# 高级数据结构与算法

# 绪论

# 线性存储结构

## 单调队列/栈。

所谓单调队列，就是队里元素是单调递增或递减的队列,一旦入队元素使得队列不单调，就把队尾元素删除直到满足单调性后再插入那个新元素，这个实际是栈的性质，本来队列是尾入头出，而不能删除尾部。所以有的地方叫做单调栈，但因为队列支持从队头出队的功能(在滑动窗口问题有用到)，所以习惯叫单调队列，它确实不是真正的队列，它同时具有栈和队列的性质。

编码上，c++有deque的双端队列可用，也可以用来手动实现。(有人说c++库慢，但之后实测一点不慢)。但是最快的是手动实现的数组双端队列。这么做在算法竞赛中只要开够了内存(2倍于峰值)，就不需要释放。

### 模型一,找字典序：

给定一个序列，找字典序最小的子序列。从头到尾构建单调队列即可

### 模型二,滑动窗口问题：

所谓滑动窗口就是给一个长度是n的数列A,对于数列每一段长度是m的区间，都要求得一个最值，**单调递减队列**可以用来求滑动窗口的**最大值(这里习惯队尾在前，队头在后，所以单调递减队列是维护最大值)**。其算法复杂度是O(n)。这是正招，如果用堆或线段树等结构，复杂度是n\*logn，单调队列的一个用途是利用其滑动窗口最值优化动态规划问题的时间复杂度。

例如poj2823，以滑动最大值为例

用单调递减队列记录最大值，构建单调队列整个过程队列头部的数所在位置和队列尾部所在位置一定不能超过m,如果超过了就要把队列头部出队(保证了求长度是m的区间内最值)。如何知道队列每个元素位置，在队列额外维护一个属性即可。

先把A序列前m个元素按照单调队列规则入队，同时保证队列长度，处理完前m个数，队尾就是最大值(很好理解，单调队列尾永远是最大的)，之后按顺序每次入队A的一个元素A[i]，每次输出队列尾部，就是区间[i-m+1,i]最大值，同时一定保证队列长度.这里要注意边界问题，因为是区间最大值，所以单调队列出队时，相等元素也要出队，这样可以更新所存的位置变量，使其更靠后

## ST表

## 后缀数组

后缀数组是处理字符串的有效工具

# 树和森林

## 线段树

## 树状数组

## 二叉查找树

AVL树，treap，fhq\_treap，伸展树 ,红黑树，替罪羊树，sbt树，2-3树，01树

## 并查集

## 可并堆

## 动态森林

# 图

## 适应性问题

## K短路问题

## K小生成树问题

## 联通分量问题

## 网络流问题

## 二分图问题

## 公共祖先问题

## 跳舞链

# 字符串和有限状态机

## AC自动机

Ac自动机是Aho-Corasick 提出的一种dfa

概念去维基看，简单来讲DFA是一张图，有一个开始节点，代表空串，其他各个点并没有实际意义(准确来讲部分自动机，但是不是所有的自动机都要求节点有意义)，点之间有边。

DFA的图的边是有实际意义，一条边代表了**一种字符**，比如在纯英文小写字符构建的dfa中，每个节点有26个分叉，但分叉不一定指向某个实际的点，也可能指向空，如果从起点出发，走出一条路径，那么这个路径可以代表一个字符串。

Ac自动机解决了多模式串匹配的问题。

给定6个模式串字符串

6

beta

alpha

haha

delta

dede

tata

主串是：dedeltalphahahahototatalpha

利用模式串构建ac自动机，完了用主串在ac自动机里跑一边即可。

下面说怎么构建ac自动机，每个节点维护三个属性，son[26]代表26个叉，fail代表失败指针，cnt是代表以这个点为结尾的字符串数量初始是0，注意ac自动机一个点只可能是一个字符串的结尾，所有构建的新节点son和fail指向NULL，事先构建好根节点root，其他节点构建分为三步：

1. 构建字典树，比较容易，每加入一个字符串，就遍历一遍每个字符，字典树从根开始如果没有这个叉，就建立一个新节点，并且连上，向这个方向走；如果有这个叉，就直接向这个方向走即可。走完后最后一个点的cnt+=1
2. 建立fail指针，从root利用广度优先搜索顺序建立，首先处理root，root->fail=root,遍历根节点所有存在的叉，把它们入队，并且把它们的fail连到root。处理完root，对于非root节点，开始广搜的循环，每次取出一个点，找它的所有叉now->son[i]：如果now->son[i]存在, now->son[i]->fail= now->fail->son[i],这里如果now->fail->son[i]不存在，now->son[i]->fail就连到root；如果now->son[i]不存在，就now->son[i]= now->fail->son[i]
3. 对于主串str，从i=0开始，ac自动机从root开始，匹配一遍：如果now->son[str[i]]存在说明匹配成功，正常的前进一步；

如果now->son[str[i]]不存在说明匹配未成功，沿着now->fail反复跳，跳几次取决于问题本身，不同问题有着不同思路，列举几个

1.求模式串在主串中出现几个，同一个多次出现也记录：

沿着fail一直跳，直到跳到根为止，每次加上cnt的值，总和就是答案。

2.求模式串在主串中出现几个，同一个多次出现不记录：

沿着fail一直跳，途中访问的节点的cnt做上标记，比如让他是-1，跳到root了或者跳到是-1的节点就退出，每次加上cnt的值

* Ac自动机指向空的指针，在代码实现时候都视为指向根，有助于简化代码

## 后缀自动机

## 回文自动机

# 多项式算法

## FFT

## NTT

## FWT

## 多项式乘法

## 多项式求逆，除法和取模

## 多项式开根号

## 多项式指数函数

## 多项式对数函数

# 可持久化数据结构

## 可持久化线段树

## 可持久化可并堆

## 可持久化并查集

## 可持久化fhq\_treap

# 查找

## 莫队分块

## CDQ分治

## 整体二分

# 数论算法

## 快速幂与扩展快速幂

## 素数的线性筛法

## 积性函数的亚线性筛法