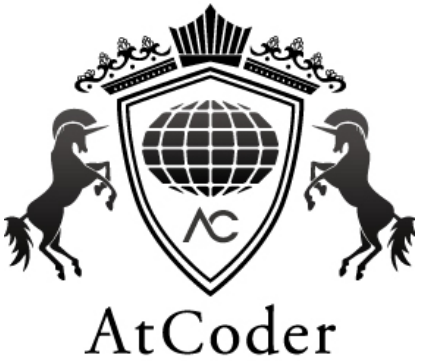
天下一プログラマーコンテスト2014
決勝 解説



AtCoder株式会社 代表取締役
高橋 直大

A問題　塙さん

1. 問題概要
2. アルゴリズム

A問題　問題概要

AtCoder

- 正の整数Xのh進数での表現が以下の条件を満たすとき、Xは塙さんであるという。
  - 同じ文字の出現回数はn回以下である
  - W桁である
  - 0から始まらない
- 塙さんの個数を求めよ。

- 制約
  - 2 ≦ h ≦ 64
  - 1 ≦ n ≦ 512
  - 2 ≦ w ≦ 2048

A問題　部分点解法

AtCoder

- h ≦ 36, n ≦ 4, w ≦ 4
  - この制約は、全探索で解くことが可能
    - h進数でw桁の整数で表現できるのはh^w通り
  - 列挙した整数に対して、条件を満たしているか調べるだけ

A問題　満点解法

AtCoder

- 動的計画法で解くことが出来る
  - 「何文字目まで考慮したか」「数字をいくつ使ったか」の2つを状態とする。
  - そこから、「次の文字をいくつ使うか」を分岐させる
    - dp[i+1, j+k] = dp[i,j] * (k文字を追加する組み合わせ)
  - 計算量はO(hnw)
  - 0から始まるパターンの処理もどうにでもなる
    - 方法1: 最初のループだけ組み合わせの処理を変える
    - 方法2: 全パターンの1/wが0から始まるので、適当に分岐

A問題　満点解法

AtCoder

- k文字を追加する組み合わせの求め方
  - 現在j文字使われている
    - 残りはw-j文字
  - そこにk文字追加したい
  - 組み合わせはCombination(w-j, k)
    - これを掛け算してあげれば良い。

B問題　天体位置観測

1. 問題概要
2. アルゴリズム

B問題　問題概要

AtCoder

- 夜空に存在するN個の星の座標と、10*10以下のM個の星座のパターンが与えられる
- 各星座が夜空にいくつ存在するかを出力せよ

- 制約
  - 1 ≦ N ≦ 10^5
    - 0 ≦ X,Y ≦ 10^6
  - 1 ≦ M ≦ 100
    - 1 ≦ L ≦ 100
    - 0 ≦ x,y ≦ 9

B問題　部分点解法

AtCoder

- N≦100の時、N≦100、M≦100なので、全探索が可能
  - それぞれの星座に対し、最も左上にある星を全探索
  - M個の星座に対し、N個のパターンを調べ、毎パターンに付きL種類の星を調べる。各星につき候補はN個。
    - O(MN^2L)
      - 1億くらいなので間に合う
      - N個をしらみつぶしに調べるのではなく、mapなどを使って調べるともっと早い

B問題　満点解法

AtCoder

- 各星座を全探索すると？
  - パターン数はMN
  - これについて、L個の星について、存在するかどうか判定
  - 判定処理をO(1)で行ったとして、計算量はO(MNL)
    - 今回の制約だと10億くらい。
    - Hashmapとかを使うと間に合いそう？
      - 残念ながら間に合いません。
    - もうちょっと高速化したい

B問題　満点解法

AtCoder

- 星座の大きさに注目する
  - 星座のサイズは0≦x,y≦9
    - つまり、10*10の100マスについて調べれば良い。
      - 夜空の任意の10*10マスについて、星が1つでも存在するパターンはN*10*10通り以下
    - 星が「一番左上に」存在するパターンに限定すれば、N通り
      - つまり、今回の制約においては10万通り以下
      - 10万通りの10*10マスについて、星座がマッチングするか調べれば良い
      - マッチングの際に、星座をずらして判定する必要はない
        - » 星座の最も左上の星の場所を固定してしまえば良い

B問題　満点解法

AtCoder

- 星座のマッチングを高速に行うには？
  - N通りの10*10マスの埋まり具合を、予め調べておく
    - 星座の種類によっては左上に星が存在しないパターンもあるので、上だけ固定して、19*10くらいまで調べておくと良い
      - 調べる時にmapだと遅いので、hashテーブル系統のものを使う！
    - これは、3つの64bit変数で表せる！
  - 各星座について、and演算を行うことで、各星座とのマッチングが一瞬で可能

C問題　シークエンサー

1. 問題概要
2. アルゴリズム

C問題　問題概要

AtCoder

- A,T,C,Gのいずれかの文字と任意の 1 文字にマッチするワイルドカード(.)からなる文字列のパターンが *N* 個与えられる。
- 1 つのパターンには最大で 1 個のワイルドカードが含まれる。
- 与えられたパターン全てを部分文字列として含む文字列 *X* のうち最も短いものの長さを答えよ
- 制約
  - 1 ≦ N ≦ 12
  - 1 ≦ |Si| ≦ 600

