(a) Change x_4 to 2 in the given example. Then this algorithm would activate the EMP at times 2 and 4, for a total of 4 destroyed; but activating at times 3 and 4 as before still gets 5.

(b) Let $OPT(j)$ be the maximum number of robots that can be destroyed for the instance of the problem just on $x_1, \ldots, x_j$. Clearly if the input ends at x_j, there is no reason not to activate the EMP then (you're not saving it for anything), so the choice is just when to last activate it before step j. Thus $OPT(j)$ is the best of these choices over all i:

$$OPT(j) = \max_{0 \le i < j} [OPT(i) + \min(x_j, f(j-i))],$$

where $OPT(0) = 0$. The full algorithm is just

```
Set OPT(0) = 0
For i = 1,2,...,n
  Compute OPT(j) using the recurrence
Endfor
Return OPT(n).
```

The running time is $O(n)$ per iteration, for a total of $O(n^2)$.

An alternate solution would define $OPT'(j,k)$ to be the best solution for steps j through n, given that the EMP in step j has already been charging for k steps. The optimal way to solve this sub-problem would be to either activate the EMP in step j or not, and $OPT'(j,k)$ is just the better of these two choices:

$$OPT'(j,k) = \max(\min(x_j, f(k)) + OPT'(j+1,1), OPT'(j+1,k+1)).$$

We initialize $OPT'(n,k) = \min(x_n, f(k))$ for all k, and the full algorithm is

```
Set OPT'(n,k) = min(x_n, f(k)) for all k.
For j = n-1, n-2, ..., 1
  For k = 1,2,...,j
    Compute OPT'(j,k) using the recurrence
  Endfor
Endfor
Return OPT'(1,1).
```

The running time is $O(1)$ per entry of OPT', for a total of $O(n^2)$.

[1] ex249.233.474