Class Notes
CIS 502 Analysis of Algorithm
2-Divide and Conquer

Da Kuang

University of Pennsylvania

This week we talked about two more example about divide and conquer:

- Polynomial Multiplication
- Convex Hulls on the Plane

1 Polynomial Multiplication

1.1 Introduction

A polynomial in the variable x over an algebraic field F represents a function $A(x)$ as a formal sum:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

We call the values a_0, a_1, $\cdots$, a_{n-1} the **coefficients** of the polynomial. The coefficients are drawn from a field F , typically the set $\mathbb{C}$ of complex numbers. A polynomial can be represented as a list of coefficients in computer.

A polynomial $A(x)$ has **degree** k if its highest nonzero coefficient is a_k. we write that degree$(A) = k$. Any integer strictly greater than the degree of a polynomial is a **degree-bound** of that polynomial. Therefore, the degree of a polynomial of degree-bound n may be any integer between 0 and $n - 1$, inclusive.

Polynomial Multiplication: if $A(x)$ and $B(x)$ are polynomials of degree-bound n, their product $C(x)$ is a

polynomial of degree-bound $2n-1$ such that $C(x) = A(x)B(x)$ for all x in the underlying field. For example,

$$\begin{array}{rrrrrrrrrr}
 & & & & & 6x^3 & + & 7x^2 & - 10x & + & 9 \\
 & & & & & - & 2x^3 & & + & 4x & - & 5 \\
\hline
 & & & & & - 30x^3 & - & 35x^2 & + & 50x & - & 45 \\
 & & & & 24x^4 & + 28x^3 & - & 40x^2 & + & 36x & & \\
 & & - 12x^6 & - 14x^5 & + 20x^4 & - 18x^3 & & & & & & \\
\hline
 & & - 12x^6 & - 14x^5 & + 44x^4 & - 20x^3 & - & 75x^2 & + & 86x & - & 45
\end{array}$$

Another way to express the product $C(x)$ is

$$C(x) = \sum_{j=0}^{2n-2} c_j x^j$$

where

$$c_j = \sum_{k=0}^{j} a_k b^{j-k}$$

Note that degree(C) = degree(A) + degree(B), implying that if A is a polynomial of degree-bound n_a and B is a polynomial of degree-bound n_b , then C is a polynomial of degree-bound $n_a + n_b - 1$. Since a polynomial of degree-bound k is also a polynomial of degree-bound $k + 1$, we will normally say that the product polynomial C is a polynomial of degree-bound $n_a + n_b$.

1.2 Algorithm

Input: Two polynomials with same degree d:

$$A = a_d x^d + \cdots + a_0$$
$$B = b_d x^d + \cdots + b_0$$

Goal: Calculate the product of the inputs.

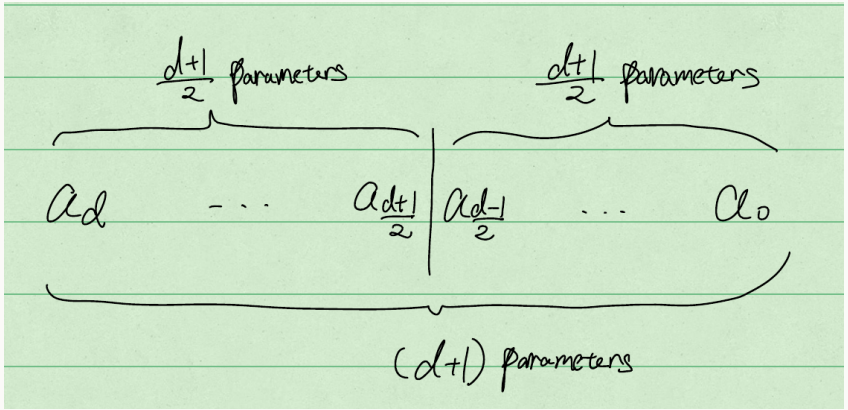
1.2.1 Baseline Method

Simply calculate each parameter based on the definition of polynomial product. Each parameter in $A(x)$ should multiply all the parameters in $B(x)$ which takes $O(n^2)$ steps.

1.2.2 Divide and Conquer

Assumption: To make sure the parameters could be nicely divided by two, we assume the degree d of both polynomials is always $2^k - 1$, where $k \in \mathbb{N}$.

