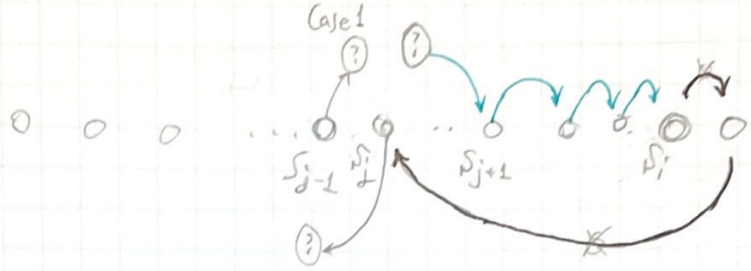
#5 Toyz's Dog

(1)

→ Subproblem: $dp(i,j)$: 从玄程最後一個 point，S_j 回程第一個 point 減去 $E(i,N)+E(N,j)$ 的得的最少 energy.

→ OPT substructure:

假設已知某optimal solution $dp(i,j)$, assume $i>j$, → S_i must be S_{N-1}



因爲 S_i 是玄程最後一個 state，S_j 是回程 1st state，又每個 state 都要走，所以可保證 S_j 一路走到 S_i 都是玄程時經過的。$S_j \sim S_i$ 的 cost:

$$C = \sum_{j<k<i} E(k,k+1)$$

至於 S_j 以前的 case 可以考慮 S_{j+1} 的前一個 state，S_j 的下個 state 的所有可能性情況：

Case 1 S_{j-1} ∈ 玄程的 path. (i.e. S_{j-1} 走向 S_{j+1})

Case 1-① 回程時 S_j 的下個 state 爲 S_0 → cost $dp(j-1,0)+E(j-1,j+1)+E(j,0)+C$

Case 1-② … S_1 $dp(j-1,1)+\cdots+E(j,1)+C$

Can reduce to $O(1)$

Case 1-③ … S_{j-2} $dp(j-1,j-2)+\cdots+E(j,j-2)+C$

Case 2 S_{j-1} ∈ 回程的 path (i.e. S_j 走向 S_{j-1})

Case 2-① 玄程時，S_{j+1} 的前個 state 爲 S_0 → $dp(0,j-1)+E(0,j+1)+E(j,j-1)+C$

Case 2-② … S_1 $dp(1,j-1)+E(1,j+1)+E(\cdots)+C$

Case 2-③ … S_{j-2} $dp(j-2,j-1)+E(j-2,j+1)+\cdots+C$

同理，考慮 $i<j$ 也是類似的情形 (已知 $dp(i,j)$ 爲 OPT，以上每個列舉的 case 的 subproblem 都是 OPT，可以使用用簡單反證法證明，若非 OPT，則可以找到更好的 OPT 使得 $dp(i,j)$ 更小了)

- recursively define OPT:

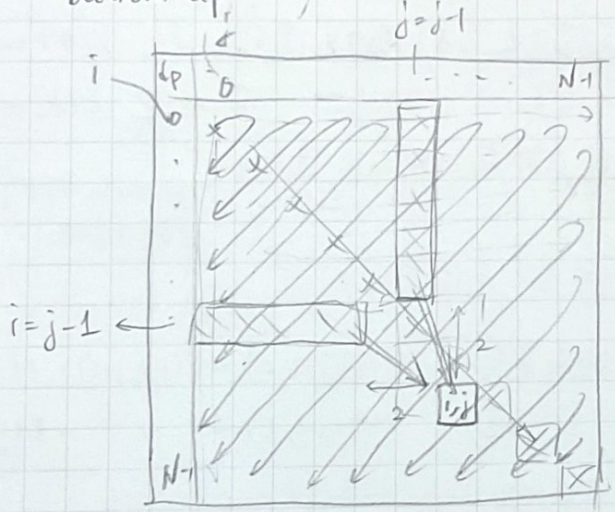
E(i,j) if j=0, i=1 (i if i=j) O(N)

