※探索される文字列をAとし，探索する文字列をBとする.

※Aの配列の大きさをnとし，Bの配列の大きさをmとする．このときn>m

文字列探索

　ある文字列Aの中から，別の文字列Bを探索すること．もし文字列Aの中に文字列Bが現れないなら存在しないと返し，現れるならばその現れた位置を返す．テキスト編集や，データ探索で用いられることが多い.

疑問点

・文字列探索と文字列照合は完全に同じことを指している？

・配列を用いるときのの表現として正しいのは 前後ろ？左右?

アルゴリズム一覧

・単純法（力任せ法）

・クヌース-モリス-プラット法（KMP法）

・ボイヤー-ムーア法（BM法）

　├ボイヤー-ムーア-ホースプール法

　├Apostolico-Giancarloアルゴリズム

　└Quick Search法（Sundayのアルゴリズム？）

・ラビン-カープ法

　・Bitapアルゴリズム（Baeza-Yates-Gonnetアルゴリズム,shift-andアルゴリズム, shift-orアルゴリズムとも呼ばれる）

・Suffix Array（接尾辞配列）

・エイホ-コラシック法

各アルゴリズムの説明（文字列の長さをn パターンの長さをmとする）

単純法：O(mn)

文字列とパターンが一致するか先頭から順に確認する

KMP法：O(n+m)

不一致時にいくつずらすかをあらかじめ表として格納しておくもの

BM法：最悪O(n)　平均O(n/m)

文字列の左から比較するが毎回の比較時はパターンの右から比較を行うことで不一致時に大きくずらすことができる

ボイヤー-ムーア-ホースプール法

BM法を簡略化したもの

Apostolico-Giancarloアルゴリズム

ある位置での比較で，一致することがわかっている文字を飛ばすことで高速化.

Quick Search法

BM法の中で最も高速であるとされる．不一致時に比較した右隣から移動量を求める

ラビン-カープ法：O(n) 稀にO（nm）

ハッシュ関数を用いたもの.

Bitapアルゴリズム

ビット演算の並列性を利用したもの.

Suffix Array：O(nlogn)?

データ構造の一つで文字列探索や全文検索に利用される．文字列から一文字ずつ削ったものをソートして考える

エイホ-コラシック法：mの線形

木を用いたもの？

詳細

単純法

最初は文字列Aの先頭と文字列Bの先頭を合わせて，先頭から順に比較を行う．

文字列Bの最後まで一致した場合に成功とし，その位置を返す．比較の途中で一致しない文字が現れたら，文字列Bを一つ後ろ（右）へずらし，また文字列Bの先頭とずらした位置に合う文字列Aのある位置の前から順に比較を行う．これを繰り返し，文字列Aの最後まで比較を行う．最後まで一致する位置が見つからなかった場合に失敗となる．

この方法の場合，ずらす回数はn-m回なのでO(n)となる．ずらす毎に行われる比較の回数はO(m)なので，全体の計算量はO(mn)となる．