COL351
Assignment 2

Mallika Prabhakar (2019CS50440)
Sayam Sethi (2019CS10399)

September 2021

Contents

1 Question 1

Question 1

Question. *Alice, Bob, and Charlie have decided to solve all exercises of the Algorithms Design book by Jon Kleinberg, Éva Tardos. There are a total of n chapters, $[1, \ldots, n]$, and for $i \in [1, n]$, x_i denotes the number of exercises in chapter i. It is given that the maximum number of questions in each chapter is bounded by the number of chapters in the book. Your task is to distribute the chapters among Alice, Bob, and Charlie so that each of them gets to solve nearly an equal number of questions.*
Device a polynomial time algorithm to partition $[1, \ldots, n]$ into three sets S_1, S_2, S_3 so that $\max\{\sum_{i \in S_1} x_i, \sum_{i \in S_2} x_i, \sum_{i \in S_3} x_i\}$ is minimized.

Solution. We propose a *Dynamic Programming* solution for this problem. The idea is to generate all possible combinations of S_1, S_2, S_3 and then find the best combination of out them. The naïve solution will have an exponential complexity ($O(3^n)$) and hence it needs to be modified so that it can be executed in polynomial time complexity.
We make the following observations to optimise our solution:

1. To find the optimal parititon of S, only the sum of each of S_1, S_2, S_3 matters
2. Order of picking elements for each set doesn't affect the solution
3. Fixing the sum of S_1 and S_2 uniquely identifies the sum of S_3

Using these observations we come up with the following DP table:

$$\begin{aligned} dp(i, s1, s2) &= dp(i-1, s1 - S[i], s2) \vee dp(i-1, s1, s2 - S[i]) \vee dp(i-1, s1, s2) \\ &\forall i \in \{1, \ldots, n\}; s1, s2 \in \{1, \ldots, sum(S)\} \\ dp(i, p, q) &= \bot, \ \forall i \in \{1, \ldots, n\}; p, q < 0 \\ dp(0, 0, 0) &= \top \end{aligned} \tag{1}$$

where $dp(i, s1, s2)$ is $\top$ if we can generate atleast one partition using the first i elements such that any two partitions have sums $s1$ and $s2$

Claim 1.1. *The dp table generated using the Equation 1 is correct, i.e., $dp(i, s1, s2) = \top$ iff there exists a partition using the first i elements with sums $s1, s2, sum(S[1:i]) - (s1 + s2)$*

Proof. We will prove the correctness of the claim by induction on i.
Base case: $i = 0$
$dp(0, s1, s2)$ is $\top$ only when $s1 = s2 = 0$ and $\bot$ otherwise. We know that we can generate only three empty sets using the first 0 elements and thus their sums will all be 0. Therefore the base case is true.
Induction step: Assume that the claim is true for $i - 1$. Consider $dp(i, s1, s2)$,
The i^{th} element will be present in exactly one of S_1, S_2, S_3. Therefore, we have three cases:

1. $S[i]$ is in S_1, then the sum of S_1 upto the first $i - 1$ elements will be $s1 - S[i]$ and the sum of the other two sets doesn't change

2. $S[i]$ is in S_2, then the sum of S_2 upto the first $i-1$ elements will be $s2 - S[i]$ and the sum of the other two sets doesn't change

3. $S[i]$ is in S_3, then the sum of S_3 upto the first $i-1$ elements will be $sum(S[1 : (i-1)]) - (s1 + s2)$ and the sum of the other two sets doesn't change

Thus, the only possibilities for valid partition sums using the first i elements are exactly those when we can generate atleast one of the above three parititons using the first $i-1$ elements. The transition equation given in Equation 1 exactly captures this. We have shown that for $dp(i, s1, s2)$ to be $\top$, atleast one of $dp(i-1, s1-S[i], s2), dp(i-1, s1, s2-S[i]), dp(i-1, s1, s2)$ must be $\top$. From induction, we know that the $(i-1)^{\text{th}}$ row of the table is true iff there exists a valid parititon. Therefore, if $dp(i, s1, s2) = \top$ then there exists a partition using the first i elements with sums $s1, s2, sum(S[1 : i]) - (s1 + s2)$. ($\implies$)

We still have to prove the converse, i.e., if there exists a partition using the first i elements with sums $s1, s2, sum(S[1 : i]) - (s1 + s2)$, then $dp(i, s1, s2) = \top$.
To prove this, we observe that the dp table considers all possible sums since $s1, s2$ iterate in the range $\{1, \ldots, sum(S)\}$. Therefore, if there exists a valid solution, the dp table considers it and is assigned $\top$. Else it is assigned $\bot$. ($\impliedby$) □

Now that we have proved the correctness of Equation 1, we present an algorithm for computing the same:

Algorithm 1 DP solution for partitioning

procedure PARTITION(S)
  $n \leftarrow size(S)$
  $s \leftarrow sum(S)$
  $dp \leftarrow$ table of size $(n+1) \times (s+1) \times (s+1)$ initialised with $\bot$
  $dp(0,0,0) \leftarrow \top$
  for i in $[1, n]$ **do**
    for $s1$ in $[0, s]$ **do**
      for $s2$ in $[0, s]$ **do**
        $dp(i, s1, s2) \leftarrow dp(i-1, s1, s2)$
        if $s1 \geq S[i]$ **then**
          $dp(i, s1, s2) \leftarrow dp(i, s1, s2) \vee dp(i-1, s1 - S[i], s2)$
        end if
        if $s2 \geq S[i]$ **then**
          $dp(i, s1, s2) \leftarrow dp(i, s1, s2) \vee dp(i-1, s1, s2 - S[i])$
        end if
      end for
    end for
  end for

  $bestPair \leftarrow (-1, -1)$
  $leastMax \leftarrow \infty$
  for all $(s1, s2) \in [0, s] \times [0, s]$ **do**
    if $\max(s1, s2, s - (s1 + s2)) < leastMax$ **then**
      $leastMax \leftarrow \max(s1, s2, s - (s1 + s2))$
      $bestPair \leftarrow (s1, s2)$
    end if
  end for
  return $backtrack(dp, bestPair)$ ▷ return the partition after backtracking on the DP table
end procedure

□

2 Question 2

2.1 2.1

2.1

Question. *insert question*

Solution. Proof is left as an exercise. □

2.2 2.2

2.2

Question. *insert question*

Solution. Proof is left as an exercise. □

2.3 2.3

2.3

Question. *insert question*

Solution. Proof is left as an exercise. □

3 Question 3

3.1 3.1

3.2 3.2

4 Question 4

4.1 4.1

4.1

Question. *insert question*

Solution. Proof is left as an exercise. □

4.2 4.2

4.2

Question. *insert question*

Solution. Proof is left as an exercise. □