Q1.

a.

Call v is a value array in every week.

Initialize v array is 0 value.

With week i=1: v[i] = max(l[1], h[1])

Formulation from week i=2, With week i>=2:

v[i] = max(v[i-1]+l[i], v[i-2]+h[i])

b.

Set array v is 0 value.

v[1] = max(l[1], h[1])

Loop from i = 2 to n:

v[i] = max(v[i-1]+l[i], v[i-2]+h[i])

print v[n]

v[n] is the maximum value.

Q2.

a. Consider n=3

xi: 10 10 4

fi: 2 12 13

In this case, according the example algorithm:

smallest number at xn: j=2, activate EMP in the 3rd second,

Continue recursive, n-j=1, no smallest number at x1 then j=1, activate EMP in the 1st second

total of robots destroyed: min(f1, x1) + min(f2, x3) = 2+4=6

This is wrong answer,

The optimal answer is: activate EMP in 2nd and 3rd second, total:

min(f2, x2) + min(f1, x3) = 10+2 = 12

b.

desrobot is the array store number robots destroyed.

At i-th second, since the last EMP activated at k-th second:

desrobot[i] = max(min(f[i], x[i]), desrobot[i-k]+min(f[i-k], x[i]))

Algorithm:

Initialize array desrobot = 0

f[0]=0; x[0]=0

For i=1 to n:

tempmax = 0

For j=1 to i:

tempmax = max(tempmax, desrobot[i-j]+min(f[i-j], x[i]))

desrobot[i] = max(min(f[i], x[i]), tempmax)

print desrobot[n]

Q3.

Build a graph with the nodes are the corresponding companies, directed edge (i, j) weight r(i, j)

Trading cycle is an opportunity cycle if r(i, j) > 1 with all edges (i, j).

Use Bellman-Ford algorithm to detect whether exist a negative cycle (opportunity cycle).

Bellman-Ford algorithm runs in polynomial-time O(VE)

Q4.

a. With i > 0, cost to do multiply Ai\*Ai+1 = x[i-1]\*x[i]\*x[i+1]

Ai\*Ai+1 \*Ai+2 = x[i-1]\*x[i]\*x[i+1] + x[i-1]\*x[i+1]\*x[i+2]

Assume cost array stores best cost of multiply from i→j (i>0,j>i), calculation Ai\*Ai+1\*...\*Aj

Formulation:

cost[i,j] = min(cost[i,k]+cost[k+1,j]+x[i-1]\*x[k]\*x[j]) with k from i→j-1

b.

Best cost of calculation A1\*A2\*...\*An : cost[1,n]

Algorithm:

Initialize array cost with 0 value, size n+1

For i=2 to n-1:

For j=1 to n-i

cost[j, i+j-1] = MAXINT

For k=j to j+i-2

cost[j, i+j-1] = min(cost[j,k]+cost[k+1,i+j-1]+x[j-1]\*x[k]\*x[i+j-1], cost[j, i+j-1])

print cost[1,n]