Homework 2

B06303131 經濟五 沈家睿

November 8, 2021

Reference:
b08501098 柯晨緯 & b08208032 胡材溢 & p10922001 黃佳琪 &
b05504066 李旻翰 & r10922130 陳見齊 & b09902064 楊冠柏 &
b08902149 徐晨祐 & b09505014 王聖文 & b07207063 廖政華 &
b08502041 李芸芳 & b09902129 黃柏鈞 & b09902068 凌暄

Problem 5 - Toyz' s Dog

Subproblem(a) (1) 下圖爲算解示意圖爲廖政華同學所繪

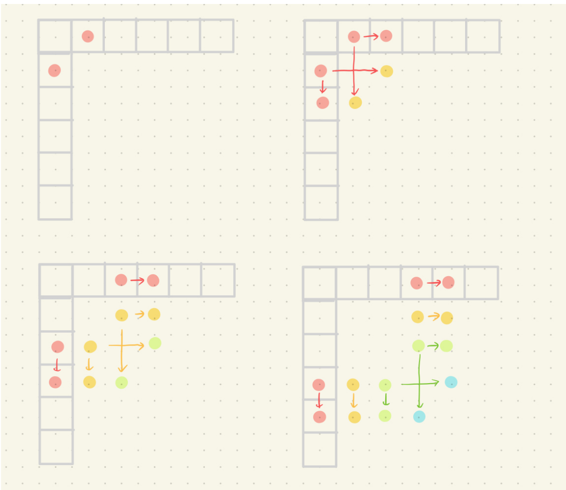

Figure 1: 算解示意圖

此爲我的 dp 轉移式

$dp(i,j) = dp(i,j-1) + E(j,j-1), if\ j-i>1$
$dp(i,j) = \min_{i' \in 0,\dots i-2} dp(i',j), if\ i-j=1$
$dp(i,j) = dp(i-1,j) + E(i-1,i), if\ i-j>1$
$dp(i,j) = \min_{j' \in 0,\dots j-2} dp(i,j'), if\ j-i=1$

最終的解爲 dp(i', N-1), $i' \in 0, \dots N-2$
及 dp(N-1, j'), $j' \in 0, \dots N-2$ 中取最小值
再加上 E(i', N) + E(N, N-1) 或 E(N-1, N) +E(N, j')
爲整趟的 min cost
正確性證明:
base case 爲 dp(0, 1) = E(1, 0) 及 dp(1, 0) = E(0, 1)
1. 考慮 j-i>1 時:
此時考慮的點爲新加入的 j 且 j>i
因此轉移方式僅可能從 dp(i, j-1) 轉移過來
因此轉移式爲 dp(i, j) = dp(i, j-1) + E(j, j-1)
所花費時間爲 O(1)
而 dp(i, j-1) 爲 min 因此此時 dp(i, j) 亦爲 optimal
2. 考慮 j-i=1 時:
此時需考慮 <i 的石頭該如何與 i 接起來
才可以達到 optimal 也就是 min cost
因此需遍歷所有小於 i 的石頭取最小值
再加上其跳至 i 所花之能量
因此所花時間爲 O(N)
且因取的值爲最小值因此 dp(i, j) 亦爲 optimal
3. 考慮 i = j 時:
因爲石頭不可踩兩次
因此不會有此 case
4. 考慮 i-j>1 時:
同 1.
5. 考慮 i-j=1 時:
同 2.
由左至右且同時由上至下
以 bottom up 的方式將 dp 式建置出來
每條 col,row 所花時間爲 O(N)
總共需更新 N 條 col, row 因此總時間複雜度爲 $O(N^2)$
因已討論所有可能情況
且每個 subcase 皆取 optimal solution
因此最終解亦爲 optimal

(2) 由 (1) 示意圖可以發現
實際上僅關心當前的 col, row
因此可以使用兩條陣列代表該 col 及 row
更新方式為由左至右且同時由上而下
先 O(1) 由左至右轉移或由上至下轉移 |i-j|>1 的格子
再將最小值填至對角線下一格或對角線右邊一格
也就是 |i-j|=1 的格子
直到最後一條 row,col 為止
最終的解為 dp(i', N-1), $i' \in 0, \ldots N-2$
及 dp(N-1, j'), $j' \in 0, \ldots N-2$ 中取最小值
再加上 E(i', N) + E(N, N-1) 或 E(N-1, N) +E(N, j')
為整趟的 min cost
因只需開兩條陣列因此空間複雜度為 O(N)

