COMP3821
Dynamic Programming Practice

Charlie Bradford z5114682

June 20, 2018

1. Given a sequence of n real numbers $A_1 \dots A_n$, determine in linear time a contiguous subsequence $A_i \dots A_j$ for which the sum of elements in the subsequence is maximised.

   Subproblem = "For each k find an $m \leq k$ such that the subsequence $A_m A_{m+1} ... A_k$ has the maximum possible sum."

   Recursion Assuming we have an optimal solution up to $k-1$, our only options are to add A_k or set $m = k$. If the previous solution is positive we will add A_k, otherwise we will replace the solution with A_k.

   Solution

   list dp
   list $bounds$
   int $max \leftarrow 1$
   $dp \leftarrow A_1$
   $bounds[1] \leftarrow (1, 1)$
   for $i \in [2..n]$ **do**

   $$dp[i],\ bounds[i] \leftarrow \begin{cases} dp[i-1] + A_i,\ (bounds[i-1][1],\ i) & dp[i-1] > 0 \\ A_i,\ (i,\ i) & dp[i-1] \leq 0 \end{cases}$$

   $$max \leftarrow \begin{cases} i & dp[i] > dp[max] \\ max & dp[i] \leq dp[max] \end{cases}$$

   od
   return $bounds[max]$

2. You are traveling by a canoe down a river and there are n trading posts along the way. Before starting your journey, you are given for each $1 \leq i < j \leq n$ the fee $F(i, j)$ for renting a canoe from post i to post j. These fees are arbitrary. For example it is possible that $F(1, 3) = 10$ and $F(1, 4) = 5$. You begin at trading post 1 and must end at trading post n (using rented canoes). Your goal is to design an efficient algorithms which produces the sequence of trading posts where you change your canoe which minimises the total rental cost.

   Subproblem = "find the minimum cost to get to trading post k"

   Recursion To find the optimum path to $k+1$, take the minimum of $F(1, k)$ and all optimum routes up to $k + F(x, k+1)$

   Solution

   list *dpCosts*
   list *dpPaths*
   $dpCosts[1] \leftarrow 0$
   $dpPaths[1] \leftarrow$ '1'
   for $i \in [2..n]$ **do**
      int $minCost \leftarrow i - 1$
      for $j \in [1..(i-1)]$ **do**

$$minCost \leftarrow \begin{cases} j & dpCosts[j] + F(j, i) < dpCosts[minCost] + F(minCost, j) \\ minCost & \text{Otherwise} \end{cases}$$

      od
      $dpCosts[i] \leftarrow dpCosts[minCosts] + F(minCost, i)$
      $dpPaths[i] \leftarrow dpPaths[minPaths] +$ ", ".format(i)
   od
   return $minPaths[n]$

3. You are given a set of n types of rectangular boxes, where the i^{th} box has height h_i, width w_i and depth d_i. You want to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is also allowable to use multiple instances of the same type of box.

Subproblem = "find the highest stack that can be built from all boxes $B_k,\ k\ \in\ [1..6n]$ that is topped by B_k"

- Note that we take six instances of each box, to account for all orientations and rotations

Recursion The maximum height for a tower ending at B_k is the maximum of all optimal solutions up to B_{k-1} + the height of B_k

Solution

Setting up all possible arrangements of boxes
list *sizes*
for $i\ \in\ [1..n]$ **do**
    $sizes.add((h_i,\ w_i,\ d_i))$
    $sizes.add((w_i,\ h_i,\ d_i))$
    $sizes.add((h_i,\ d_i,\ w_i))$
    $sizes.add((d_i,\ h_i,\ w_i))$
    $sizes.add((d_i,\ w_i,\ h_i))$
    $sizes.add((w_i,\ d_i,\ h_i))$
od
$sizes\ \leftarrow$ **sort**$(sizes)$
Finding the tallest tower
list *dp*
$dp[1]\ \leftarrow\ sizes[1][3]$
for $i\ \in\ [2..6n]$ **do**
    int $maxHeight = sizes[i]$
    for $j\ \in\ [1..(i-1)]$ **do**
        int $height\ \leftarrow\ dp[j]\ +\ sizes[i][3]$

