Project 2 Report

COT4400

Dalton Splinter
Khoa Doan
Mohamed Suleum Salim Al Hamzy

1.) How you can break down a problem instance of aligning a set of n teeth (a1, a2, . . . , an) with a set of m teeth (b1, b2, . . . , bm) into one or more smaller instances? Your answer should include how you calculate the minimum height for the original problem based on the solution(s) to the subproblem(s).

- Aligning a set of teeth as outlined in the problem is simply the task of minimizing the max height of any single pair of teeth. The minimum height of the two interlocking extending teeth is the min height of the teeth in front of it and the min height of the teeth behind it. For the pair a(1), b(1) and the pair a(n), b(m), their min height is simply the sum of the pair, if you extend one pair to its lower neighbor, you have to extend the opposite pair so they are the same length, the end result is these two pairs of teeth create lower bound of total height.
  The inner teeth a(2) to a(n-1) and b(2) to b(m-1) min height can be found by recursively looking at the min height of the sum of the previous index and the next index.
  The min height of the two arrays would be the max(all pairs of teeth) after extending them all.

2.) What are the base cases of this recurrence?

- The base cases would be when you've recursed through both arrays and found the min height for every position of both upper and lower sets of teeth and so that they are now both equal in length.

3.) What data structure would you use to recognize repeated problems? You should describe both the abstract data structure, as well as its implementation.

- A 2D array to store the sums of each pair with its opposing potential pairs.

4.) Give pseudocode for a memoized dynamic programming algorithm to find the minimum height when aligning extensible teeth (a1, a2, . . . , an) and (b1, b2, . . . , bm).

```
Wrapper(a,b)
Create Array() with height set to -1
Return (Min(dynamic(arr[i, j])
Dynamic(arr[i, j])
If (arr[i, j] == -1)
if ((i == 0 && j == 0) || (i == n && j == m))
arr = max(edge pairs)
else
arr[i,j] = min(arr[i, j..m], arr[i…n, j], arr[i…n, j…m]
Else
Return arr[i,j]
```

5.) What is the worst-case time complexity of your memoized algorithm?

- The worst case for the algorithm above would be $O(n * m)$. You should only need to do a constant amount of operations per index of the matrix.

6.) Give pseudocode for an iterative algorithm to find the minimum height when aligning extensible teeth (a1, a2, . . . , an) and (b1, b2, . . . , bm). This algorithm does not need to have a reduced space complexity relative to the memoized solution.

```
NewTopArray
NewBotArray
Iterative()
Height = 0
While you haven't reach the end of both arrays
If top array leads to a smaller equal height
Increment top array
If bot array leads to a smaller or equal height
Increment bot array
If neither do
Set new min height
continue
Push the current value of the two arrays to the new arrays
```

7.)

Can the space complexity of the iterative algorithm be improved relative to the memoized algorithm? Justify your answer.

- The iterative algorithm can be optimized to only use specific columns or rows that are being looked at and does not need to store the whole matrix in memory at once.