H: LCP Queries

原案 : mtsd
問題文 : climpet
データセット : mtsd
解答 : beet, climpet, hos, kotatsugame, mtsd
解説 : mtsd

問題概要

- 英子文字列集合 S = {S_1, ..., S_n}, T = {T_1, ..., T_m} が与えられる
- 以下のクエリに Q 回答えよ
  - 長さ K の数列 a_1, ... , a_k (1 <= a_i <= m) が与えられる
  - 文字列 X を T の a_1 番目の文字列 + ... + T の a_k 番目の文字列と定義する
  - 「S_iと X の最長共通接頭辞の長さ」の総和を返答する
- 制約
  - S に含まれる文字列の長さの総和<= 200,000
  - T に含まれる文字列の長さの総和<= 200,000
  - クエリの K の総和 <= 200,000

想定解1: トライ木 + 文字列のハッシュ

- 前処理:
  - S に対応するトライ木を作り、各頂点に対して以下の情報を計算する
    - その頂点に対応する文字列のハッシュ
    - その頂点に対応する文字列がクエリに来た時の答え
  - 文字列のハッシュ -> トライ木の頂点 となる map を作成する
- クエリの計算
  - 二分探索でクエリの文字列の prefix でトライ木に含まれる最長の文字列を求める
  - トライ木の頂点に記録している答えを参照する
- 計算量

  O(sum |S_i| + sum |T_i| + |S| log |S| + (sum K) log(max |T_i|) log |S|)

  (※2項目の log |S| は接尾辞配列などを使えば消せる)

想定解2: パトリシア木 + 文字列一致判定

- S に対応するパトリシア木の深さが sqrt(sum |S_i|) であることを利用する
  - パトリシア木(Patricia Trie)
    - ざっくり言うと、トライ木を経路圧縮したもの
    - トライ木では辺が 1 文字に対応していたが、パトリシア木では辺は文字列に対応する
- 前処理
  - S に対応するパトリシア木を作り、各頂点に対して以下の情報を計算する
    - その頂点に対応する文字列がクエリに来た時の答え
    - その頂点に対応する文字列を含む S_i の個数
  - S と T の部分文字列の一致比較が可能なデータ構造を作成する
    - ハッシュや接尾辞配列
- クエリ
  - クエリ文字列について、パトリシア木を順に下り、クエリ文字列がパトリシア木のどこに対応するかを計算する（辺上の可能性もあることに注意）
- 計算量

  O(sum |S_i| + sum |T_i| + sum K + Q sqrt(sum |S_i|))

想定解3: 接尾辞配列＋クエリ先読み

- 各クエリ文字列について以下の値を求めておく
  - ソートされたS に対して辞書順でどこの間に入るか
  - その際の前後の文字列とのLCP の値
    - A = (S_1)$...$(S_n)(T_1)...$(T_n) （$ は区切りに対応する文字）
      という文字列に対する接尾辞配列と LCP 配列を利用する
- S とクエリ文字列はソートされた状態で管理し、昇順・降順それぞれの方向に計算を行う
  - ソートされたクエリ文字列に対して、今見ているS_i との LCP の値の列を stack で管理する
  - 各クエリ文字列の LCP の総和を segment tree などで管理し、LCP の値が減少する際に（複数のクエリ文字列に対して同時に）LCP の総和の加算処理を行う
- 計算量
  O(sum |S_i| + sum |T_i| + sum K + |Q| log |Q|)

統計情報

- Acceptances
  - 3 + 3 teams
- First Acceptance
  - SPJ (145 min)