E(j,j) if i=0, j=1

$$dp(i,j)=\begin{cases}\min\left\{\min_{0\le k\le j-2} dp(j-1,k)+E(j,k)+E(j-1,j+1),\ \min_{0\le k\le j-2} dp(k,j-1)+E(k,j+1)+E(j,j-1)\right\}+\sum_{j<k<i} E(k,k+1) & \text{if } i>j\\ \min\left\{\min_{0\le k\le i-2} dp(i-1,k)+E(i+1,k)+E(i-1,i),\ \min_{0\le k\le i-2} dp(k,i-1)+E(i+1,i-1)+E(k,i)\right\}+\sum_{i<k<j} E(k+1,k) & \text{if } j>i\end{cases}$$

→ O(4*N) = O(N)

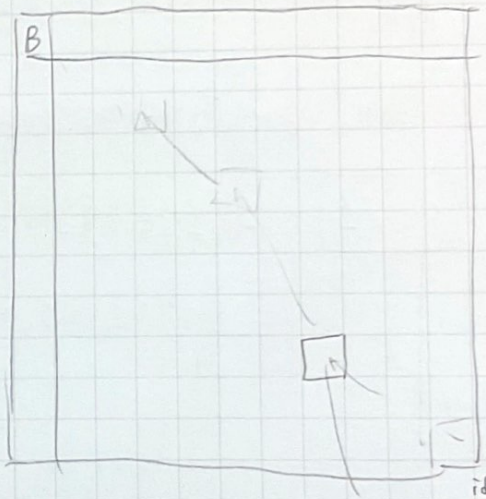
- buttom-up



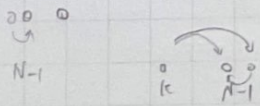
space = N^2

time complexity = $O(N*N^2)$ = $O(N^3)$

- backtracking



- output: $\min\left\{\min_{0\le k\le N-2}\left[dp(N-1,k)+E(N-1,N)+E(N,k)\right],\ \min_{0\le k\le N-2}\left[dp(k,N-1)+E(N,N-1)+E(k,N)\right]\right\}$



(2)(3)

若要解上述作弊減到 $O(N)$ space complexity, $O(N^2)$ time complexity, 可以建下面4個array

$A\_i(i) = \min_{k<i} dp(i,k)$ (i.e. 以i為起點最後一個state, 且 $i>j$ → 目標在 S_{i+1})

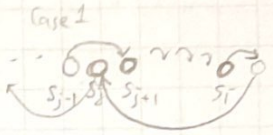
$B\_i(i) = \operatorname{argmin}_{k<i} dp(i,k)$ 上述的 k

$A\_j(j) = \min_{k<j} dp(k,j)$ (i.e. 以j為回程第一個state且 $j>i$ → 目標為 S_{j+1})

$B\_j(j) = \operatorname{argmin}_{k<j} dp(k,j)$ 上述的k

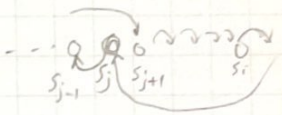
- subproblem

已知某optimal solution $dp(i,j)$ if $i>j$ then:



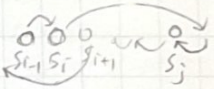
Case 1 S_{j-1} ∈ 去程 $\xrightarrow{cost}$ $A\_i(j-1) + E(j-1,j+1) + E(j, B\_i(j-1)) + \sum_{j\le k<i} E(k,k+1)$

Case 2 S_{j-1} ∈ 回程 $\xrightarrow{cost}$ $A\_j(j-1) + E(j,j-1) + E(B\_j(j-1), j+1) + \sum_{j\le k<i} E(k,k+1)$



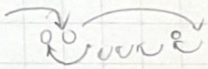
if $j>i$ then

Case 1 S_{j-1} ∈ 去程 $\xrightarrow{cost}$ $A\_i(i-1) + E(i-1,i) + E(i+1, B\_i(i-1)) + \sum_{i\le k<j} E(k+1,k)$



