Minkowski Sum & Well-spaced triples

Fast Fourier Transform and using it to speed up polynomial multiplication is the topic of the two problems[1] in this note.

[1] Jeff Erickson. Algorithms — Extended Dance Remix: Fast Fourier Transforms. https://jeffe.cs.illinois.edu/teaching/algorithms/notes/A-fft.pdf, 2021. [Online; accessed 07-May-2022]

Problem

Given two sets of integers $X \subset \mathbb{Z}$ and $Y \subset \mathbb{Z}$, compute the size of the Minkowski sum: $X + Y = \{x + y : x \in X, y \in Y\}$ in $O(n \log n)$ time.

Solution. A pretty straightforward way of calculating the Minkowski sum is to generate all possible pairs (that is a nested loop, so $O(n^2)$) and then also making sure the resulting values form a set, so only occur once. This can be achieved by storing the values as we go in a balanced binary search tree. For each value the cost would be $O(\log n)$, so the straightforward solution has a runtime of $O(n^2 \log n)$ if using a binary search tree or $O(n^2)$ if using a hashtable.

Can we do better? This problem is an exercise in the FFT chapter of Jeff Erickson's Algorithms book. So the answer is: yes we can. To do so, we remember that multiplying two polynomials given in coefficient form can be done in $O(n \log n)$ using the Fast Fourier Transform.

But what polynomials should we consider? Let's explore polynomial multiplication with a couple of examples:

$$\begin{aligned}(1 + x)(x^2 + x^5) &= 1(x^2 + x^5) + x(x^2 + x^5) \\ &= x^2 + x^5 + x^3 + x^6\end{aligned}$$

$$\begin{aligned}(2x + x^4)(1 + 3x^3) &= 2x(1 + 3x^3) + x^4(1 + 3x^3) \\ &= 2x + 6x^4 + x^4 + 3x^7\end{aligned}$$

Notice how we multiplied each monomial[2] from the first polyno-

[2] A monomial is an individual term of a polynomial A polynomial is a sum of monomials. In our example the monomials of $1 + x$ are 1 and x. The monomials of $x^2 + x^5$ are x^2 and x^5.

mial with each mononial from the second polynomial. In the second example it is also visible that we haven't yet collected together all the monomials of degree four of the multiplication result to better demonstrate that each monomial from the first polynomial is multiplied with each monomial from the second polynomial.

When we multiply two monomials, their coefficients get multiplied and their exponents get added. Each monomial from one polynomial is paired with each monomial from the other polynomial in an operation (multiplication) and in that operation the exponents are added. This strongly suggests[3] that we should define a polynomial from one of the given sets of integers by making the members of the set be the exponents of the monomials that form the polynomial.

For example: if $X = \{2, 5, 6\}$ then the corresponding polynomial could be $x^2 + x^5 + x^6$. The polynomial coefficients have been arbitrarily chosen to all be one [4].

Thus we define the two polynomials $p_X(x)$ and $p_Y(x)$ from the given integer sets X and Y:

$$p_X(x) = \sum_{i \in X} x^i$$
$$p_Y(x) = \sum_{j \in Y} x^j$$

and we multiply them

$$p_X(x) p_Y(x) = \sum_{i \in X} \sum_{j \in Y} x^{i+j}$$

The exponents of the monomials of the polynomial product are the members of the Minkowski sum $X + Y$. The size of $X + Y$ is the number of monomials [5].

This gives us a way to calculate the size of $X + Y$ in $O(n \log n)$ because we can do the polynomial multiplication using FFT in $O(n \log n)$. □

[3] In the problem each member of the first set is paired with each member of the second set in an operation (addition).

[4] There is one small wrinkle in this scheme. In the problem the given sets are sets of integers, so they can be negative. We cannot have monomials with negative exponents. For now let us assume X and Y only contain non-negative integers with the promise that at the end of this solution we will address how to drop this assumption.

[5] As promised: what can we do when X and Y contain negative integers. Let $d > 0$ be a constant such that both $d + X = \{d + a : a \in X\}$ and $d + Y = \{d + b : b \in Y\}$ have only non-negative members.

We define the polynomials slightly differently:

$$p_X(x) = \sum_{i \in X} x^{i+d}$$
$$p_Y(x) = \sum_{j \in Y} x^{j+d}$$

and we multiply them

$$p_X(x) p_Y(x) = \sum_{i \in X} \sum_{j \in Y} x^{i+j+2d}$$

Again, the size of $X + Y$ is the number of monomials.

Problem

Given is a bit string $B[1 \ldots n]$. A well-spaced triple is a triple (i, j, k) of indices such that $1 \leq i < j < k \leq n$ and $B[i] = B[j] = B[k] = 1$. Detect in $O(n \log n)$ time if bit string B contains a well-spaced triple.

Solution. Again the brute-force solution is easy to describe: for each middle index in the triple we consider all possible distances to left

and right indices and check the condition. This is a nested loop with runtime $O(n^2)$.

We will try to improve on the brute-force by again employing polynomial multiplication even though it is not an obvious choice. After all we are only given one bit string, not two bit strings and multiplication needs two operands. But maybe we can derive a polynomial from the bit string and then square that polynomial which would be a polynomial multiplication.

Let's explore what would happen if we use the given bit string B directly as a coefficient vector of a polynomial

$$p_B(x) = \sum_{i=0}^{n-1} B[i+1]x^i$$

Squaring $p_B(x)$ we get

$$\begin{aligned}(p_B(x))^2 &= p_B(x)p_B(x) \\ &= (\sum_{i=0}^{n-1} B[i+1]x^i)(\sum_{j=0}^{n-1} B[j+1]x^j) \\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} B[i+1]B[j+1]x^{i+j}\end{aligned}$$

For all $0 \leq j < n$ monomial x^j is present in $p_B(x)$ if $B[j+1] = 1$. If monomial x^j is present in $p_B(x)$ then monomial x^{2j} is present in $(p_B(x))^2$. This is because monomial x^j pairs with itself in the polynomial multiplication. So the coefficient of monomial x^{2j} in $(p_B(x))^2$ is at least one (namely $B[j+1]B[j+1] = 1$). Can it be larger than one? It would mean that some other monomial pairing has exponent sum equal to $2j$. Let i and k be the two indices for which $B[i+1] = 1$, $B[k+1] = 1$ and $i + k = 2j$. But $i + k = 2j$ is equivalent to $k - j = j - i$ which means that (i, j, k) form a well-spaced triple. This (i, k) monomial pairing appears twice in the polynomial multiplication (once for monomial x^i from the left and monomial x^k from the right and once reversed with monomial x^k from the left and monomial x^i from the right). That means that the well-spaced triple contributes the value two to the coefficient of x^{2j} in $(p_B(x))^2$ and we have found our criteria for detecting well-spaced triples: if $(p_B(x))^2$ has any monomials with even degree and coefficients greater or equal to three, then B has a well-spaced triple.

This gives us a way to detect well-spaced triples in $O(n \log n)$ because we can do the polynomial multiplication using FFT in $O(n \log n)$.

□

Bibliography

Jeff Erickson. Algorithms — Extended Dance Remix: Fast Fourier Transforms. https://jeffe.cs.illinois.edu/teaching/algorithms/notes/A-fft.pdf, 2021. [Online; accessed 07-May-2022].