# *Save Energy - Editorial*

## Difficulty:

Medium

## Prerequisites:

Dynamic Programming - (Tutorial)
Greedy Algorithm - (Tutorial)

## Problem in Brief:

There are N towns in a line. Distance between town i and j is abs(j-i). We want to go from town 0 to town N-1. From town i, we can jump to any town j > i and that costs $(j-i)A[i] + (j^2 - i^2)A[i]^2$, where A is given as input. We want to find the minimum cost needed. N <= 10^5.

## Editorial:

There are 2 solutions. First solution using Greedy approach:

If you write cost function for path i->j->k and compare it with the cost function for path i->k you can see that the former is better (costs less energy) if and only if either |A[j]| < |A[i]| or |A[j]| = |A[i]| with A[i]>0.

From this observation it is quite easy to see that the optimal path would be greedy jumping from 0 to first i such that |A[i]| < |A[0]| or A[i] = -A[0] (if A[0]>0) or directly to n-1 if there is no such i and then doing the same for i till we reach n - 1. Jumping greedily to n-1 solves the problem with O(N) complexity.

# *Save Energy - Editorial*

Second solution using Dynamic Programming:

If we look at the nature of cost function it is easy to see that we can split each jump into consecutive steps, we can also solve it in more straightforward way — by computing a DP[i][j] which will tell us smallest cost to reach position i while using some A[x] = j. Now from position i we can either keep moving forward using same "old" value of A, or start using A[i]. Such solution works in O(N * 10^3) where 10^3 is count of distinct possible values of A[i].

## Time Complexity:

**O(N)**

## Similar Problems:

First

Second