(3) 下圖為找 path 示意圖亦為廖政華同學所繪

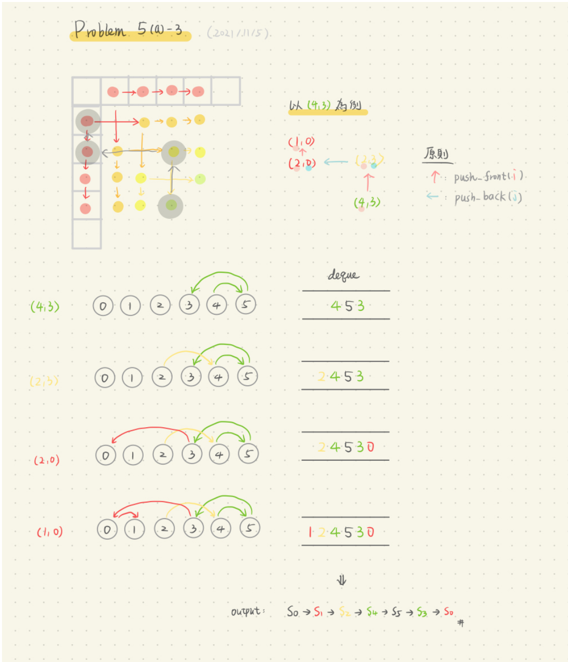

Figure 2: 找 path 示意圖

由示意圖及上兩題的演算法可知
可以在轉移時記錄產生最小值的座標
並使用額外兩條陣列紀錄之
此紀錄因使用相同之演算法
只是在轉移時順便記錄座標
因此時間複雜度依然為 $O(N^2)$
在回溯路徑時可藉由這兩條陣列
交互比較重建出所選之最佳路徑
因額外開的兩條陣列亦與 N 有關
因此空間複雜度亦為 O(N)

Subproblem(b) (1)

1. 考慮 i > j 時僅需考慮 i+1 這顆石頭如何放置
因此 dp 轉移式爲:
$dp(i+1, j, H - D_{i+1}) = dp(i, j, H) + E(i, i+1)$
此爲將 i+1 這顆石頭放在去程最後一點
$dp(i, i+1, H') =$
$\min(dp(i, i+1, H'), dp(i, j', H') + E(i+1, j'))$
此爲將 i+1 這顆石頭放在回程第一點
2. 考慮 j > i 時僅需考慮 j+1 這顆石頭如何放置
因此 dp 轉移式爲:
$dp(j+1, j, H' - D_{j+1}) =$
$min(dp(j+1, j, H' - D_{j+1}), dp(i', j, H') + E(i', j+1))$
此爲將 j+1 這顆石頭放在去程最後一點
$dp(i, j+1, H) = dp(i, j, H) + E(j+1, j)$
此爲將 j+1 這顆石頭放在回程第一點
若 H-D<1 則 dp(i, j, H) $= \infty$
若這樣建置 dp 轉移式
則會 bottom up 將表全部填完
因此會花 $O(HN^2)$ 時間

(2) 正確性證明:

base case 爲 $dp(0,1,H) = E(1,0)$
及 $dp(1,0,H-D_1) = E(0,1)$
1. 當 i > j 時考慮 i+1 這顆石頭如何放置
只會有兩種可能
一種爲放在去程最後一點
此時因爲 i > j
因此可以直接 O(1) 轉移到 $dp(i+1, j, H-D_{i+1})$
另一種情況爲放在回程第一個點
此時需去搜尋接過去哪個 j 點會使 dp 值更小
並將其依序填入各個 H 的表內
因爲每個點實際上需搜尋高度爲 H 的表
且最多有 N 個點要搜
因此會花費 O(HN) 的時間
2. 當 j > i 時考慮 j+1 這顆石頭如何放置
只會有兩種可能
一種爲放在去程最後一點
此時因爲 j > i
此時需去搜尋哪個 i 點接過來會使 dp 值更小
因爲每個點實際上需搜尋高度爲 H 的表
且最多有 N 個點要搜
因此會花費 O(HN) 的時間
另一種情況爲放在回程第一個點
因此可以直接 O(1) 轉移到 $dp(i, j+1, H)$
而若產生 H-D<1 之情況
則 dp 值爲 ∞
結合以上之分析
因每個 subcase 皆討論到
且會取得 optimal subproblem 的解
因此可以得到最終的 optimal solution
且因要填完所有三維表格
時間複查度爲 $O(HN^2)$

