hw3

資工三 110590002 王熿竑

1.

1.a.

if maximum subarray contain $A[j+1]$
then because subarray is continuously
therefore maximum subarray is $A[i..j+1]$, for some $1 \leq i \leq j+1$
else if maximum subarray not contain A[j+1]
then maximum not change by $A[j+1]$,
that is maximum is same as $A[i..j]$ for some $1 \leq i \leq j$

1.b.

根據上面的觀察結果，可以知道當加入一個新的數字時，
最大的 subarray 要麼是包含這個數字，要麼是不包含這個數字 。
當包含新的數字時，可能會是之前的數字繼續連續加總，也可能是從新的數字開始連續加總。
所以用一個變數 Max 來記錄目前最大的 subarray 總和，
可以用一個變數 back 來記錄包含新的數字但不包含之前最大連續數列的連續加總，
用一個變數 between 來記錄從原本的最大總和數列之後直到目前的數字的連續加總。
當 back 的值小於 0 時，代表這個數字不會對最大總和數列有貢獻，所以 back 歸零。
當 back 的值大於 Max 時，代表這個數字對最大總和數列有貢獻，所以更新 Max 爲 back。
當 Max+between 的值大於 Max 時，代表這個數字對最大總和數列有貢獻，所以更新 Max 爲 Max+between。
當 Max+between 與 back 的值都大於 Max 時，取比較大的值更新 Max。

```
def maxSubArray(self,nums:List[int]) ->int:
        arr = []
        back =0
        Max = -inf
        between = 0
        for i in nums:
            back += i
            between += i
            m = max(back ,Max +between)
            if m > Max:
                Max = m
                between=0
                back = 0
            if back< 0:
                back =0
        return Max
```

2.

因爲 A 經過 x 的凸包到達 B 的兩條路徑（經過兩側到達另外一端），會把所有 x 都包起來
所以 A 經過 x 的凸包到達 B 的路徑會相等於 A 和 B 加上所有 x 的凸包

所以 the algorithm is :
把 A B 加入到所有點的集合內，然後取得凸包

在取得凸包以後，可以得知 A 經過所有 x 的點到達 B 的兩條路徑，
取短的路徑極爲最短路徑

3.

P is origin problem
Q is finding two closest number in sorted array problem
Sort the array of x in ascending order.
thus, reduct P to Q
then for each the sorted array and get the minimum difference between two adjacent numbers.
lower bound of Q is $\Omega(n)$
reduct time is $O(n\log(n))$ then the tight lower-bound of P is $O(n\log(n)) + \Omega(n)$