C問題　部分点解法

AtCoder

- まず、ワイルドカードが存在しないケースから考える
- 動的計画法で解きたい
  - N≦12なので、「どのパターンを使ったかの集合」と「最後に使った文字列」を状態と出来る。
    - この状態で動的計画法をするにはどうしたら良いか？

- パターンA -> パターンBと接続した時に、間にパターンCが入っている場合がある。
  - 例：“abcd”,“cdef”を連結させた時、間に”cde”が入っている
  - これを上手い事処理しなければならない！

C問題　部分点解法

AtCoder

- 難しいパターンの例
  - “abbb”, “bbba”, “bbbb”の時
    - “abbbba”と繋げばよいのだが、”abbb”->”bbba”だと”abbba”に
    - “abbb” -> “bbbb” -> “bbba”と繋げば”abbbba”が得られる
  - “abbb”, “bbba”, “bb”の時
    - “abbba”と繋げば、答えが得られる
    - “abbb” -> “bb” -> “bbba”の時、”bb”に最後に繋いだ、という情報しか残していないと、最後にbを余計に1つ足してしまう
  - この辺りの処理を上手くまとめてあげる必要がある

C問題　部分点解法

AtCoder

- 仮説：ある二つの文字列を繋ぐ時、最短となる繋ぎ方だけ考慮すれば良い
  - 例えば、”abbb”, “bbba”の時、”abbba”だけ考慮し、”abbbba”などを考慮しなくて良い？
    - “bbbb”のような入力がある時は困るが、これは”abbb”->”bbbb”->”bbba”の順なら”abbbba”が得られるから問題がない。
  - これが成立するのであれば、このパターンだけ調べてbitDPすれば良い。

C問題　部分点解法

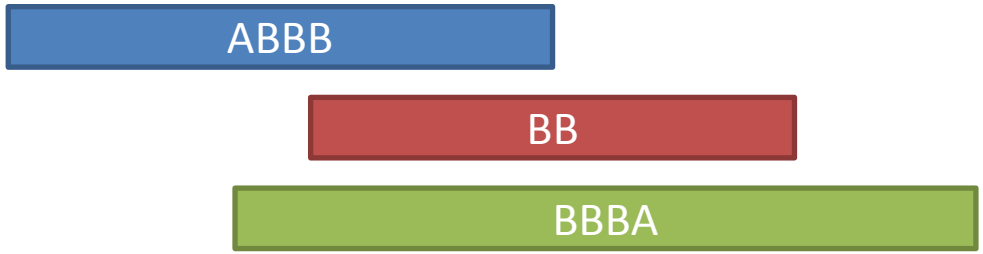
AtCoder

- 仮説：ある二つの文字列を繋ぐ時、最短となる繋ぎ方だけ考慮すれば良い
  - 反例となり得るパターンは以下の2つ
    - 冗長にすることにより、間に入るパターンが存在する場合
    - 冗長にすることにより、後に入るパターンが普通に繋ぐより早く繋げるパターン
  - この2つのパターンが存在しないことが分かれば、最短となる繋ぎ方だけを考慮すれば良いことが解る

C問題　部分点解法

AtCoder

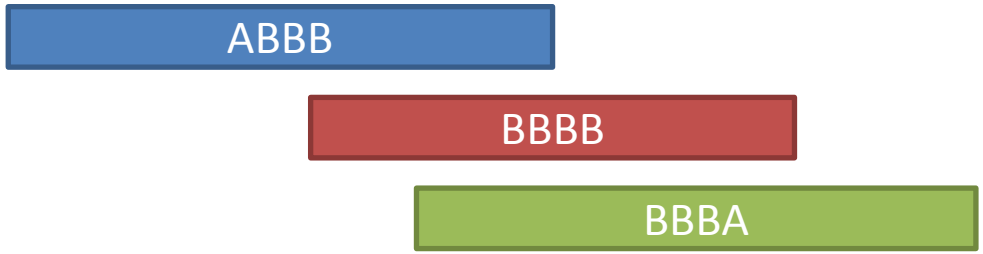
- 冗長にすることにより、間に入るパターンが存在する場合
  - 以下のように、赤の左端が緑の左端より右に存在する場合は、そもそも赤は緑の部分文字列となっているため、青との接続関係に依存しない。
    - 青→緑と接続すれば良く、赤はその際にカウントしてあげれば良い



