JAG 模擬地区予選 2017

H: Separate String

原案: camypaper

問題文: darsein

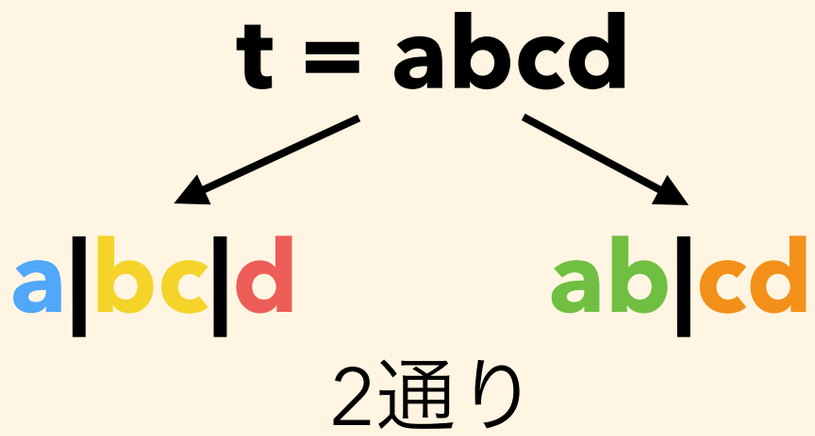
解答: climpet, darsein, not

解説: darsein

問題概要

- 1つの文字列 t と辞書となる文字列集合 S がある
- t を複数の文字列に分割するとき、分割後の文字列すべてが S に含まれるような分割はいくつあるか？
  - S に含まれる文字列が複数回分割に現れてもよい
- 制約: $1 \leq |t| \leq 10^5$, $1 \leq |S| \leq 10^5$, $1 \leq |x| \leq 10^5 (x \in S)$, $\Sigma_{x \in S} |x| \leq 2 \times 10^5$

S = {a, ab, bc, cd, d}



想定解法: DP

- dp[i] := t[1:i] を分割する方法の数

  (t[l:r] := t の l~r 文字目の部分文字列)

- $dp[i] = \Sigma_{1 \le j \le i,\ t[j:i] \in S} dp[j-1]$
  - ただし、dp[0] := 1
- 素直に計算すると $O(|t|^2 \times t[j:i] \in S$の判定時間)

  → TLE

S = {abcde, cde, e}

dp[5] = dp[4] abcdefg
+dp[2] abcdefg
+dp[0] abcdefg

考察

- $L = \Sigma_{x \in S} |x|$ とおく
- $N \leq 10^5, L \leq 2 \times 10^5$
  →文字列の長さの種類は高々 $\min(N, \sqrt{L}) < 450$
- DPで参照すべき区間 [j:i] は各iにつき高々450個

  → 各 i ごとに S のどれかにマッチする区間 l_i を

  前処理で求めておけば、

  $dp[i] = \Sigma_{j \text{ s.t. } [j:i] \in l_i} dp[j-1]$

  とでき、$O(|t| \min(N, \sqrt{L}))$ で計算可能

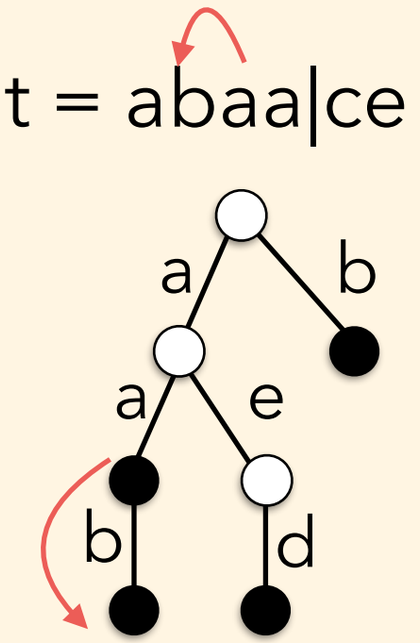
想定解法: 文字列マッチングパート

1. 普通にAho-Corasickをする
    - 構築: O(L), 判定: O(|t| + |マッチ箇所|)
    - マッチ箇所は高々 O(|t| min(N, √L)) 個なので OK
2. 短い文字列 (多いかも) と長い文字列 (少ない) に分けて別々に前処理
    - 両方ローリングハッシュが使えるので実装が楽かも
    - 以降詳しく見るが、結局 O(|t| min(N, √L)) でできる

想定解法: for short strings

- もしSの文字列がすべて短かったら...
  - (インクリメンタルに判定できる) 複数文字列検索データ構造を用いる
    - 例: Trie, ローリングハッシュ, etc.
      - S中の文字列の逆文字列に対してデータ構造を構築
      - t[j:i]の計算後、t[j-1:i] (の逆) を1文字追加で検索
  - 構築: O(L), 判定: O($\max_{x \in S} |x|$)
- 前処理にO(L)、DPにO(|t| $\max_{x \in S} |x|$) で解ける

S = {aa, b, baa, dea}



想定解法: for few strings

- もしS中の文字列の数が少なかったら...
  - 普通の文字列検索アルゴリズムで前処理する
    - 例: KMP, Suffix Array, ロリハ, etc.
      - $x \in S$ごとに普通に文字列検索、出現区間をマーク
      - 全体で構築: O(L), 判定: O(L + |t|) とか
- 前処理にO(L+|t|)、DP時は O(|t| |S|) で解ける

x = abcab

t = abcabcab

[1:5]

[4:8]

Writer 解

- climpet: 92 行 1622 bytes (C++, Aho-Corasick)
- darsein: 188 行 3769 bytes (C++, Trie + KMP)
- not: 57 行 1457 bytes (C++, ロリハ)

統計情報

- AC / submissions
  - 17 / 68 (25%)
- First Acceptance
  - hankan_rta (53:18)

文字列アルゴリズム資料

- Knuth-Morris-Pratt (KMP) 法:

  http://snuke.hatenablog.com/entry/2014/12/01/235807

- Trie (競プロで double array まで必要になることはあまりない):

  http://d.hatena.ne.jp/takeda25/20120219/1329634865

- Aho-Corasick 法:

  d.hatena.ne.jp/naoya/20090405/aho_corasick

- ローリングハッシュ (Rabin-Karp 法):

  https://topcoder.g.hatena.ne.jp/spaghetti_source/20130209/1360403866