Problem F
Sakura Poetry
桜詩 ~願はくは花の下にて春死なむ~

原案: 野田
解答作成: 野田・田山・岩田
解説作成: 田山

概要

- 接続辞書に従って詩を作りなさい
  - 季語が一度だけ登場するように
- 詩は何通りに作られる？
  - MOD 1,000,000,007 で

方針

- 言うまでもなく全探索はダメ
- DP
  - (長さｍの)
  - (単語ｗで終わる)
  - (i番目の季語に頭ｊ文字までマッチした)
  - (既に季語をｃ個含んでいる)
  - 文字列の個数を求めていく

季語のマッチ状態

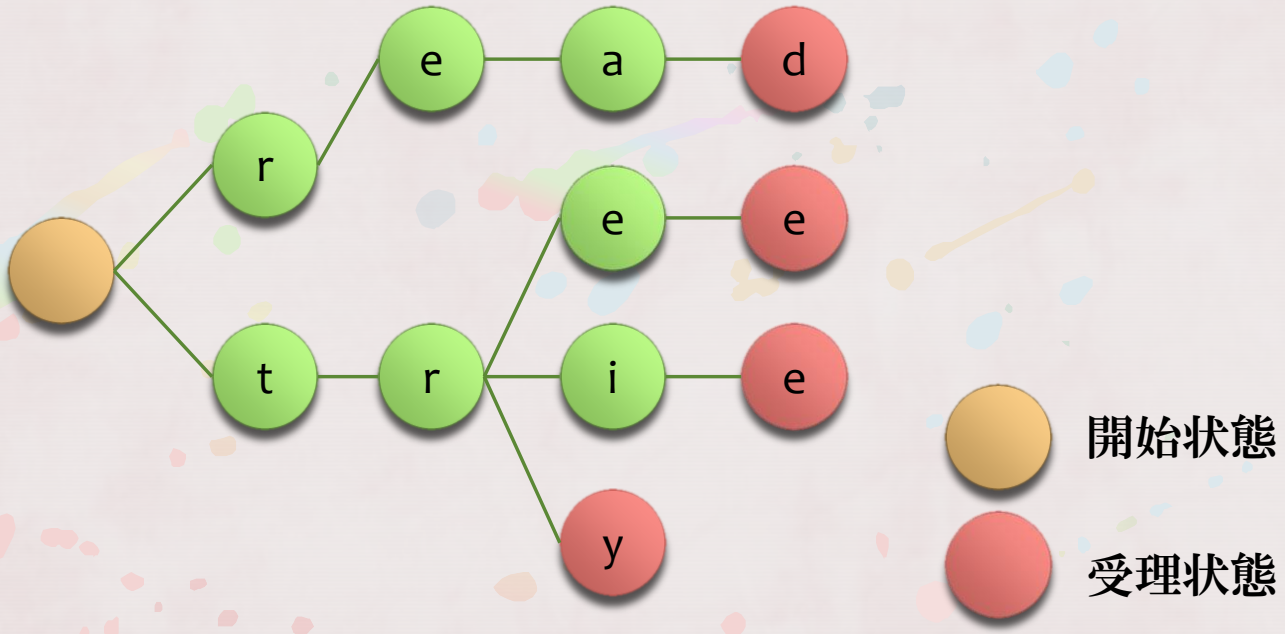
- (i番目の季語に頭 j 文字までマッチした)
  - 状態数 K^L
    - L = 季語の最大長 = 20
  - とても大きいので工夫が必要
    - 「どの文字列にどこまでマッチしているか」という情報を簡潔に表せる構造がほしい
    - K^L 個の状態のほとんどはありえない状態
      - ‘sakura’ の 頭 2 文字、‘hana’ の頭 1 文字に同時にマッチする文字列などない

パターンマッチングオートマトン

- 季語の集合に対して パターンマッチング オートマトン (PMA) を構成する
  - 文字列 s の接尾辞がパターンにマッチすること
  - 文字列 s を PMA に入力したとき 受理状態にあること
  - この2つが同値であるようなオートマトン
- Aho-Corasick algorithm として知られる
  - パターンが1つだけの場合 Knuth-Morris-Pratt algorithm