Case 2 S_{j-1} ∈ 回程 $\xrightarrow{cost}$ $A\_j(i-1) + E(i+1,i-1) + E(B\_j(i-1), i) + \sum_{i\le k<j} E(k+1,k)$

//Notes//



所以在填前一頁的表時, 一面求 $dp(i,j)$, 一面更新

if $i>j$ 且 $dp(i,j) > A\_i(i)$ → $A\_i(i) = dp(i,j)$, $B\_i(i) = j$

if $j>i$ 且 $dp(i,j) > A\_j(j)$ → $A\_j(j) = dp(i,j)$, $B\_j(j) = i$

- OPT:

$$dp(i,j) = \begin{cases} E(i,j) & \text{if } j=0, i=1 \\ E(j,i) & \text{if } i=0, j=1 \\ \min\{ A\_i(j-1) + E(j-1,j+1) + E(j, B\_i(j-1)) + \sum_{j\le k<i} E(k,k+1), \\ \quad A\_j(j-1) + E(j,j-1) + E(B\_j(j-1), j+1) + \sum_{j\le k<i} E(k,k+1) \} & \text{if } i>j \\ \min\{ A\_i(i-1) + E(i-1,i) + E(i+1, B\_i(i+1)) + \sum_{i\le k<j} E(k+1,k), \\ \quad A\_j(i-1) + E(i+1,i-1) + E(B\_j(i-1), i) + \sum_{i\le k<j} E(k+1,k) \} & \text{if } i<j \end{cases}$$

O(1)

define on $i \neq j$, $0 \le i \le N$, $0 \le j \le N$

上述作法 time complexity 為同前面 遍歷順序走過所有 i,j 並更新 $A\_i(i)$, $B\_i(i)$, $A\_j(j)$, $B\_j(j)$ (並更新 由 OPT ~ $O(1)$)

來比較 $\sim O(N^2)$

且只須存 $A\_i(i)$, $B\_i(i)$, $A\_j(i)$, $B\_j(j)$ 來比較, space complexity $\sim O(N)$

- output: $\max\{A\_j(N-1)+E(B\_j(N-1),N)+E(N,N-1),\ A\_i(N-1)+E(N,B\_i(N-1))+E(N-1,N)\}$

optimal path: 若 output 選 $A\_j(N-1)$, 則 $S_{B\_j(N-1)+1} \sim S_{N-2} \in$ 回程 state

$S_{B\_j(N-1)} \in$ 去程 state

再同理,由判斷 $\max\{A\_j(B\_j(N-1)-1),\ A\_i(B\_j(N-1)-1)\}$

一路 backtrack 回去 until $B(\cdot\cdot)=0$

若 output 選 $A\_i(N-1)$ 則 $S_{B\_i(N-1)+1} \sim S_{N-2} \in$ 去程 state

$S_{B\_i(N-1)} \in$ 去程 state.

再同理,由判斷 $\max\{A\_j(B\_i(N-1)-1),\ A\_i(B\_i(N-1)-1)\}$

一路 backtrack

(b) - modify (2)、(3) 小題定義的 $A\_i(i)$、$A\_j(j)$ 為：

$A\_i(i,h)$：沒有接最後一個 state（不考慮以有 i 以後的指標）

生命還有 h 所花費的 minimal every cost

$A\_j(j,h)$：j 為回程第一個 state，且目標在 $j+1$ 時，

生命還有 h 所花費的 minimal every cost

一則存在 optimal substructure.

$$A\_i(i,h)=\begin{cases} x & \text{if } i=0,\ h<D_0 \\ 0 & \text{if } i=0,\ h\geq D_0 \\ \min\Big\{ \min\limits_{\substack{j<i \\ \text{condition1}}}\Big\{ A\_i(j-1,h-D)+E(j-1,j+1)+E(j,B\_i(j-1,h-D)) \\ \qquad +\sum_{j<k<i} E(k,k+1)\Big\}, \\ \min\limits_{\substack{j<i \\ \text{condition2}}}\Big\{ A\_j(j-1,h-D)+E(j,j-1)+E(B\_j(j-1,h-D)) \\ \qquad +\sum_{j<k<i} E(k,k+1)\Big\}\Big\} \end{cases}$$

