ECE374 Assignment 6

Due 03/27/2023

Group & netid

Chen Si   *chensi3*

Jie Wang   *jiew5*

Shitian Yang   *sy39*

Problem 2

2. Suppose you are given n poker chips stacked in two stacks, where the edges of all chips can be seen. Each chip is one of three colors. A turn consists of choosing a color and removing all chips of that color from the tops of the stacks. The goal is to minimize the number of turns until the chips are gone. For example, consider the stacks (RRGG,GBBB). Playing red, green, and then blue suffices to clear the stacks in three moves. Give an $O(n^2)$ dynamic programming algorithm to find the best strategy for a given pair of chip piles.

Solution:

Intuition:

(1) Marking the length of two stacks as a and b respectively, we could construct a (a+1)×(b+1) table with each element T[i, j] represents the minimum number of turns to get the chips gone.

(2) Recurrence Cases
Base Case
When both stacks are gone, we could mark T[0, 0] = 0 as there are no turns needed.

General Case
For the stacks S1 and S2, denote the number of remaining chips after removing all red chips on the top as R1, R2, respectively. Similarly, we could denote the number of remaining chips after removing all blue chips on the top as B1, B2, and the number of remaining chips after removing all green chips on the top as G1, G2. Therefore, we could have the following recurrence relation that T[i, j] = 1 + min (T[R1, R2], T[B1, B2], T[G1, G2]), representing that the minimum number of turns at [i, j] is 1 turn of removal plus the minimal number of turns to remove the rest chips.

(3) Therefore, we have the recurrence function as

$$T[i,j] = \begin{cases} 0, & i = 0 \text{ and } j = 0 \\ 1 + \min \begin{cases} T[R1, R2] \\ T[G1, G2], \\ T[B1, B2] \end{cases} & \text{otherwise} \end{cases}$$

The final result is in the T[a, b] of the table, which represents the minimum step to "disappear" a stack of a chips and a stack of b chips.

Therefore, the algorithm is:

```
BestStrategy(S1[1…a], S2[1…b]):
    T = table(a+1, b+1)

    for i ← 0 to a:
        for j ←0 to b:

            if (i==0 && j==0):
                T[i, j] = 0    // Base Case T[0,0]=0
            else:
                R1, R2 = NumRemain(S1, S2, Red)
                G1, G2 = NumRemain(S1, S2, Green)
                B1, B2 = NumRemain(S1, S2, Blue)

                T[i, j] = 1 + min(T[R1, R2], T[G1, G2], T[B1, B2])

    return T[a,b]

// Helper function that takes two stacks and one color
// and return the remaining number of chips after removing
// all 'color' chips at top
NumRemain(S1[1…a], S2[1…b], color):
    k1 = 0
    while (S1[k1+1] == color) && (k1+1 <= a):
        k1 ++

    k2 = 0
    while (S2[k2+1] == color) && (k2+1 <= b):
        k2 ++

    return a-k1, b-k2
```

Since we fill up a (a+1)×(b+1) table, with a and b proportional to the total number of chips n, and that the cost of operation at each element in the table is constant, we have an algorithm of

$O(ab) = O(pn \times (1-p)n) = O(p(1-p)n^2) = O(n^2)$. In the worst case, we have $a = \frac{n}{2}, b =$

$\frac{n}{2}, O(ab) = O\left(\frac{n^2}{4}\right) = O(n^2)$, the run time upper bound still holds.

Ps:

We count the minimum number of turns in the optimal strategy instead of the best strategy according to this Piazza post:

○ Resolved   ○ Unresolved   @298_f4 ⊖

Anonymous Mouse 3 days ago
What is the output? Should it be the strategy (the order we remove the chips) or just the number of turns?
helpful! | 0

    i Sung Woo Jeon 3 days ago
    Number of turns is enough.
    good comment | 0