Hashing

Time limit: 0.6s **Memory limit:** 16M

As a programmer for some large software company, you've been enlisted to make a "cryptographic hash" (a way of "uniquely" encoding a string so that it cannot be decoded).
These hashes can be used for things such as passwords - you encrypt the password so that you can check whether two passwords are equal without ever knowing the passwords themselves.

Although many such algorithms already exist, your manager has decided to reinvent the wheel and make an entirely new algorithm.
Your co-workers have come up with the following code through much trial and error, but sadly it is far too slow for anything practical.
The code:

```
#include <iostream>
#include <string>
using namespace std;

unsigned int hash1(string s)
{
    if (s == "") return 0;
    unsigned int ret = 0;
    for (int j = 1; j <= s.size(); j++)
        for (int k = 0; k < j; k++)
            ret = ret + s[k];

    ret += hash1(s.substr(1));
    return ret;
}

unsigned int hash2(string s, string t)
{
    if (s == "") {
        unsigned temp = 0;
        for (int i = 0; i < t.size(); i++)
            temp += t[i];
        return temp;
    }

    unsigned ret = 0;

    int index = hash1(s) % s.size();
    char ch = s[index];
    s.erase(index, 1);

    ret += hash2(s, t);
    ret += hash2(s, ch + t);
    ret %= 2147483647;
    return ret;
}

int main()
{
    string s;
    getline(cin, s);

    unsigned int hash = hash1(s);
    hash += hash2(s, "");

    printf("⍰X\n", hash);
    return 0;
}
```

Your job is to optimize the code so that it can actually be usable for strings up to, say, length $1\,000\,000$.
Your code must produce exactly the same output as the above code for any input.