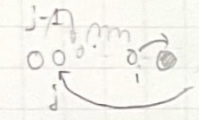
where $D=\sum_{j+1\leq k\leq i} D_k$

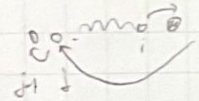
(condition1) for the chosen "j", $\sum_{j\leq k\leq i} D_k \leq h$.

and $A\_i(j-1,h-D)$ 或 $A\_j(j-1,h-D)$ 不可為 x。 上面為 min　下面為 min

if there is no subproblem satisfies the condition

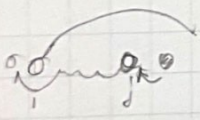
let $A\_i(i,h)=x$

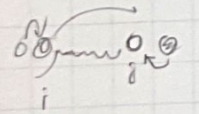




令 $B\_i(i,h)$ 為上面 $\hat{\min}$ 找到的 j

$$A\_j(j,h)=\begin{cases} 0 & \text{if } j=0 \\ \min\Big\{ \min\limits_{\substack{i<j \\ \text{condition3}}}\Big\{ A\_i(i-1,h-D_i)+E(i-1,i)+E(i+1,B\_i(i-1,h-D_i)) \\ \qquad +\sum_{j<k<i} E(k+1,k)\Big\}, \\ \min\limits_{\substack{i<j \\ \text{condition4}}}\Big\{ A\_j(i-1,h-D_i)+E(i+1,i-1)+E(B\_j(i-1,h-D_i),i) \\ \qquad +\sum_{i<k<j} E(k+1,k)\Big\} \\ \Big\} \end{cases}$$





令 $B\_j(j,h)$ 為上面 min 找到的 i

(condition 2) for the chosen "i", $D_i \leq h$

and $A\_i(i-1,h-D_i)$ 或 $A\_j(i-1,h-D_i)$ 不可為 x　上面 min　下面 min

if there is no subproblem satisfies the cond

let $A\_j(j,h)=x$.

- output: $\min_{h \le H} \{ A\_i(N-1, h),\ A\_j(N-1, h) \}$

- backtrack: using B_i(i,h)、B_j(i,h), the way is exactly the same as described at previous question.

2.

- time complexity. 建 A_i & A_j & B_i & B_j 的 dp 表，各自為 N*H 的大小，

  用前頁規則找到每個entry的 cost 為 O(N)

  ∴ total time ~ $O(N^2 H)$

- correctness. 前頁2個 min 內考慮的 subproblem case 和 (a) 小題類似，

  先窮舉所有可能的 去程 first point (for A_i) 或 回程 last point (for A_j)，

  再考慮該點的前一個點在去程或在回程的2種情形。
  ↳ subproblem.

  optimal substructure: 可由反證法證，若包含的 subproblem 非代入OPT解的話 (OPT: cost最小)，

  則代入該subproblem的OPT會使原本的OPT更好 (cost更小)，矛盾，

  所以原先OPT所包含的subproblem 必定本來也須是OPT.

(字很亂...
助教辛苦了@@
我以後會乖²打latex)

Problem 6

(1) P=[1,0,9,81,4,12] under assumption 3 (multiple of 3), maximum concatenation result.

no assumption → 9 81 4 12 10 0 ※

餘 → ① ① ② (total) 須刪2個餘①的數

assumption 3 → 9 81 120 #

(2) algorithm. find maximum concatenation under assumption 1. (always single digit num) O(nlogn)

sort input array {Pi} 從大到小、將結果從大到小 concate and output

↳ O(NlogN) ※

mergesort

(3) algorithm ... under assumption 2 (always positive integer) O(nlogn)

- 將每個數視為string，由左至右比大小，若2者位數不同，特比較小的數的——後面每一位不足都以最小位數(某數)來比(的數)。

ex 98>987 ; 981>987

- 用上述sorting rule 做 NlogN (mergesort) 排序後，將結果由大到小 concate and output.

