Ứng dụng của "Đa thức nội suy Lagrange" trong tin học (Lagrange Interpolation Polynomial)

1. Phát biểu công thức nội suy Lagrange

Nếu $a_0,\ a_1, \dots,\ a_n$ là $n+1$ số thực dương đôi một khác nhau, và $b_0,\ b_2, \dots,\ b_n$ là $n+1$ số thực bất kì và thỏa mãn:

$$F(a_0) = b_0,\ F(a_1) = b_1, \dots,\ F(a_{n-1}) = b_{n-1}, F(a_n) = b_n.$$

Thì đa thức $F(x)$ có bậc bé thua hoặc bằng n sẽ được xác định là:

$$F(x) = \sum_{i=0}^{n} (b_i \times \prod_{j=0 \to n}^{j \neq i} \frac{x - a_j}{a_i - a_j})$$

(Việc chứng minh các bạn có thể tìm hiểu trên mạng).

2. Ứng dụng trong tin học

Khi gặp các bài tin học mang chất "Toán", chúng ta thường phải mày mò tìm cách đưa ra công thức từ các bài toán đơn giản đến phức tạp cũng chỉ có cách duy nhất là lần mò công thức. Ví dụ bài toán tính tổng $(1 + 2 + \cdots + N)$ thì chúng ta sẽ đưa ra công thức là $\frac{N \times (N+1)}{2}$, nâng cao hơn một chút có thể là bài toán tính tổng $(1^2 + 2^2 + \cdots + N^2)$ thì cũng rất phổ biến bởi công thức $\frac{N \times (N+1) \times (2N+1)}{6}$, hay chúng ta cũng có thể sử dụng qui nạp với câu hỏi đưa ra cách tính tổng quát cho $(1^3 + 2^3 + \cdots + N^3)$ chính là $\frac{N^2 \times (N+1)^2}{4}$ …

Nhưng, nó thực sự khó nếu đưa ra cách tính cho $(1^k + 2^k + \cdots + N^k)$ bất kì, công thức tổng quát không phải không tồn tại nhưng có lẽ nó sẽ rất phức tạp và chúng ta không thể đi tìm công thức đó trong thời gian ngắn được!

Dễ thấy hàm $F(N) = 1 + 2 + \cdots + N = \frac{N \times (N+1)}{2} = \frac{1}{2}N^2 + \frac{1}{2}N.$

Hay $G(N) = 1^2 + 2^2 + \cdots + N^2 = \frac{N \times (N+1) \times (2N+1)}{6} = \frac{1}{3}N^3 + \frac{1}{2}N^2 + \frac{1}{6}N.$

Và $P(N) = 1^3 + 2^3 + \cdots + N^3 = \frac{N^2 \times (N+1)^2}{4} = \frac{1}{4}N^4 + \frac{1}{2}N^3 + \frac{1}{4}N^2.$

Như vậy ta nhận ra được rằng công thức tổng quát của hàm $F(N), G(N), P(N)$ là các đa thức với bậc lớn hơn 1 so với số mũ của các số hạng trong hàm.

Từ đây ta có cơ sở để kết luận rằng: ***hàm*** $\boldsymbol{Q(N) = (1^k + 2^k + \cdots + N^k)}$ ***là một đa thức bậc*** $\boldsymbol{(k+1)}$.

Trong tin học, chúng ta không nặng nề việc phải chứng minh những nhận xét mà chúng ta linh cảm rằng nó là đúng, chúng ta có thể làm theo những gì cảm nhận được và hoàn toàn có thể kiểm tra xem nó đúng hay sai, và đó là cách làm tốt nhất khi không có phương án nào tối ưu hơn nữa. Trong trường hợp này thì nhận xét đó là đúng, do đó chúng ta sẽ áp dụng *Đa thức nội suy Lagrange* để giải quyết tính hàm $Q(N)$ với phương án tốt nhất.

Thuật toán như sau:

Bước 1: Chúng ta sẽ chạy "trâu" để tính các $Q(1), Q(2), \ldots, Q(k+2)$ với độ phức tạp $O(k * \log(k))$ để có $k+2$ cặp số $(a_i, b_i) = (i, Q(i))$.