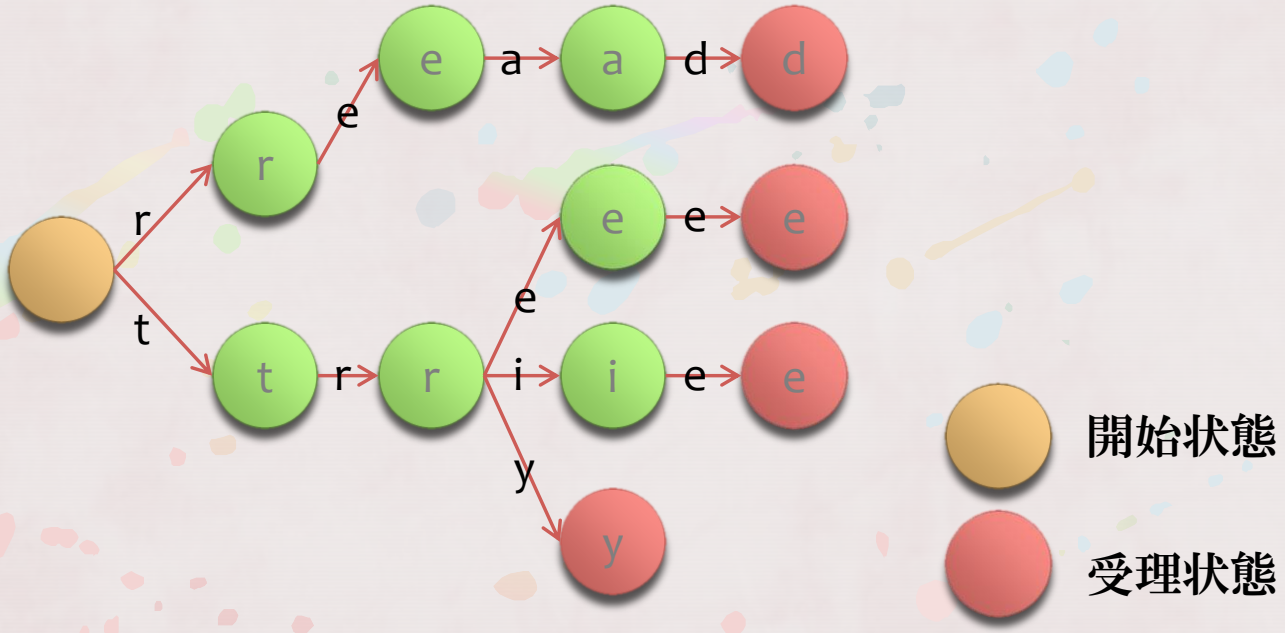
パターンマッチングオートマトン

- パターンの集合を Trie 木で表す
  - 例: { ‘read’, ‘tree’, ‘trie’, ‘try’ }



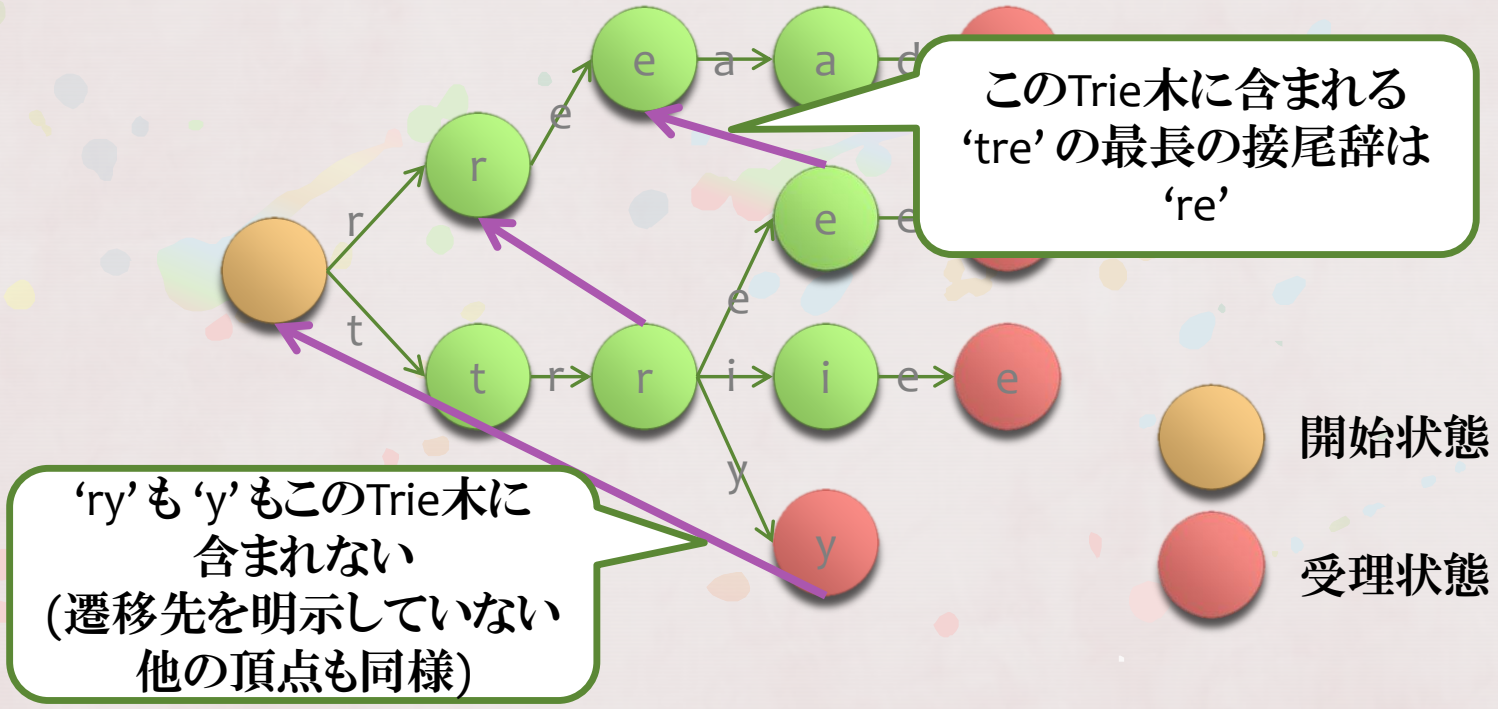
パターンマッチングオートマトン

- 一致時の遷移
  - 自明



パターンマッチングオートマトン

- 不一致時の遷移 (failure link)
  - Trie 木に含まれる、その頂点の最長の接尾辞へ遷移



Aho Corasick Algorithm

- PMA を用いた文字列の検索
  - PMA に文字列を流し込む
    - 遷移先の頂点があれば、その頂点に進んで文字列を1文字読み進める
    - 無ければ、文字列を読み進めずに failure link で遷移し、マッチを試みる
      - それでも無ければ、もう一度 failure link で遷移してマッチを試みる
      - Trie 木の根でも遷移先が見つからなければ、諦めて文字列を1文字読み進める
- 詳しくは他の資料を
  - Failure link の効率的な構築方法なども
    - この問題ではDP部分の計算量が支配的なので、PMAの構築は手抜きをしてもよい

方針

- 明らかに全探索はダメ
- DP
  - (長さｍの)
  - (単語ｗで終わる)
  - **(i番目の季語に頭ｊ文字までマッチした)**
    - **これを PMA の頂点で置き換えられる**
  - (既に季語をｃ個含んでいる)
  - 文字列の個数を求めていく

DP

- DP[n][s][w][c] :=
  - 長さ n で
  - PMA に流し込むと頂点 s に辿り着く
  - 単語 w で終わる
  - 既に完成した季語を c (= 0 or 1) 個含んだ
  - 文字列の総数
- を計算していく
  - 単語 w の後に w' を接続できるならば