C問題　部分点解法

AtCoder

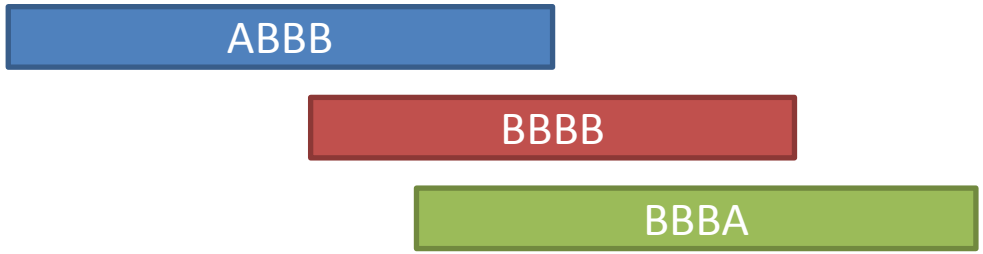
- 冗長にすることにより、間に入るパターンが存在する場合
  - 以下のように、青の左端、赤の左端、緑の左端が昇順になっているようなパターンの場合は、青→赤→緑という順番で調べた時に検出できるパターンなので、無視して良い。
    - 赤を最短で置いた時に、緑が適切におけるかどうかは、次のパターンを参照



C問題　部分点解法

AtCoder

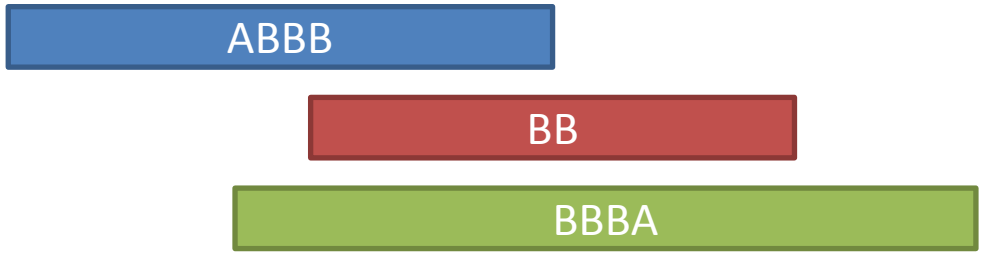
- 冗長にすることにより、後に入るパターンが普通に繋ぐより早く繋げるパターン
  - 以下のように、左端が昇順の場合は、緑は青と赤の位置関係に影響を受けないため、赤を最も左に配置していても問題ない



C問題　部分点解法

AtCoder

- 冗長にすることにより、後に入るパターンが普通に繋ぐより早く繋げるパターン
  - 以下のように、緑の左端が赤の左端より左にある場合は、そもそも緑は赤に含まれているため、青→緑と繋げば良い。



C問題　部分点解法

AtCoder

- これまでの考察より、文字列にワイルドカードが存在しない場合は、可能な限り左に寄せて接続して良いことが解る
  - これを利用してDPを行う。
- 状態は、[今まで使ったパターンの集合][最後のパターン]
  - ここから、[次に使うパターン]ごとに分岐
  - O(2^N * N^2)
- 前処理として、各パターン同士を接続した時に、自動的に含まれるパターンと、いくつずらして繋ぐかを計算しておく
  - O(2^N * |S|^2)

C問題 満点解法

AtCoder

- ワイルドカードは各パターンにつき1つしか含まれないので、4通りの文字列を予め作ってしまえば良い。
  - DP部分は、最後に置いたパターンが4倍に増えるだけ
  - 前処理は、パターンが4倍になるので、16倍になる
    - 16倍程度であれば、大体何をやっても間に合う

D問題　高橋君

1. 問題概要
2. アルゴリズム

D問題 問題概要

AtCoder

- 文字列 S は、次の条件を満たすとき高橋君であるという。
  - S は、0, 1からなる文字列である。
  - S の長さが n である。
  - S は高々 k 個の 1 を含む。
- 高橋君の個数を 1000000007 で割った余りを求めよ。

- 制約
  - 1 ≦テストケース数T ≦ 100000
  - 1 ≦ n, k ≦ 100000

D問題　部分点解法

AtCoder

- 求めるべきものは、i≦kの全てのiに対して、ΣnCiを求めれば良い。
- これを、全てのn,kについて事前に計算しておく
  - nCiは、パスカルの三角形を用いて計算可能
  - その計算結果に対し、累積和を用いることで、i≦kの全てのiに対してのΣnCiも簡単に求めることが出来る
  - O(n^2 + T)
  - 部分点であれば間に合う

D問題　満点解法

AtCoder

- 基本的に求めるものは同じ
  - だが、10万以下の全てのn,kに対して事前計算することは不可能
- 平方分割で、必要な部分だけを事前計算しておく！
  - nが500の倍数のみ事前計算しておく
    - そこから補完してあげれば良い！
    - 例：N=1503, K=54の時
      - 1500文字の文字列について考える
        - » 1が51個以下の場合、残り3文字は何を入れても高橋君になる
        - » 1が52個の場合、残り3文字中、1が2文字以下なら高橋君
        - » 1が53個の場合、残り3文字中、1が1文字以下なら高橋君
        - » 1が54個の場合、残り3文字中、1が0文字以下なら高橋君
        - » 1が55個以上の場合、残り3文字に拠らず、高橋君でない
- 計算量は$O(n\sqrt{n} + T\sqrt{n})$