$$maxHeight\ \leftarrow\ \begin{cases} height + sizes[i][3] & height > sizes \\ & \wedge(sizes[j][1]\ >\ sizes[i][1]\ \wedge\ sizes[j][2]\ >\ sizes[i][2]) \\ maxHeight & \text{otherwise} \end{cases}$$

    od
    $dp[i]\ \leftarrow\ maxHeight$
od
return $max(dp)$

4. Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a_1 \dots a_n$ and n cities on the northern bank with x-coordinates $b_1 ... b_n$. You want to connect as many north-south pairs of cities as possible, with bridges such that no two bridges cross. When connecting cities, you are only allowed to connect the the i^{th} city on the northern bank to the i^{th} city on the southern bank.

5. You are given a boolean expression consisting of a string of the symbols true and false and with exactly one operation and, or, xor between any two consecutive truth values. Count the number of ways to place brackets in the expression such that it will evaluate to true. For example, there is only 1 way to place parentheses in the expression true and false xor true such that it evaluates to true.

6. Consider a row of n coins of values $v_1 \dots v_n$, where n is even. We play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

7. You have n_1 items of size s_1, n_2 items of size s_2, and n_3 items of size s_3. You would like to pack all of these items into bins, each of capacity C, using as few bins as possible.

8. Find the number of partitions of n, i.e. the number of sets of integers $\{p_1, p_2, \dots, pk\}$ such that such that $p_1 + p_2 + \dots, p_k = n$. (Hint: try relaxation with respect to the number of elements in such a sum and the size of the largest number in such a sum.)

9. You are given a sequence of numbers with operations $+$, $-$, $\times$ in between, for example $1 + 2 - 3 \times 6 - 1 - 2 \times 3 - 5 \times 7 + 2 - 8 \times 9$. Your task is to place brackets in a way that the resulting expression has the largest possible value.

10. You have an amount of money M and you are in a candy store. There are n kinds of candies and for each candy you know how much pleasure you get by eating it, which is a number between 1 and 100, as well as the price of each candy. Your task is to chose which candies you are going to buy to maximise the total pleasure you will get by gobbling them all.

11. You have invented a new version of hopscotch! On the ground you have drawn a line of $n + 1$ evenly spaced hopscotch squares indexed from 0 to n, and have written an integer (positive or negative) in each square. These squares can be represented by an array $A[0..n]$ where $A[i]$ is the number written in the square with index i. You start in square 0 and will hop until you reach square n. Each time, you will jump from the current square to a square with a strictly greater index. Specifically, if you are at square with index i then you will jump to a square with index $i + k$ for some $k > 0$. Since hopping is tiring, the length of each jump cannot exceed the length of the previous jump. Your first jump may be of any length. Your score from such a jump sequence is the sum of the scores in all squares you have hopped onto. What is the maximum score you can earn?

    (a) Design an $O(n^3)$ time algorithm that determines the maximum score you can earn.

    (b) If necessary, explain how you might improve your algorithm for part (a) to run in $O(n^2)$ time.

    Subproblem = "The maximum score starting as square i with a maximum jump j"

    Recursion Subproblem(i, j) = Subproblem(i, $j-1$) if $j \geq i$ or the max of subproblem(i, $j-1$) and subproblem($i+j$, j)

    Solution

```
matrix[n][n] dp
dp[1][1] ← a[n]
for i ∈ [2..n] do
    dp[1][i] ← a[n]
    dp[i][1] ← dp[i − 1][1] + a[n + 1 − i]
od
for i ∈ [2..n] do
    for j ∈ [2..n] do
        if j ≥ i then
            dp[i][j] ← dp[i][j − 1]
        else
            min ← max { dp[i][j − 1]
                      { dp[i − j][j] + a[n + 1 − i]
        fi
    od
od
return dp[n][n]
```

12. Some people think that the bigger an elephant is, the smarter it is. To disprove this you want to analyze a collection of elephants and place as large a subset of elephants as possible into a sequence whose weights are increasing but their IQs are decreasing. Design an algorithm which given the weights and IQs of n elephants, will find a longest sequence of elephants such that their weights are increasing but IQs are decreasing.

