ADA HW2

Problem 5

(a)

(1)
首先把dp值都初始化成INT_MAX
dp[0][0] = 0
轉移式：
令 $k = max(i, j)$
$dp[k+1][j] = min(dp[k+1][j],\ dp[i][j] + E(i, k+1))$
$dp[i][k+1] = min(dp[i][k+1],\ dp[i][j] + E(k+1, j))$

對於每次轉移都是向後轉移，而且每次轉移是$O(1)$，表的大小為n^2
因此時間複雜度為$O(n^2)$

令$k = max(i, j),\ dp[i][j]$是走過前k個的最小cost值。
而在i, j之間的石頭會被逐一踩過所以當從dp[i][j]推到踏過第k+1個石頭時，可以是在去程走過第k+1個石頭或在回程走過，分別是$dp[k+1][j]$和$dp[i][k+1]$。
又因為不能回頭，所以k+1的前一步必是i、後一步必是j

(2)
我先計算j值相同時改變i所需要填的表，也就是直的填下來
每次填表時只需要紀錄當前的行和下一行
但因為填的時候可能會改變到$dp[i][i+1],\ dp[i+1][i] (0 < i < n)$
所以還需要再開兩個陣列紀錄這些位置，總共需要4個長度為n的陣列
因此空間複雜度可以壓在$O(n)$
最後還需要紀錄i或j為n-1時dp陣列的最小值，其中最小的就是答案

(3)
對於對角線之下的每一格而言，走到他們的方式一定會是從同行上面的對角線下一格$(dp[a+1][a], a \in [0, n])$一路走下來
對於在對角線之上的每一格而言，走到他們的方式一定會是從同列右邊的對角線上一格$(dp[a][a+1], a \in [0, n])$一路走過來

因此首先要記錄$dp[a+1][a]$是從哪裡走來，但因為他只有可能從上面同行的某個點走來，所以更新時只要記錄是從哪一列走來
接著還要記錄$dp[a][a+1]$是從哪裡走來，而這些在對角線上面一格的點只會從他同列的左邊某行過來，這樣就會再回到最開始的狀態，而這邊可以壓縮成他的上一步是哪行

這樣要記錄的就是對角線的上下一格是從哪些地方走來，對於每次更新最大值的時候再同步更新路徑，這樣額外需要紀錄的東西就是對角線上下一格的來源以及當前最大值的路徑，因此多紀錄的東西大小是$O(n)$，原本的空間複雜度也是$O(n)$
因此總複雜度為$O(n)$

(b)

(1)
重新定義dp陣列為dp[i][j][h]
i, j的定義和前面(a)的一樣，h代表則是在這格時剩下h的血量
dp值則代表在i, j的定義下，血量剩下h的最小cost
首先把dp值都初始化成INT_MAX
dp[0][0][H] = 0
轉移式：
令 k = max(i, j)

```
if(h - D[k+1]>0&&dp[i][j][h]!=INT_MAX)
    dp[k+1][j][h - D[k+1]] = min(dp[k+1][j][h -
D[k+1]], dp[i][j][h]+E(i, k+1)
dp[i][k+1][h] = min(dp[i][k+1][h], dp[i][j]
[h]+E(k+1, j)
```

(2)
對於dp[i][j][h]
dp表的大小為n*n*h，每次都需走遍整張表來確認當前生命值是否能繼續往城堡前進，轉移時間為$O(1)$，共填N*N*H格，因此時間複雜度為$O(HN^2)$
因為只有在前往城堡的時候才會引爆炸彈，所以只有在每次dp修改i的時候才需要確認Health是否足夠去走下一步，回程則可以當作每次炸彈的傷害為0，所以可以直接貼上

Problem 6

(1)

NO Assumption : 98141210
Assumption 3 : 981120

(2)

先把preferance sort成遞減的數列，然後把所有數字concatenate起來就是答案。
sort時可以使用merge sort，如此複雜度就是$nlogn$

(3)

同樣對整個數列進行merge sort，不同的是cmp function要做些更動
compare function要變成先把兩個數a, b組合成ab和ba
如果$ba > ab$則把b排到a前面，否則就把a排到b前面
每次compare複雜度是$O(1)$，所以總複雜度同樣是$O(n\ log\ n)$
最後再將排列好的數列從大到小組合起來就是最大值

(4)

先把輸入的數字分別記錄各有幾個(Ex 9有5個)，總共10個數字
接著把這些數字分成mod3後為0, 1, 2的組別並記錄各組有
k_0, k_1, k_2 個，時間複雜度為O(1)
mod3為0的組可以全取，而mod3=1和mod3=2的組最多可以取$min(k_1, k_2)$個
令$p = min(k_1, k_2)$，我們要從mod3=1和mod3=2中由大到小各取出p個
因為每組只有3種數字所以時間複雜度為O(1)
接著取出的數字們可以用index在0~9的陣列裡儲存各有幾個
最後再把數字從9開始向下取然後串在一起，因此時間複雜度為$O(n)$

(5)

98653

(6)

先分別對P和M建立各一組dp陣列$dp1[i][j], dp2[i][j]$，分別代表在P、M中前i個取j個有最大值的值和取法$(j <= k)$
$dp1[i][j] = max(dp1[i][j-1], dp1[i-1][j-1] * 10 + P[j])$
$dp2[i][j] = max(dp2[i][j-1], dp2[i-1][j-1] * 10 + M[j])$

如此可以得到在P、M中取k個$(k \in [0, n])$有最大值的取法
每次轉移取max是$O(N)$，需要填$k * n$大小的表，所以總複雜度是$O(kn^2)$

對於$dp1[i][n],\ dp2[i][n]$紀錄在P、M之中能在長度為i時得到最大值的取法
當有了$dp1$和$dp2$後開始枚舉所有取法作為當前狀態，也就是從P取j個，從M取k-j個 $j \in [0, k]$
取出後要開始合併，採用類似merge sort中merge時的技巧
對於兩個陣列$dp1[j][n]$、$dp2[k-j][n]$，每次只關注他們當前最前面的值
每次都從兩者中取較大的和當前狀態組裝起來，如果最前面的值相同則開始向後比較
直到有一方較大或全部相同為止，全部取完後再和前面狀態中的最大值比較
每次比較最糟都是$O(n^2)$、要比較k次，因此複雜度為$O(kn^2)$