Divide the $(d + 1)$ entries of polynomials as follows to reduce the problem size:



Call the right hand side of polynomial as A_l. Extract $x^{\frac{d+1}{2}}$ and call the left hand side of polynomial as $x^{\frac{d+1}{2}}A_h$. we have,

$$A = x^{\frac{d+1}{2}}A_h + A_l$$
$$B = x^{\frac{d+1}{2}}B_h + B_l$$

Based on the algebra,

$$\begin{aligned} A \times B &= (x^{\frac{d+1}{2}}A_h + A_l)(x^{\frac{d+1}{2}}B_h + B_l) \\ &= x^{d+1}A_hB_h + x^{\frac{d+1}{2}}A_hB_l + x^{\frac{d+1}{2}}A_lB_h + A_lB_l \end{aligned}$$

Therefore, we reduce the problem into four sub-problems with half of the size. Suppose the number of parameters is n, i.e. the degree of polynomials is $d = n - 1$. The recursion equation as follows:

$$T(n) \leq 4T(\frac{n}{2}) + cn$$

Based on master theorem, $T(n) = O(n^2)$. Unfortunately, we have not get any improvement from those nicely splits comparing with the baseline. If you think about it, the result is actually not surprising because even though the problem size is reduced, we could not skip any of the necessary multiplication. In other words, there are still n^2 necessary multiplications to get the production.

Motivation: When you have a recursive algorithm, saving one operation could have a cascade effect and reduce the run time quickly. Adding polynomials only costs $O(n)$ steps. If we could reduce multiplications by introducing addition. Then we are able to achieve better performance.

To reduce the multiplications by adding additions, we have

$$\begin{aligned} A \times B &= x^{d+1} A_h B_h + x^{\frac{d+1}{2}} A_h B_l + x^{\frac{d+1}{2}} A_l B_h + A_l B_l \\ &= x^{d+1} A_h B_h + x^{\frac{d+1}{2}} (A_h B_l + A_l B_h) + A_l B_l \end{aligned}$$

Therefore, there are three items to calculate

(1) $A_h B_h$

(2) $A_l B_l$

(3) $A_h B_l + A_l B_h$

(1) and (2) are two multiplications. Moreover, (3) can be calculate by one more multiplication as following:

$$\begin{aligned}&(A_h + A_l)(B_h + B_l) - A_hB_h - A_lB_l\\ =&A_hB_h + A_hB_l + A_lB_h + A_lB_l - A_hB_h - A_lB_l\\ =&A_lB_h + A_lB_l\end{aligned}$$

Therefore, the recursive function becomes

$$T(n) \leq 3T(\frac{n}{2}) + cn$$

It can be solved by master theorem, where $a = 3, b = 2, c = 1$.

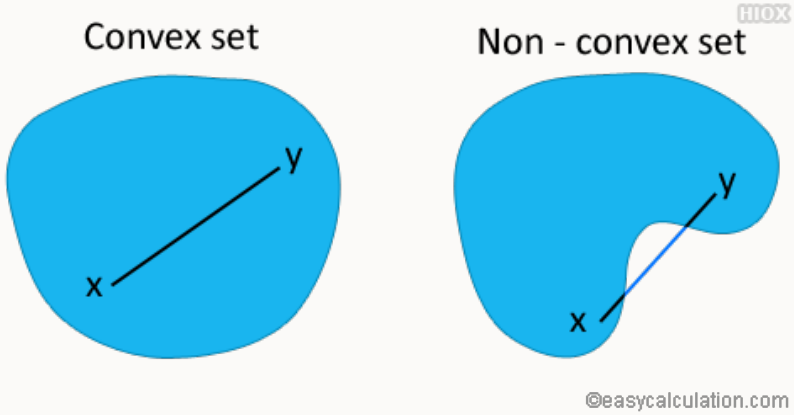
Since $\log_b a > 1 = c$, so $T(n) = O(n^{\log_2 3}) \approx O(n^{1.6})$

2 Convex Hull on the Plane

2.1 Introduction

2.1.1 Convex set

Convex set A set $S \subseteq \mathbb{R}^n$ is said to be convex if $\forall x, y \in S$, the line segment joining x and y is contained in S.