(但比較的overhead 為 max{P} 的integer長度 w，假設 w≪N)

(4) algorithm under assumption 1&3 — single digit num & result%3 ==0

- 一個 array of num {x1, x2, x3 ... x9} 代表每個數字的數量。

- Scan the input array, 同時 update 每個數字的加總餘數 y. %3.

以即 {x1, x2, x3 ... x9} ~O(N)

- 若最後 y=1，表示餘1。因此須從餘1的最小的數中取一個 P∈{1,4,7} (且 $x_p \geq 1$)，x_p--

若沒有P，則餘2的數中取2次 P∈{2,5,8} (且 $x_p \geq 1$)

每次都取最小且 $x_p \geq 1$ 的，取後 x_p--

若沒有合法的 P1、P2 令 x1, x4, x7, x2, x5, x8 = 0

~O(constant)

- 若最後 $y=2$，表示餘 2，因此從餘 2 的最小的數取一個 $p \in \{3,5,8\}$ ($p \in \{3,5,8\}$, $x_p \geq 1$)

x_p -- ← can choose if $x_p \geq 1$

若沒有合法 p，從餘 1 的最小的數取 2 次，取後 x_p 且 $x_p \geq 1$ ($p \in \{1,4,7\}$)

若也沒有合法的 p_1, p_2，令 $x_1, x_4, x_9, x_2, x_5, x_8 = 0$.

↳ $O(\text{constant})$

- 最後將 k, $1 \leq k \leq 9$ 由大到小，用 x_k 個 k concate 在一起印出. $\sim O(N)$

i.e.

$\underbrace{9\cdots9}_{x_9}\ \underbrace{8\cdots8}_{x_8}\ \underbrace{7\cdots7}_{x_7}\ \underbrace{6\cdots6}_{x_6}\ \underbrace{5\cdots5}_{x_5}\ \underbrace{4\cdots4}_{x_4}\ \underbrace{3\cdots3}_{x_3}\ \underbrace{2\cdots2}_{x_2}\ \underbrace{1\cdots1}_{x_1}$

total $O(N)$

(5) $n=5, k=5$ C $P_i = [3, 4, 6, 5, 0]$

E $M_i = [9, 0, 5, 8, 3]$

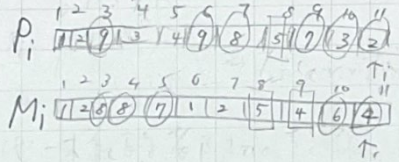
maximum

98653

$E_1 E_4 C_3 C_4 E_5$

可以 brute force 找

(6) $O(kn^2)$ algorithm to find the maximum given P_i & M_i



$\{p_i\} = \{3, 6, 7, 9, 10, 11\}$

$= 7 - 6 4 3 2$

- 定義 Maximum decreasing subarray $\{p_i\}$ 為 P_i 中所能找到最長，且滿足所有 $p \in \{p_i\}$ 的數，其值皆大於所有 index 比之大的數，tag 起來

同理，也找一個 M_i 的 maximum decreasing subarray. $\sim O(N^2)$ ($\{m_i\}$)

(先找第1大，再找第1大往後找第大 …)

大至小 ✓

- 若找到的這 2 個 subarray 長度加總 $\leq k$，則直接以 2-pointer 的操作由左至右把它們 concate 為一個 decreasing num string，concate 到長度 $=k$，後面小的丟掉就好 $\sim O(N)$

- 若找到的這 2 個 subarray 長度加總 $m > k$，則須再加入 $k-m$ 個數，因此對於前面沒選過 subarray 的區間，選一個最右邊的區間，再找 maximum subarray. 若該 subarray 長度 $> k-m$，一樣將後面小的丟掉，因為有 P_i, M_i 2 個要考慮，以後區間往右的第一個 $\in$ maximum decreasing subarray 的數若不足 $k-m$，再同理找另一個 subarray，直到足夠為止. $\sim O(kN^2)$

比後小的優先