    DP[n+length(w')][s'][w'][c'] += DP[n][s][w][c]
  - s', c' は Aho-Corasick で求まる

計算量

- 空間計算量
  - PMAの総頂点数は季語の長さの総和程度
  - 単語数は高々接続辞書のサイズの2倍
  - $O(NMKL)$
    - L = (季語長) = 20
    - DPテーブルのサイズは 1.2GB 程度
- 時間計算量
  - $O(NMKL)$
    - 数億回のループ

Memory Limit Exceeded

- 国内予選なので実行制限時間がない
  - 数億回のループはなんとかなる
- メモリはなんとかならない
  - 以下のどちらかの工夫が必要
    1. 状態を map で持つ
    2. 配列を使い回す
  - PCの性能によってはなんとかなってしまうかも

状態を map で持つ

- DPテーブルの多くの要素は 0
  - 0 でない要素だけを map で持つ
    - std::map (C++), java.util.TreeMap (Java), etc.
  - Map<Status, int> dp[N];
    - Status に s, w, c を持たせる
- メモリ消費量は見積もりにくい
  - 実際には実行速度も速い

配列を使い回す

- DPしていく過程で、DP[i] と DP[i+L+1] が同時に必要になることはない
  - DP[L+1][S][W][C] だけ宣言すれば十分
    - 48MB程度

Judge Solutions

- 野田: 254行 (C++)
- 田山: 194行 (C++)
- 田山: 115行 (C++)
- 岩田: 103行 (Java)

Result

- #Submitted Teams: 3
- #Accepted Teams: 3

1. ~usaaaaagi (1:37:35)
2. _____ (1:50:33)
3. STAFF (2:21:19)

Congratulations!