2.1.2 Convex Combination

Given two points $x = (x_1, x_2)$ and $y = (y_1, y_2) \in \mathbb{R}^2$, a convex combination of x and y is defined by

$$\lambda x + (1 - \lambda)y = \lambda x_1 + (1 - \lambda)y_1 + \lambda x_2 + (1 - \lambda)y_2, \text{ for } \lambda \in [0, 1]$$

Changing the value of λ, we walk thought the segment. In general, is the area confined by the edges formed by points.

Example Consider a practical example, suppose there are four wells (w_1, w_2, w_3, w_4) to generate petroleum and the composition (A, B, C) of petroleum from each well are different as follows:

		A	B	C
λ_1	w_1	0.4	0.4	0.2
λ_2	w_2	0.6	0.3	0.1
λ_3	w_3	0.3	0.4	0.3
λ_4	w_4	0.2	0.7	0.1

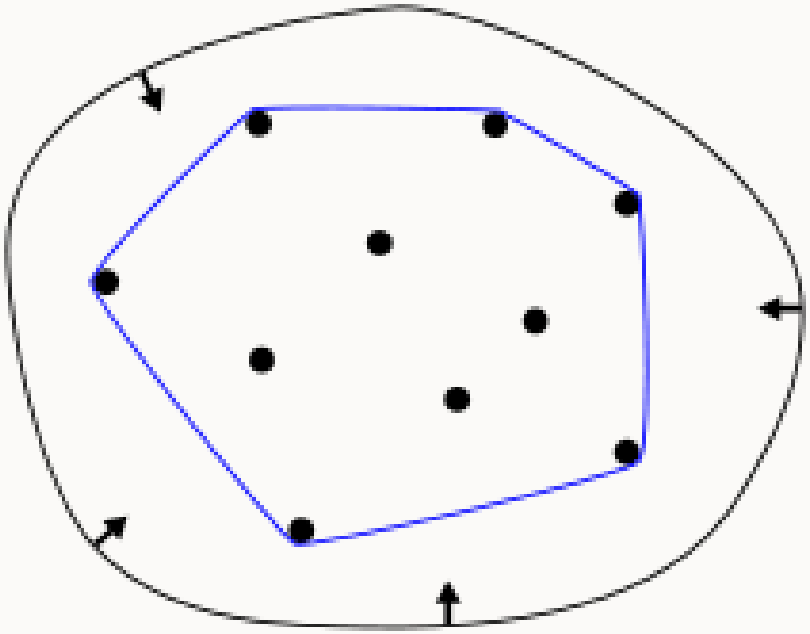
Therefore, all the possible compositions from the mixture of the four kinds of petroleum is actually a convex combination of the four petroleum. The four wells are like four points in a 3-D space and the convex set is the object whose vertices are those four points.

2.1.3 Convex Hull

Definition 1: Convex hull of a finite set of points S is the set of all points that can be expressed as convex combinations of points in S.

Definition 2: Convex hull of a set of points is the smallest convex set that contains these points.

More intuitively, convex hull is like a rubber band stretched from infinite and struck on the nails on points.



2.1.4 Convex polygon

Convex polygon is a polygon which is a convex set. It always contains interior angle which smaller than 180 degree.

2.2 Algorithm

- Input: n points, $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$.

- Output: The sequence of points on the boundary in counter clock order.

2.2.1 Relative Location of Line and Points

Given a line by two points $(x_1, y_1), (x_2, y_2)$, a joining segment can be represented as follows:

$$y - y_1 = \frac{y_1 - y_2}{x_1 - x_2}(x - x_1)$$

For any third point (x_3, y_3), it is possible determine which side of the line the third point lies. For instance, the thirds point is above the segment if

$$y_3 - y_1 > \frac{y_1 - y_2}{x_1 - x_2}(x_3 - x_1)$$

2.2.2 Convex Hull Successive Points Lemma

- If there is a segment joining two points such that all the other points lie on the same side of the segment, the segment represents two points on the convex hull.
- Conversely, if you have a segment on the convex hull and extending it, then every point from the convex hull will lie on the same side of the line.

2.2.3 Baseline

1. Use the extreme point as a start x.

2. Take every pair of points with x. write the equation for the line joining them and find the line where all other the points lie on the same side.
3. The line is a segment of convex hull.
4. Set x to the other point of the segment and then repeat step 2.