13. You have to cut a wood stick into several pieces at the marks on the stick. The most affordable company, Analog Cutting Machinery (ACM), charges money according to the length of the stick being cut. Their cutting saw allows them to make only one cut at a time. It is easy to see that different cutting orders can lead to different prices. For example, consider a stick of length 10 m that has to be cut at 2, 4, and 7 m from one end. There are several choices. One can cut first at 2, then at 4, then at 7. This leads to a price of $10 + 8 + 6 = 24$ because the first stick was of 10 m, the resulting stick of 8 m, and the last one of 6 m. Another choice could cut at 4, then at 2, then at 7. This would lead to a price of $10 + 4 + 6 = 20$, which is better for us. Your boss demands that you design an algorithm to find the minimum possible cutting cost for any given stick.

14. You are given an $n \times n$ chessboard with an integer in each of its n^2 squares. You start from the top left corner of the board; at each move you can go either to the square immediately below or to the square immediately to the right of the square you are at the moment; you can never move diagonally. The goal is to reach the right bottom corner so that the sum of integers at all squares visited is minimal.

    (a) Describe an algorithm which always correctly finds a minimal sum path and runs in time $O(n^2)$.

    (b) Describe an algorithm which computes the number of such minimal paths.

15. A palindrome is a sequence of at least three letters which reads equally from left to right and from right to left.

(a) Given a sequence of letters S, find efficiently its longest subsequence (not necessarily contiguous) which is a palindrome. Thus, we are looking for a longest palindrome which can be obtained by crossing out some of the letters of the initial sequence without permuting the remaining letters.

(b) Find the total number of occurrences of all subsequences of S which are palindromes of any length ≥ 3.

16. There are N lily pads in a row. A frog starts on the leftmost lily pad and wishes to get to the rightmost one. The frog can only jump to the right. There are two kinds of jump the frog can make:

- The frog can jump 3 lily pads to the right (skipping over two of them)
- The frog can jump 5 lily pads to the right (skipping over four of them)

Each lily pad has some number of flies on it. Design an algorithm that maximises the total number of flies on lily pads the frog lands on getting to the rightmost lily pad.

17. You have been handed responsibility for a business in Texas for the next N days. At the beginning of each day, you may hire an illegal worker, keep the number of illegal workers the same or fire an illegal worker. At the end of each day, there will be an inspection. The inspector on the i^{th} day will check that you have between l_i and r_i illegal workers (inclusive). If you do not, you will fail the inspection. Design an algorithm that determines the fewest number of inspections you will fail if you hire and fire illegal employees optimally.

- The subproblem $P(i,\ j)$ is the minimum number of inspections failed during the first i days, given you have j workers on the i^{th} day.
  - $0 \leq j \leq i$
- $P(i,\ j) = \begin{cases} 1 & \text{if } l_i \leq j \leq r_i \\ 0 & \text{otherwise} \end{cases} + max \begin{cases} P(i-1,\ j-1) & \\ P(i-1,\ j) & j \leq i-1 \\ P(i-1,\ j+1) & j \leq i-2 \end{cases}$

matrix $m \leftarrow \{\{0\} * N\} * N$
int $i,\ j \leftarrow 0,\ 0$
for $i \leq N$ **do**
  $j \leftarrow 0$
  for $j \leq i$ **do**
    if $j \leq i-2$ **then**

$$m[i][j] \leftarrow \begin{cases} 1 & \text{if } l_i \leq j \leq r_i \\ 0 & \text{otherwise} \end{cases} + max \begin{cases} m[i-1][j-1] \\ m[i-1][j] \\ m[i-1][j+1] \end{cases}$$

