Двойна серия 1

Много от вас са чували за Станчо от Клуба на любителите на минерална вода. Освен с рекордните количества минерална вода, които може да изпие, обаче, в университетските сред той е известен и с разнообразни научни постижения. Тъй като Станчо има постижения в почти всички области на науката, алгоритмите с низове не правят изключение. Поради тази причина той е решил да използва такива алгоритми и да подобри постижението си на идващото състезание по пиене на минерална вода в дисциплината "Двойна серия". При дисциплината "Двойна серия" много бутилки минерална вода, от разнообразни марки са наредени последователно на бара. Състезателят пие от последователни бутилки в редицата, като Журито записва марките минерална вода, които са били изпити. Това продължава дотогава, докато състезателят обяви, че е направил "серия". След това, според правилата, той трябва да повтори серията (т.е. да изпие още толкова от следващите в редицата бутилки така, че видовете им да образуват същата последователност). Оценяването се определя от големината на серията, времето за изпълнение и това дали състезателят е успял наистина да направи двойна серия. Станчо държи рекорда на "Двойна серия", но иска да го подобри. Стратегията му трепач е специално оптимизирана – той тръгва от някакво място в редицата, и се движи надясно изпивайки всичко по пътя си. След определен брой бутилки обявава серия, и продължава със същия брой бутилки от мястото, където е завършил първата серия. За да бъде успешна стратегията, Станчо се нуждае от програма, която да избере мястото в редицата, от което да тръгне и броя бутилки които да образуват първата му серия. Поради специалните алгоритми за аритметика които ползва, е необходимо броя на пропуснатите бутилки да се дели без остатък на дължината на серията. Вие трябва да напишете тази програма вместо него.

Жокер: Още една задача решваща се с Rolling Hash алгоритъма. В предните задачи използуващи този алгоритъм споменахме само частния случай за пресмятане на Rolling Hash:

Ako std::string s;
то:

Hash(s.substr(i, k)) = Hash(s.substr(i-1, k+1)) - s[i-1] * pow(HASH_BASE, k) (формула 1)

Общия случай разбира се е супер подобен:

Hash(s.substr(i, k)) = Hash(s.substr(j, i-j+k)) - Hash(s.substr(j, i-j)) * pow(HASH_BASE, k) (формула 2)
където j < i

Сами виждате, че замествайки j с i-1 във формула 2 ще получите по-горната формула 1, тъй като Hash(s.substr(i-1, 1)) = 0 * HASH_BASE + s[i-1] = s[i-1]

Има още едно важно следствие от общата формула за Rolling Hash, а тя е когато j=0:

Hash(s.substr(i, k)) = Hash(s.substr(0, i+k)) - Hash(s.substr(0, i)) * pow(HASH_BASE, k) (формула 3)

С други думи хеша на някакъв подстринг s.substr(i, k) на s е равен на хеша на стринга започващ от първия символ на целия стринг (s[0] и завършващ на последния символ на подстринга (s[i+k-1]), минус хеша на стринга започващ от началото на стринга (s[0]) и завършващ на символа преди първия символ на дадения подтстринг s[i-1], умножен по базата на хеш функцията повдигната на степен дължината на дадения подстринг s.substr(i, k):

```
string s1=..., s2=...;
string s = s1 + s2; //concatenation
```

hash(s2) = hash(s) - hash(s1) * pow(HASH_BASE, s2.length())

Така че формулите за Rolling Hash, които трябва разберете и да запомните (аз не ги помня, просто мога да ги изведа от базовата рекурентна формула за хеш H(i) = H(i-1)*HASH_BASE + s[i]) са вече две:

Hash(s.substr(i, k)) = Hash(s.substr(i-1, k+1)) - s[i-1] * pow(HASH_BASE, k) (формула 1)

и

Hash(s.substr(i, k)) = Hash(s.substr(0, i+k)) - Hash(s.substr(0, i)) * pow(HASH_BASE, k) (формула 3)

Ето как ще използуваме тази формула 3: В масив hashes, пресмятаме всички текущи хешове hash(substr(0,i)) с изместване един символ:

```
string s;
cin >>s;
```

```
int hashes[MAX_STRING_LEN+1];
hashes[0] = 0;
for (int i = 1; i <= s.length; i++)
   hashes[i] = hashes[i-1] * HASH_BASE + s[i-1];
```

Използувайки този масив можем лесно да сметнем хеша на всеки подстринг на s:

```
Hash(s.substr(i, k)) = hashes[i+k] - hashes[i] * PRECOMPUTED_POWERED_BASES[k];
```

Знаейки това вече можем да завъртим цикъл (в низходяща посока: i--) по i за всички възможни дължини на повтарящия се в серия подстринг. Вътре в него ще завъртим втори вложен цикъл по j сравняващ всеки два съседни подстринга с дължина i, започващи на позиции кратни на i: Hash(s.substr(j*i, i)) == Hash(s.substr((j+1)*i, i)) Това че всички възможни съседни подстрингове започват от позиции кратни на i идва от условието: 'Поради специалните алгоритми за аритметика които ползва, е необходимо броя на пропуснатите бутилки да се дели без остатък на дължината на серията'. Сигурно се чудите дали няма да стане комбинаторен взрив от двата вложени цикъла. Не порядъка на този алгоритъм е O(s.length() * log(s.length()))

s.length() идва от външния цикъл
log(s.length()) идва от вътрешния цикъл, поради математическия факт, че 1/2 + 1/3 + 1/4 + ... + 1/(n-1) + 1/n ~= log(n), където log е натурален логаритъм. Това от своя страна е така, защото Инеграл(1/x) = log(x)

Input Format

На всеки ред на стандартния вход е зададен по един низ с дължина от 2 до 4000000, съставен от малки латински букви. Всяка буква означава една марка минерална вода.

Constraints

брой тестови примера <= 18
2 <= дължина низ <= 4000000
размер тестов файл <= 10000000 символа

Output Format

За всеки низ програмата трябва да изведе на отделен ред на стандартния изход броя бутилки които трябва да пропусне Станчо от ляво надясно до началото на серията и дължината на серията, разделени с интервал, така че в резултат да се

получи двойна серия с максимална дължина. Ако има повече от една максимална двойна серия, програмата трябва да посочи тази, която започва най-вляво. Винаги има поне една такава двойна серия във входните данни.

Sample Input 0

```
gagagaaabaabab
kakaribaribarak
```

Sample Output 0

```
6 3
4 4
```