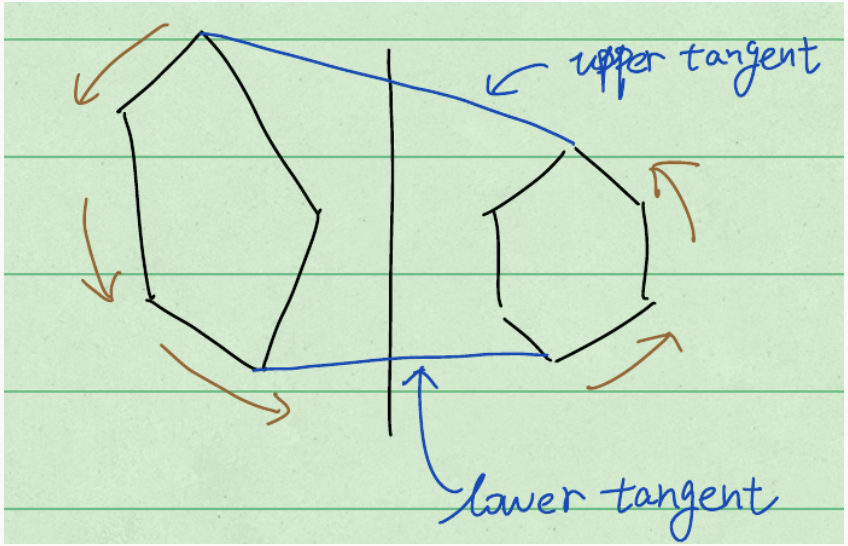
Runing Time Analysis

- Determine all the points are on the same side of segment takes $O(n)$ steps.
- To find one convex hull segment, we should check $O(n)$ segments.
- There are at most $O(n)$ convex hull segment.
- Therefore, the total run time is $O(n^3)$.

2.2.4 Merge Hull

Tips: To be efficient, we could like the sizes of sub-problems in divide and conquer to be same.

1. Divide points into those with x-coordinates $\leq x_{median}$ and those with x-coordinates $> x_{mdian}$.
2. Recursively find the convex hull for the points on the left and right.
3. Find the upper tangent and lower tangent of the two convex hull.
4. Start from the left of upper tangent and walk through the left convex hull counterclockwise until meet the lower tangent. Then move to the right convex hull and keep walking until back to the upper tangent.



Remarks:

- Points on one hull can "**see**" a point on the other hull if the segment joining the two points does not intersect either hull.
- Upper tangent, lower tangent are pairs of points that see each other.
- **Upper tangent** is the highest line segment joining two vertices, one from each hull, that see each other.
- Similar with the lower tangent. Therefore the total running time is $O(n)$.

Stab A line pq stabs to C_2 if the continuation of the line goes to the interior of C_2.

How to check stabbing

- If the segment pq stabs C_2, then the clockwise predecessor of q is on the one side of the segment and the clockwise successor of q is one the other side.
- If pq stabs C_2 then p must be able to see the clockwise predecessor and successor of q.

How to find the upper tangent: The upper tangent can be found by a "swirling" algorithm.

- Choose p to be the right most point in the left hull (C_1) and q to be the left most point in the right hull (C_2). Then p and q see each other.
- If pq stabs to C_1, move p to its counterclockwise successor.
- If pq stabs to C_2, move q to its clockwise successor.
- repeat until pq does not stab to any hull. Then pq is the upper tangent.

Lower tangent can be find with the similar strategy.

Running Time Analysis We never repeat the same pair of vertices since once you decide to abandoned a vertex in searching of upper tangent, you never come back to it. It means at most n vertices to find the upper tangent.

Similarly for the lower tanget.

2.2.5 Quick Hull

Like QuickSort, Quick Hull does not have worst case analysis.

- Pick the left most point p and right most point q on the convex hull and draw a segment joining them.
- Recursively find the convex hull for each side of the points. Each hull must contains the segment pq.
- Merge the two hull by glue them on segmeng pq.

We did not spend too much time on how to find the next two pair of points at each iteration. But here are some remarks.

Remarks

- Always pick points on x direction my casue the worst case, which is all the points are aggregated to one side of line.
- Like quicksort, introducing randomness helps achieve the optimal average running time. We randomly choose a direction and project all the points on it. Pick the two extremes as the points for the next iteration.