**Ví dụ**

**Input**

3

4

ACGT

ACGTGCGT

ACCGTGC

ACGCCGT

Chọn {ACGT, ACGTGCGT, ACGCCGT}

“ACG” → 3 \* 3 = 9

3

CGCGCGCGCGCGCCCCGCCCGCGC

CGCGCGCGCGCGCCCCGCCCGCAC

CGCGCGCGCGCGCCCCGCCCGCTC

Nếu chọn từng xâu → 24

Nếu chọn cả 3 xâu: “CGCGCGCGCGCGCCCCGCCCGC” chung 22 ký tự → 22 \* 3 = 66

2

CGCGCCGCGCGCGCGCGCGC

GGCGCCGCGCGCGCGCGCTC

Chọn 1 trong 2 xâu → 20 \* 1 = 20

**Output**

Case 1: 9

Case 2: 66

Case 3: 20

Một tập chuỗi, sao cho tích của tiền tố chung dài nhất \* số lượng chuỗi là lớn nhất

Đề bài yêu cầu xử lý tiền tố → **Trie**

**Node {**

**child[];**

**n\_common: Đếm số lượng tiền tố kết thúc tại node này**

**}**

**Giải thuật**

**Bước 1:** Nhập dữ liệu và khởi tạo cây **Trie**

**Bước 2:** Thêm từng xâu vào cây Trie, tại mỗi node của cây, tăng biến đếm số lượng tiền tố lên 1 (n\_common)

**Bước 3:** Kết quả là độ dài tiền tố \* số lượng xâu chứa tiền tố đó

**Mã giả**

**struct Node {**

**n\_common = 0; // số lượng chuỗi DNA có chứa tiền tố kết thúc tại node này**

**child = [];**

**}**

**function addWord(root, s):**

**res = 0;**

**temp = root;**

**for i = 0 to s.size - 1:**

**ch = s[i] - ‘A’;**

**if temp.child[ch] == NULL:**

**temp.child[ch] = new Node();**

**temp = temp.child[ch];**

**temp.n\_common += 1;**

**res = max(res, temp.n\_common \* (i + 1));**

**return res;**

**read(T);**

**for tc = 1 to T: {**

**read(n);**

**root = new Node();**

**result = 0;**

**for i = 1 to n:**

**read(s);**

**result = max(result, addWord(root, s));**

**print(‘Case ’, tc,’: ‘, result);**

**}**

Độ phức tạp: **O(T \* n \* |s|)**

**Trong đó:**

**T** là số lượng testcase

**n** là số lượng chuỗi DNA

**|s|** là độ dài của chuỗi