Problem 6 - Howard’s Desiring Order of Courses

(1) No Assumption：98141210
Under Assumption 3：981120

(2) greedy choice property:
因爲輸進來的數字僅有 0-9
因此可以開一條長度爲 10 的陣列分別存下他們的次數
使用 counting sort 的技巧
將原本輸入的數字由大到小排好
此步驟會花 O(n) 時間
接著再將所有數字接起來
便可得到最大數
此步驟亦會花 O(n) 時間
因此總時間複雜度爲 O(n)

(3) greedy choice property:
使用 merge sort 的技巧
但 merge 的比較方式改爲
考慮兩個數 a, b
採一前一後及一後一前接起來
產生出來的兩個數 ab, ba 做比較
若比較大者爲 ab
則 a 先放入
反之則爲 b
將依此方法 sort 完之陣列所有數接起來
則可產生最大的數
因此總時間複雜度爲 $O(n \log n)$

(4) greedy choice property:
使用類似 (2) 之演算法
一樣紀錄 0-9 的 digit 個數分別有幾個
且將其的總和求出
爲了滿足 Assumption3 因此將總和 mod 3 後分 case 討論
case 1. 餘 1:
若有 1, 4, 7 則只需拔除其中最小的一個 digit
若完全沒有 1, 4, 7 之任一一個 digit
則一樣由小到到大選擇拔除兩個 2, 5, 8
case 2. 餘 2:
若有 2, 5, 8 則只需拔除其中最小的一個 digit
若完全沒有 2, 5, 8 之任一一個 digit
則一樣由小到到大選擇拔除兩個 1, 4, 7
接著將剩餘的 digit 由大到小放好
此步驟花 O(n)
接著再將 sort 完之陣列所有 digit 接起來
即可得最大數
此步驟亦花 O(n)
若發生無法拔除 digit 以滿足 Assumption3
則直接輸出 0
因此總時間複雜度爲 O(n)

(5) maximum satisfying value = 98653

(6) greedy choice property:
分爲兩步驟
第一步爲取 i 個元素且不改變其順序的情況下
能得到的最大數組
所採 strategy 爲
先初始化一空陣列作爲輸出之陣列
再算出最多能丟掉幾個數字 pop_k
在沒花完 pop_k 時
由左至右遍歷原陣列
若當前 digit 比欲輸出之陣列的尾端元素大時
則以當前 digit 不斷取代之
反之則推入尾端
重複直到看完原陣列所有元素
接著從輸出之陣列中取前 i 個元素
即可得到取 i 個元素且不改變其順序的最大數組
則此演算法之複雜度爲 O(n)
有了上述之演算法
接著可以窮舉 P, M 兩陣列共取 k 個元素下的所有組合
進行第二步的演算法
由第一步可得兩陣列
長度分別爲 i 及 k-i
挑選填入最高位數的 strategy 爲
將兩陣列各自所有 digits 接起來
個別產生兩個數 a 及 b
此複雜度爲 O(n)
接著將兩數以 ab, ba 的方式接起來
比較 ab 及 ba 之大小
若 ab 較大
則將產生 a 之陣列的第一個元素放入最高位數
反之則將產生 b 之陣列的第一個元素放入最高位數
放入最高位數的元素即從其所屬陣列拔除
重複上述直到兩陣列被清空
因此最多需重複 O(n) 回
因此第二步演算法時間複雜度爲 $O(n^2)$
而因需窮舉所有 i, k-i 由 P,M 產生之 subsequeuce
因此總時間複雜度爲 $O(kn^2)$