    elif $j = i-1$ **then**

$$m[i][j] \leftarrow \begin{cases} 1 & \text{if } l_i \leq j \leq r_i \\ 0 & \text{otherwise} \end{cases} + max \begin{cases} m[i-1][j-1] \\ m[i-1][j] \end{cases}$$

    elif $j = i$ **then**

$$m[i][j] \leftarrow \begin{cases} 1 & \text{if } l_i \leq j \leq r_i \\ 0 & \text{otherwise} \end{cases} + m[i-1][j-1]$$

    fi
    $j \leftarrow j + 1$
  od
  $i \leftarrow i + 1$
od
return $m[N][N]$

18. Devise a dynamic programming algorithm that counts the number of non-decreasing sequences of integers of length N, such that the numbers are between 0 and M inclusive.

19. You are given a rooted tree. Each edge of the tree has a cost for removing it. Devise an algorithm to compute the minimum total cost of removing edges to disconnect the root from all the leaves of the tree.

20. You are given an ordered sequence of n cities, and the distances between every pair of cities. You must partition the cities into two subsequences (not necessarily contiguous) such that person A visits all cities in the first subsequence (in order), person B visits all cities in the second subsequence (in order), and such that the sum of the total distances travelled by A and B is minimised.

- $P(m,\ i)$ = "the minimum distance such that one travelling salesman ends at m and the other ends at i."
- If $m \neq i-1$ then $P(m,\ i)\ P(m,\ i-1) + d(i-1,\ i)$

- Else we pick some k and then $P(i-1,\ i)\ =\ \begin{cases} \min\limits_{0\ \leq\ k\ \leq\ i\ -\ 2}(P(i-1,\ k)\ +\ d(k,\ i)) \\ \sum\limits_{k=1}^{i-2} d(k,\ k+1) \end{cases}$

```
matrix m ← 0 * N * N
∫ i , j ← 1, 1
for i ≤ N do
    j ← 1
    for j < i do
        if j = i − 1 then
            m[i][j] ← min_{0 ≤ k ≤ i−2}(m[i − 1][k] + d(k, i))
                      Σ_{k=1}^{i−2} d(k, k + 1)
        else
            m[i][j] ← m[i − 1][j] + d(i − 1, i)
        fi
        j ← j + 1
    od
    i ← i + 1
od
return min_{0 ≤ k ≤ N−1} m[N][k]
```

21. A company is organising a party for its employees. The organisers of the party want it to be a fun party, and so have assigned a fun rating to every employee. The employees are organised into a strict hierarchy, i.e. a tree rooted at the president. There is one restriction, though, on the guest list to the party: an employee and their immediate supervisor (parent in the tree) cannot both attend the party (because that would be no fun at all). Give an algorithm that makes a guest list for the party that maximises the sum of the fun ratings of the guests.

22. For bit strings $X\ =\ x_1 \ldots x_m$, $Y\ =\ y_1 \ldots y_n$ and $Z\ =\ z_1 \ldots z_{m+n}$ we say that Z is an interleaving of X and Y if it can be obtained by interleaving the bits in X and Y in a way that maintains the left-to-right order of the bits in X and Y. For example if $X\ =\ 101$ and $Y\ =\ 01$ then $x_1x_2y_1x_3y_2\ =\ 10011$ is an interleaving of X and Y, whereas 11010 is not. Give an efficient algorithm to determine if Z is an interleaving of X and Y.

23. You are given n turtles, and for each turtle you have its weight and its strength. The strength of a turtle is the maximum weight you can put on top of it without breaking its shell. Find the largest possible weight you can put on top of a turtle without cracking its shell.

    - If there exists a tower of height k there must exist $k-2$ turtles such that the sum of each turtles strength and weight is equal to or less than the previous turles strength, one turtle that has weight less than the strength of the weakest turtle, and one final turtle that has strength equal to or greather than the sum of all the weights of the other turtles.
    - Alternatively, there is a tower of height k that is non-decreasing with respect to the sum of the weight and height of each turtle.