players, any number C of classes, and table `E` with estimates:

   (a) Summarize the idea for a naïve recursive algorithm.

   Answer: *Within the recursion function, we call the same function C times to assign a player the role of a knight, cook, or pirate, but we must reduce the number of places (P) by 1. We must do this until the player's place is zero. This would mean that we have already allocated all possible locations, we would need to return the sum of the estimates for the combination of location assignments.*

   (b) Clearly identify *overlapping subproblems* [CLRS, §14.3] (provide an explicit example for this problem).

   Answer: *When we call a recursive function, we have an overlap because we consider all distributions, not taking into account that this combination can already be repeated only in a different order of function calls. If we take a certain class a on the i-th function call and class b on the (i + 1)-th, then this will be the same as we take class b on the i-th function call and class a on the (i + 1)-th. Example: If we take 3 knights first and then 2 pirates, it will be the same if we take 1 knight, then 2 pirates, then 2 knights, because the combination will be the same.*

   (c) Write down pseudocode for the dynamic programming [CLRS, §14] algorithm that solves the problem (top-down or bottom-up). It is enough to compute the optimal performance without keeping track of the exact class assignments. **Answer:**

Algorithm 1 Dynamic Algorithm

```
Input: C, P, &E

Create an array Answer of size C × (P + 1), initialized to zero
for i = 0 to P (inclusive) do
    Answer[0][i] ← E[0][i]
end for
for i = 1 to C (exclusive) do
    for j = 0 to P (inclusive) do
        for z = 0 to j (inclusive) do
            Answer[i][j] ← max(Answer[i][j], Answer[i − 1][z] + E[i][j − z])
        end for
    end for
end for
return Answer[C − 1][P]
```

2. Provide asymptotic worst-case time complexity with justification

   (a) for the naïve recursive algorithm

   Answer: *The asymptotic worst-case time complexity for naïve recursive algorithm is $\Theta(c^p)$. In each call to the recursive function, we will call this function C more times to iterate through all possible classes for the given position we are in until we reach the last position.*

There are P positions in total, and C classes, which means that there will be c^p total calls to the recursive function. And all operations in the function itself are performed in constant time.

(b) for the dynamic programming algorithm

Answer: *The asymptotic worst time complexity for dynamic programming algorithm $\Theta(c * p^2)$. As we could see in the pseudo-code, we go through the list of "for" in all possible classes, inside this class we go through the list of "for" in possible places (number of people), we also go through another list of "for" up to the number of places to sort through all possible combinations. We do this in nested loops because we need to understand what is the best combination we can put together having a certain number of classes, and then based on this information we perform the next iteration and so on to the end. And as we can see, since we have 3 nested loops that are executed (c, p, in the worst case p) and other operations cost less, therefore the asymptotic execution time is $\Theta(c * p^2)$*

3. Apply the dynamic programming algorithm to an instance of the problem below with 3 classes and 6 players:

(a) What is the best possible performance? What combination of classes gives that performance?

Answer: *The best possible perfomance is 70, and the combination that ensures maximum performance is (1 knight, 2 pirates, 3 cooks).*

(b) Provide the table with solutions used in the dynamic programming algorithm for subproblems that are computed in the algorithm.

Number of players	0	1	2	3	4	5	6
KNIGHT	0	12	23	33	42	55	62
PIRATE	0	13	24	33	40	45	50
COOK	0	11	22	34	43	52	61

Answer:

Number of players	*0*	*1*	*2*	*3*	*4*	*5*	*6*
KNIGHT	*0*	*12*	*23*	*33*	*42*	*55*	*62*
PIRATE	*0*	*13*	*25*	*36*	*47*	*57*	*68*
COOK	*0*	*13*	*25*	*36*	*47*	*59*	*70*

References

[CLRS] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2022. *Introduction to algorithms, Fourth Edition*. MIT press.