2. **(25 points)** Pots of Gold

There are N pots of gold arranged linearly. Alice and Bob are playing the following game. They take alternate turns, and in each turn they remove (and *win*) either of the two pots at the two ends of the line. Alice plays first. Given the amount of gold in each pot, design an algorithm to find the maximum amount of gold that Alice can assure herself of winning.

設給定 $p[1,\dots,n]$ 為 pots of gold 且 n 為 even
而 2 个 player, 輪流從 pots 的最前/後端取走一 pot, 求 player A 最大可獲利金額?

Example: $p[1,\dots,4]$: 8, 15, 3, 7

A 取 8
B 取 15
A 取 7
B 取 3

$\Rightarrow$ A不為 optimal　optimal 為 $7+15=22$

(∴不能用 greedy algorithm)

定義子問題為 $d[i,j]$ 為 $p[i,\dots,j]$ 在A為先手下可得到的最大獲利

$$
d[i,j] = \begin{cases} 0 & \text{if } i \ge j \\ \max\{p[i], p[j]\} & \text{if } j-i=1 \\ \max\{\underline{d[i+1,j-1]+p[i]},\ d[i+1,j-1]+p[j]\} & \text{otherwise} \end{cases}
$$

∵選 $p[i]$ 後 B 不一定選 $p[j]$, 同理

Example:

$i \backslash j$	1	2	3	4
1	0	15	8	23
2	0	0	15	15
3	0	0	0	7
4	0	0	0	0

$\Rightarrow$ Wrong

∵ B也會以最佳選法行动

(a) **(10 points)** Clearly state the set of subproblems that you will use to solve this problem.

Solution: We will represent the pots as an array P with $P[i]$, for $1 \leq i \leq N$, equal to the amount of gold in i-th pot. Our subproblem will be $V[i,j]$, for $1 \leq i \leq j \leq N$, the maximum amount of gold that Alice can assure herself of winning when facing the (contiguous) subsequence $P[i:j]$ of the pots and being the first one to play.

(b) **(10 points)** Write a recurrence relating the solution of a general subproblem to solutions of smaller subproblems.

Solution: Consider a situation in which Alice is facing a sequence $P[i:j]$ of the pots, for $1 \leq i \leq j \leq N$, and it is her turn to move.

If $i = j$ then $V[i,i] = P[i]$, since she can just grab the only remaining pot of value $P[i]$.
If $i+1 = j$, she knows that whichever among the two pots of values $P[i]$ and $P[i+1]$ she grabs the other one will be removed by Bob. So, $V[i,i+1] = \max\{P[i], P[i+1]\}$.
Now, consider the case of $i+1 < j$. Alice has an option of removing either the i-th or the j-th pot.

If she decides to remove i-th pot then she gains $P[i]$ gold and Bob has to make his move while facing the sequence $P[i+1:j]$. During his move he can remove either $(i+1)$-th or j-th pot, which would leave Alice facing the sequence $P[i+2:j]$ and $P[i+1:j-1]$ respectively. Note that at this point, it would be Alice's turn again so the maximum amount of gold she could assure herself of winning would be $V[i+2:j]$ and $V[i+1:j-1]$ respectively. So, summarizing, if Alice decides to remove i-th pot, she can be sure of winning either $P[i]+V[i+2:j]$ or $P[i]+V[i+1:j-1]$ *depending on Bob's choice*. Since we don't know anything about Bob's strategy, the only amount of gold Alice can be sure of winning after removing i-th pot is the *minimum* of these two values.

Analogous reasoning implies that if Alice decides to remove j-th pot then the maximum amount of gold she can be sure of winning is $\min\{P[j]+V[i+1:j-1], P[j]+V[i:j-2]\}$. So, since Alice can choose between the removal of the i-th and the removal of the j-th pot, her guaranteed amount $V[i,j]$ of gold to be won in this situation is

$$V[i,j] = \max\{P_1[i,j], P_2[i,j]\},$$

不知道bob的strategy

where

$$P_1[i,j] = \min\{P[i] + V[i+2:j], P[i] + V[i+1:j-1]\},$$

and

$$P_2[i,j] = \min\{P[j] + V[i+1:j-1], P[j] + V[i:j-2]\}.$$

This leads to the following recurrence relation for $V[i,j]$:

$$V[i,j] = \begin{cases} P[i] & \text{if } i = j \\ \max\{P[i], P[i+1]\} & \text{if } i+1 = j \\ \max\{P_1[i,j], P_2[i,j]\} & \text{otherwise} \end{cases}.$$

Comment: Note that the above reasoning is a bit subtle. In particular, we are not allowed to assume anything about Bob's behavior, i.e. he does not need to follow a strategy that maximizes his gain. Of course, one might reduce the reasoning to the cases when he actually follows such a strategy[1]*, but such argument must be made explicitly.*

Note that once such an argument is made, one can develop yet another (more concise) recurrence relation for $V[i,j]$: 最佳的策略 ！＝每次拿兩邊中最大的

bob用最佳的策略

$$\boxed{V[i,j] = \begin{cases} P[i] & \textit{if } i = j \\ SumP[i:j] - \min\{V[i+1:j], V[i:j-1]\} & \textit{otherwise} \end{cases},}$$

where the $SumP[i:j]$ is the total amount of gold in subsequence $P[i:j]$, and this is computed efficiently using memoization.

(c) **(5 points)** Analyze the running time of your algorithm, including the number of subproblems and the time spent per subproblem. Hint: your overall algorithm should be $O(N^2)$.

Solution:

First, we note that in our recurrence relations the subproblems to which we are reducing evaluation of $V[i,j]$ correspond to strictly smaller number of pots. Thus we know that our recursion will eventually terminate by reaching base cases.

Comment: See comment to problem 1 (c).

Note that $V[1,N]$ is the final answer we are looking for. We proceed to bounding the running time of our memoization-based evaluation of the recurrence relation presented above. There is $O(N^2)$ subproblems $V[i,j]$ since there is at most that many pairs of indices (i,j) such that $1 \le i \le j \le N$. Computation of a single subproblem (assuming all the required subproblems are already computed) can be performed in $O(1)$ time. Therefore, the total running time is $O(N^2)$, as desired.