Bài 1. Substring

Khi chuyển từ lập trình trên Pascal sang lập trình trên C, một điểm khác biệt rất dễ sinh lỗi mà Bờm rất hay mắc phải là: tên biến trong Pascal không phân biệt chữ hoa hay chữ thường, còn ngược lại, tên biến trong C có phân biệt chữ hoa hay chữ thường. Biết vậy, vừa rồi để giúp Bờm nắm vững các thao tác xử lý xâu, Cuội đưa cho Bờm một bài toán thách đố sau đây: Cho xâu *S* gồm các chữ cái latin in hoa hay in thường, hãy cho biết xâu *S* có bao nhiêu xâu con khác nhau? Lưu ý là: Ta gọi xâu con của *S* là một dãy gồm 1 hoặc một số ký tự liên tiếp nhau trong *S*, và khi so sánh hai xâu ta có phân biệt chữ hoa với chữ thường.

Yêu cầu: Hãy giúp Bờm giải quyết bài toán mà Cuội thách đố nó.

Dữ liệu: Vào từ file văn bản SUBSTRING.INP chứa xâu *S* gồm các chữ cái latin in hoa hay thường có độ dài không quá 10^4.

Kết quả: Ghi ra file văn bản SUBSTRING.OUT số lượng xâu con khác nhau của xâu *S*.

Ví dụ:

SUBSTRING.INP	SUBSTRING.OUT
AAaaab	17

Giải thích: Xâu 'AAaaab' có 17 xâu con khác nhau:

1. A	4. AA	7. AAa	10. AAaa	13. AAaaa	16. AAaaab
2. Aa	5. Aaa	8. Aaaa	11. Aaaab	14. a	17. aa
3. aaa	6. aaab	9. aab	12. ab	15. b	

Bài 2. Evolution

Các nhà khoa học ở Viện nghiên cứu di truyền Z từ lâu đã biết rằng DNA lưu trữ thông tin di truyền của các cá thể sống. Hiện nay, các nhà khoa học dễ dàng phân tích DNA của cá thể bất kỳ để giải mã và làm rõ các đoạn quan trọng phục vụ cho công việc nghiên cứu. Trong di truyền học, thuật ngữ giải mã có nghĩa là đưa ra mã di truyền dưới dạng một xâu gồm các ký tự A, C, T và G mô tả đầy đủ mã di truyền.

Các nhà khoa học còn phải tiếp tục làm việc cật lực để có thể hiểu được những bí mật ẩn giấu trong DNA, tuy nhiên, đã có những thành tựu đáng kể trong hướng nghiên cứu này. Ví dụ, gần đây cả thế giới đã được biết sự kiện về việc có thể xác định bậc tiến hóa của cá thể thông qua những đoạn mã cụ thể. Các nhà khoa học sẽ sử dụng DNA của cá thể, chọn ra đoạn gồm *N* nucleotide và tính toán độ phức tạp của nó. Độ phức tạp của đoạn được chọn càng lớn cho biết cá thể đã trải qua càng nhiều giai đoạn của quá trình tiến hóa.

Độ phức tạp của đoạn được tính toán như sau:

- đối với đoạn gồm *N* nucleotide, tất cả các đoạn con gồm *K* nucleotide liên tiếp sẽ được xét;
- với mỗi đoạn con tính số lượng các nucleotide cấu thành mỗi loại (tức là tính 4 số là số lượng nucleotide của 4 loại (A, C, T, G) trong đoạn);

- độ phức tạp được định nghĩa là số lượng các tập con định lượng khác nhau tính được. Hai tập con định lượng được coi là bằng nhau nếu số lượng các loại nucleotide mỗi loại là như nhau.

Sau khi thực hiện xong việc khảo sát này, các nhà khoa học sẽ thu thập các bậc độ phức tạp của các cá thể khác nhau. Họ đã có tất cả các chuỗi DNA của các cá thể, tuy nhiên việc khảo sát tất cả các chuỗi là công việc đòi hỏi nhiều thời gian nếu như thực hiện bằng tay.

Yêu cầu: Hãy viết chương trình trợ giúp các nhà khoa học trong việc xác định bậc tiến hóa của cá thể dựa trên chuỗi DNA của nó.