Bước 2: Áp dụng công thức Lagrange để tính:

$$Q(N) = \sum_{i=1}^{k+2} (Q(i) \times \prod_{j=1 \to k+2}^{j \neq i} \frac{N-j}{i-j})$$

Nhận xét: Dường như ở đây N không ảnh hưởng đến độ phức tạp thuật toán! Độ phức tạp cho bài toán này thông thường sẽ là $O(k^2)$ để áp dụng tính $Q(N)$ tuy nhiên chúng ta có thể biến đổi thành:

$$Q(N) = \sum_{i=1}^{k+2} (Q(i) \times \frac{L[i-1] * R[i+1]}{(-1)^{k+2-i} * (i-1)! * (k+2-i)!})$$

Trong đó $L[i] = \prod_{j=1}^{i}(n-i)$ và $R[i] = \prod_{j=i}^{k+2}(n-i)$.

Với công thức biến đổi ở trên thì chúng ta hoàn toàn có thể chuẩn bị trước và tính ra $Q(N)$ trong độ phức tạp $O(k * \log(k))$ dưới trường modulo là số nguyên tố (tùy theo dạng modulo để đưa ra cách giải hợp lý nhất).

Như vậy, bài toán tính $Q(N) = (1^k + 2^k + \cdots + N^k)\ \%\ P$ (P là số nguyên tố) nếu giải bằng *Đa thức nội suy Lagrange* sẽ có đpt $O(k * \log(k))$ và tôi nghĩ rằng đây là thuật toán tối ưu nhất! Lợi thế hơn rất nhiều so với cách tính thông thường có đpt $O(N * \log(k))$ hay nhân ma trận có đpt $O(k^3 * \log(N))$…

Ứng dụng này có thể coi là một ý tưởng mới và áp dụng được cho nhiều bài toán. Tương tự chúng ta có thể có các bài toán:

- Tính :

$$F(n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{t=1}^{n}|i-j| * |j-k| * |k-t| * |t-i|$$

  Thông thường, bài toán này chúng ta sẽ for "trâu" để tính với đpt $O(n^4)$, và nếu cải tiến thêm thì cũng chỉ tốt hơn là $O(n^3)$. Và điều đó lại phải phụ thuộc vào n. Tuy nhiên, thực ra hàm $F(n)$ này chính là đa thức bậc 8 khi mình tìm được công thức tổng quát cho nó, và khi có ý tưởng nó là một đa thức bậc 8 rồi thì chúng ta chỉ việc tính $F(1) \to F(9)$ rồi áp dụng *Đa thức nội suy Lagrange* để tính với đpt $O(9 * \log(9))$, một con số quá nhỏ so với máy tính, một điều hiển nhiên là nó không bị phụ thuộc vào n nữa!

- Hoặc, tính:

$$G(n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{t=1}^{n}|i-j| * |j-2k| * (k^2 - t^3) * (t + i^4).$$

  Một biểu thức loằng ngằng và nó làm cho các phương pháp toán học thông thường bó tay nếu đi tìm công thức tổng quát. Nhưng bài toán thật đơn giản nếu chúng ta sử dụng *Đa thức nội suy Lagrange* để tính với độ phức tạp $O(const)$ khi không phải phụ thuộc vào n. Ở đây, bài toán này cũng như bài toán trên về cách làm như nhau tuy nhiên nếu nhìn đa thức thứ hai thì coi như chúng ta không còn sự lựa chọn khác nữa, qua đó để biết độ "mạnh" của thuật toán này…

Đa thức nội suy Larange là một công cụ quá tốt để hỗ trợ mọi bài toán đưa về công thức tổng quát mà chúng ta không cần phải tốn thời gian suy nghĩ và "nháp" ra được các công thức loằng ngoằng. Quan trọng là chúng ta phải cảm nhận được "đâu là đa thức?". Có rất nhiều biểu thức toán học không phải là đa thức mà chúng ta phải cẩn thận, nhưng theo quan điểm và kinh nghiệm của bản thân thì một biểu thức là đa thức nếu nó chỉ chứa dấu cộng, dấu trừ hoặc phép nhân (bao gồm cả phép lũy thừa); còn nó không phải là đa thức nếu nó chứa phép chia phần dư, chia lấy phần nguyên, … Các biểu thức như $\left(\left[\frac{N}{1}\right] + \left[\frac{N}{2}\right] + \cdots + \left[\frac{N}{N}\right]\right)$ hay $((N \,\%\, 1) + (N \,\%\, 2) + \cdots + (N \,\%\, N))$ đều không phải là các đa thức, chúng ta cần phải thận trọng khi gặp các loại bài liên quan đến các toán tử này để tránh ngộ nhận và dẫn đến sai sót!