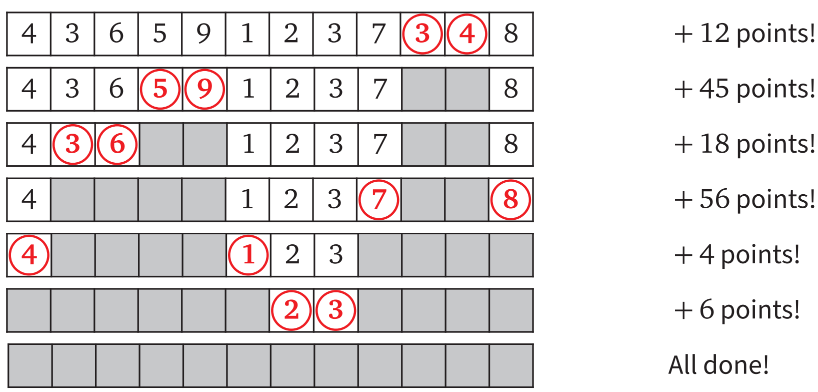
1. Prove that every integer (positive, negative, or zero) can be written in the form $\sum_i \pm 3^i$, where the exponents i are distinct non-negative integers. For example:

$$\begin{aligned} 42 &= 3^4 - 3^3 - 3^2 - 3^1 \\ 25 &= 3^3 - 3^1 + 3^0 \\ 17 &= 3^3 - 3^2 - 3^0 \end{aligned}$$

2. After being bombarded for months by TikTok ads, you finally break down and download the latest viral mobile puzzle game Number Blast.

   Each Number Blast puzzle begins with a long row of numbered squares, of even length. On each turn, you choose two arbitrary squares *with no other numbered squares between them*. You then earn points equal to the *product* of your two chosen numbers, and both chosen squares are removed. Once a square is removed, it cannot be chosen in any future turns. Your goal is to remove *all* of the numbers and to earn as many points as possible.

   For example, the following sequence of turns earns a total of 141 points. (This is not necessarily the highest possible score for this sequence of numbers.)



   Describe and analyze an algorithm to find the maximum number of points you can earn in a single Number Blast puzzle. The input to your algorithm is an array $A[1..2n]$ of positive integers. *[Hint: Consider your last turn first.]*

Questions 3 and 4 are on the back of this page.

3. Suppose we are given a bit string $B[1..n]$.

   (a) A ***well-spaced triple*** in B is a set of three ***distinct*** indices $\{i, j, k\}$ such that $B[i] = B[j] = B[k] = 1$ and $k - j = j - i$. Describe an algorithm to determine the number of well-spaced triples in B.

   (b) An ***offset triple*** in B is a set of three ***distinct*** indices $\{i, j, k\}$ such that $B[i] = B[j] = B[k] = 1$ and $k - j = \mathbf{2} \cdot (j - i)$. Describe an algorithm to determine the number of offset triples in B.

   For full credit, both algorithms should run in $O(n \log n)$ time.

   For example, given the bitstring $B =$ `10101000101` as input, your algorithm for part (a) should return 2, for the well-spaced triples $\{1, 3, 5\}$ and $\{1, 5, 9\}$, and your algorithm for part (b) should return 2, for the offset triples $\{3, 5, 9\}$ and $\{11, 9, 5\}$.

4. Suppose you are given a sequence of n words that you want to wrap in a nice paragraph, breaking lines between words. Adjacent words on the same line must be separated by exactly one space. Each line has space for M characters. The *badness* of a single line is given by a function *badness*(x), where x is the number of spaces at the end of the line, and the *total badness* of the paragraph is the sum of the badnesses of its lines. You want to find a layout with minimum total badness.

   For example, the total badness of the following paragraph is *badness*(6) + *badness*(2) + *badness*(0) + *badness*(8). The dots at the end of each line indicate trailing spaces. (This is not necessarily the best layout for this example.)

```
Never forget that it is a waste of······
energy to do the same thing twice, and··
that if you know precisely what is to be
done, you need not do it at all.········
```

   Describe an algorithm that finds the smallest possible total badness for a given sequence of words. The input to your algorithm consists of the positive integer M and an array $L[1..n]$, where $L[i]$ is the length of the ith word. You have access to a subroutine BADNESS(x) that computes *badness*(x) in $O(1)$ time. Assume that $L[i] \le M$ for all i and that *badness*$(x) > 0$ for all x. (Yes, even when $x = 0$.)