Dữ liệu: Vào từ file văn bản EVOLUTION.INP:

- Dòng đầu tiên chứa hai số nguyên N, K được ghi cách nhau bởi một dấu cách $(1 \le K \le N \le 10^5)$ tương ứng là số lượng nucleotide trong đoạn của chuỗi DNA và độ dài của các đoạn cần khảo sát.
- Dòng thứ hai chứa N ký tự tương ứng với chuỗi nucleotide. Mỗi ký tự chỉ là A, C, T hoặc G.

Kết quả: Ghi ra file văn bản EVOLUTION.OUT một số nguyên là độ phức tạp của đoạn DNA cho trong dữ liệu vào.

Ví dụ:

EVOLUTION.INP	EVOLUTION.OUT
5 1 ACGTA	4
8 2 AACACGTA	5

Giải thích:

- Trong ví dụ thứ nhất: Trong đoạn đã cho có 4 đoạn con độ dài 1 khác nhau.
- Trong ví dụ thứ hai: Trong đoạn đã cho có 7 đoạn con độ dài 2: AA, AC, CA, AC, CG, GT, TA. Các đoạn con AC và CA có các tập con định lượng bằng nhau, do đó độ phức tạp của chuỗi DNA đã cho là 5.

Bài 3. Sắp xếp phân đoạn

Bờm rất yêu thích các bài toán liên quan đến việc thực hiện các phép biến đổi với dãy số. Biết vậy, vừa qua Cuội đã cung cấp cho Bờm một thuật toán sắp xếp dãy số khá thú vị có tên gọi là "Sắp xếp phân đoạn". Trước hết, ta nói dãy số $a_1, a_2, ..., a_N$ là đã được sắp xếp nếu như bất đẳng thức $a_i \le a_{i+1}$ là được thực hiện với mọi $1 \le i < N$.

Ý tưởng của phương pháp sắp xếp phân đoạn là rất đơn giản: Ta sẽ chia dãy số $a_1, a_2, ..., a_N$ ra làm M đoạn. Đoạn thứ nhất gồm các số từ vị trí 1 đến k_1, đoạn thứ hai gồm các số từ vị trí k_1+1 đến k_2, …, đoạn cuối cùng gồm các số từ vị trí k_{M-1} đến k_M. Tiếp theo, để sắp xếp dãy số ta chỉ cần tìm vị trí thích hợp cho các đoạn.

Ví dụ: Để sắp xếp dãy số 3, 4, 5, 6, 1, 2, ta chia nó ra là ba đoạn (3, 4), (5, 6) và (1, 2). Tiếp theo xếp đoạn (1, 2) vào vị trí thứ 1, tiếp đến đoạn (3, 4) vào vị trí thứ hai và đoạn (5, 6) vào vị trí cuối cùng, ta thu được dãy được sắp xếp.

Vấn đề đặt ra là: Đối với một dãy số cho trước cần chia nó ra thành ít nhất bao nhiêu đoạn để áp dụng thuật toán sắp xếp phân đoan ta thu được dãy kết quả là được sắp xếp.

Yêu cầu: Giúp Bờm giải quyết vấn đề nêu trên.

Dữ liệu: Vào từ file văn bản SEGSORT.INP:

- Dòng đầu tiên chứa số nguyên N ($1 \leq N \leq 500000$) là số lượng phần tử của dãy số;
- Dòng thứ hai chứa N số nguyên $a_1, a_2, ..., a_N$ ($1 \leq a_i \leq N$), hai số liên tiếp được ghi cách nhau bởi dấu cách, là các phần tử của dãy số.

Kết quả: Ghi ra file văn bản SEGSORT.OUT một số nguyên là số lượng nhỏ nhất các đoạn tìm được.

Ví dụ:

SEQTRAN.INP	SEQTRAN.OUT
6 5 6 4 3 1 2	4
3 1 2 1	2

Giải thích:

Trong ví dụ thứ nhất: Dãy đã cho có thể chia thành 4 đoạn (5 6), (4), (3) và (1 2). Tiếp đến để tiến hành sắp xếp dãy số ta đưa đoạn (1 2) vào vị trí thứ nhất, đoạn (3) vào vị trí thứ hai, đoạn (4) vào vị trí thứ ba, và đoạn (5 6) vào vị trí cuối cùng. Dãy số thu được là được sắp xếp.

Trong ví dụ thứ hai: Dãy đã cho cần chia là hai đoạn (1 2) và (1).