① a-) Let's first prove that

$$A[i,j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j]$$

$$\Rightarrow A[i,j] + A[k, j+1] \leq A[i, j+1] + A[k, j]$$

where $i < k$

prove it by induction

Base case $\boxed{k = i+1}$ as for the inductive step, $\boxed{k = i+n}$

We want to prove it for

$k+1 = i+n+1$ If we add the given to assumption

We get

$$= A[i,j] + A[k, j+1] \leq A[i, j+1] + A[k, j]$$

$$= A[k, j] + A[k+1, j+1] \leq A[k, j+1] + A[k+1, j]$$

$$\Rightarrow A[i,j] + A[k, j+1] + A[k, j] + A[k+1, j+1] \leq$$

$$A[i, j+1] + A[k, j] + A[k, j+1] + A[k+1, j]$$

$$\Rightarrow A[i, j] + A[k+1, j+1] \leq A[i, j+1] + A[k+1, j]$$

b-) $A[i, j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j]$

	0	1	2	3	4
0	10	17	13	28	23
1	17	22	16	29	23
2	24	28	22	34	24
3	11	13	6	17	7
4	45	44	32	37	23
5	36	33	19	21	6
6	75	66	51	53	34

$10 + 22 \leq 17 + 17$

$32 \leq 34$ ✓ special array

for this part we must change only element for example

32	23	22	32
21	6	7	10
53	34	30	31
32	13	9	6
63	21	15	8

$23 + 7 \leq 6 + 22$

$30 \leq 28$

is not special

So we can change 22 with 24

10	24
6	7

$23 + 7 \leq 6 + 24$

$\Rightarrow$ This is a special array

c-) We can think a X matrix We can construct a submatrix X of X consisting of the even numbred rows of X and we can recursively determine the leftmost minimum for each row of X. Then we can compute the leftmost minimum for each row X. Then we can compute the leftmost minimum in the odd-numbered rows of A.

If y_i is the index of the i-th row's leftmost minimum then we have

$$y_{i-1} \leq y_i \leq y_{i+1}.$$

for $i = 2k+1$, $k \geq 0$ finding y_i takes $y_{i+1} - y_{i-1} + 1$ steps at most, since we only need to compare with those numbers. Thus

$$T(m,n) = \sum_{i=0}^{m/2-1} (y_{2i+2} - y_{2i} + 1)$$

$$= \sum_{i=0}^{m/2-1} y_{2i+2} - \sum_{i=0}^{m/2-1} y_{2i} + m/2$$

$$= \sum_{i=1}^{m/2} y_{2i} - \sum_{i=0}^{m/2-1} y_{2i} + m/2$$

$$= y_m - y_0 + m/2$$

$$= n + m/2$$

$$= O(m+n)$$

d-) The divide time is $O(1)$, the conquer part is $T(m/2)$ and the merge part is $O(m+n)$. Thus

$$T(m) = T(m/2) + cn + dm$$

$$= cn + dm + cn + dm/2 + cn + dm/4 + \cdots$$

$$= \sum_{i=0}^{\log m - 1} cn + \sum_{i=0}^{\log m - 1} \frac{dm}{2^i}$$

$$= cn \log m + dm \sum_{i=0}^{\log m - 1}$$

$$< cn \log m + 2dm$$

$$= O(n \log m + m)$$

(2) We use divide and conquer method. Because if we make merge the array at first and find the kth element later Our algorithm is not efficient. By using a divide and conquer approach similar to the one used in binary search. We can attempt to find the kth element in a more efficient way.

We use follow algorithm :

Compare middle element of A and B let us call these indices midA and midB respectively.

Assume A[midA] then clearly element after midB cannot be required element.

We then set the last element of B to be B[midB]

finally we define a new subproblem with half the size of one of the arrays.

Time Complexity is O(log n + log m)

(3) We can solve this problem with naive method but it is to run two loop and time complexity is $O(n^2)$. So divide and conquer approach is more efficiently because time complexity is $O(n \log n)$ for divide and conquer approach.

maxSubArraySum() is a recursive method and time complexity can be expressed as following recurrence relation.

$$T(n) = 2T(n/2) + \theta(n)$$

Divide and Conquer method algorithm following:

→ Divide the given array in two halves

→ Return the maximum of following three

- maximum subarray sum in left half
- maximum subarray " " right half.
- " " " such that the subarray crosses the midpoint

find the maximum sum starting from midpoint and ending at some point on left of mid, then find the maximum sum starting from mid+1 and ending with sum point on right of mid+1. Finally combine the two and return.

Simple approach where we divide the array in two halves, reduces the time complexity from $O(n^2)$ to $O(n \log n)$.

Worst case of this algorithm is $O(n \log n)$

(4) We can use BFS algorithm for this problem. Because BFS based on decrease and conquer technique.

A bipartite graph is possible if the graph coloring is possible using two colors such that vertices in a set are colored with the same color.

Algorithm for this code :

-> We assign a color to the source vertex
-> Color all the neighbors with another color except first one color
-> Color all neighbor's neighbor with first color.
-> Assign color to all vertices such that it satisfies all the constraints of two coloring problem
-> If we find a neighbour which is colored with same color as curent vertex,

In this algorithm, each vertex of the graph needs to be traversed once, and each neighbor of a vertex is traversed once. Since we are using an adjacency matrix, this results worst time complexity case

$O(V^2)$

(5) This algorithm find maximum gain of days. We design algorithm with divide and conquer approach.

We check left and right of Price and Cost arrays.

We find middle element.

We use recursive approach for divide and conquer algorithm.

The algorithm worst case complexity $O(\log n + \log m)$