Q-1) According to question electricity can not can only be transferred until broken fuse and can not reach remaining fuses.

So input will be like that:

fuses = [1, 1, 0, 0, 0]

1: there is electric

0: there is no electric

After first zero remaining will be zero because there is not any electricity. Finding first zero will be enough to find broken fuse.

I used an algorithm like binary search. It checks middle element. Then if middle element is 1, it goes right half side of array. If middle element is 0, it goes left half side of array.

It divides two part for each recursion call. So this is decrease and conquer (decrease by a constant factor)

$T(n) = T(n/2) + \Theta(1)$

$a = 1 \quad a = b^d$
$b = 2 \quad 1 = 2^0$
$d = 0$

$T(n) = n^d \log n$

$T(n) = \Theta(\log n)$

Q-2) According to question pixels are monotonically increases or decreases. First my function detects if it is increasing or decreasing. Then for each row of image it compares two column. If normally it supposed to be decreased but it is increased (vice versa), it is brighter than it's neighbours. If everything is normal, it finds brightest one from corner.

To compare columns it calls function recursively. For each recursive call number of column remained will decrease 1. This decrease and conquer.

$T(n) = T(n-1) + \Theta(1) \Rightarrow$ reccurence relation

For the best case it finds in the second column

$T_b(n) = \Theta(1)$

For the worst case it finds the end of the row so it checks all indexes.

$T_w(n) = \Theta(n)$

$T_{Average}(n) = \Theta(n)$

Q-3) This algorithm sums up all elements until finding first negative element. After first negative element, it calls itself recursively and it sends rest of array as parameter. So it decrease size of input.

Basically it goes all the elements so it's time complexity is $T(n) = \Theta(n)$

$T(n) = T(n-m) + m * \Theta(1)$

Example

[2, 1, -4, 2, 1, 8, -44, 3, 11, -2, 15, 1]

3 7 -30 14

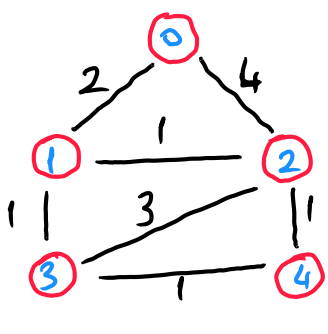
Backtrace of recursive

puts 14 to the temp

3+7 = 10

temp which is 14 is bigger than 10

Q-4) I used dfs algorithm to explore all possible paths. I used list to faciliate their mutable behaivor. While traversing all the vertices, it saves minimum weight (latency) of graph.



from 0 to 4

it finds

0, 1, 2, 4

latency = 4

It explores all the paths. It will traverse all unvisited neighbours for all path.

So $T(n) = O(V+E)$ where V represents the number of vertices and E represents the number of edges in the graph.

Q-5) To find Max and min resources and to apply divide and conquer at the same time I prefered to use merge sort. In the end of sort operation the task which requires least resources will be first element of tasks array and the task which requires most resources will be last element of tasks array.

$T(n) = 2T(n/2) + O(n)$

$T(n) \in O(n \log n)$