#### 评测系统剖析与选手参赛策略

唯一听懂了的一节课。

1. 通过 /proc/<pid>/status 可查看一个进程的各项状态，其中包括内存用量。
2. 如果一个程序占用过多内存了？ulimit -v 内存限制（KB数）

#### 量子计算初步

？？？

#### 信息学竞赛中的字符串算法的应用

Q1: 对于字符串S，O(nlg2n)预处理，O(lgn)回答区间不同子串个数。

考虑倒着加入字符，在加入S[l]时，考虑S[l:r]的上一次出现在S[k:k+r-l]，那么在线段树上[r,k+r-l)区间+1，即当询问右端点落在这个区间内时有1的贡献，然后求线段树上r点的值作为询问[l,r]的答案。

考虑后缀树上的一条路径(x,y)，如果这条路径上最近一次都是被后缀S[k:]所影响，接下来要被S[l:]影响，那么我们就可以一起处理这条路径，并且可证新加入的字符只会影响O(lgn)段这样的路径。怎么处理，令len[parent[x]]=a，len[y]=b那么从后缀树的根到y的构成字符串的(a,b]的前缀在s[l:]上出现，上一次出现时在S[k:]，所以对应到上面求贡献的方法，在线段树的区间+1，考虑一整条路径，可以发现r每次+1，k和l不变，k+r-l每次+1，相当于区间加等差数列。

什么你不会区间加等差数列？巧了我也不会。答案是在线段树上记录每段的首相和公差即可。

Q2: 对于字符串S，O(nlg2n)预处理，O(lg2n)回答子串对应后缀树节点数。

同样考虑倒着加入字符。考虑当前所有后缀建成的后缀字典树T1，后缀树T2上节点数=T1的叶子节点数+T1上有两个儿子的节点数=已加入的后缀数量+T1上至少有两个儿子的节点数-既是后缀节点又有至少两个儿子的节点数。

* T1上至少有两个儿子的节点数

在后缀树上定义一个结点的实儿子为所有儿子中最后被访问到的儿子，用LCT维护。

在左端点从 l+1 移动到 l 之后，需要找到因为后缀P[l,n]而产生的新的两个儿子的结点。

插入后缀P[l,n]时，相当于加了一条链，但只有在轻重链切换的地方有可能产生新的贡献。

具体地，用lastt 表示字符串 t 上一次出现的位置，用difft 表示字符串 t 上一次出现不

同后继的位置（这个在access时维护），如图所示： 

那么子串要在P[l+1,r]没有两个不同后继，在P[l,r]有两个不同后继字符，所以r要在区间[lastt+|t], difft+|t|)中，就是对线段树这个区间做+1操作，询问时单点查询即可，这一部分是预处理O(lg2n)，询问O(lgn)的。

* 既是后缀节点又有至少两个儿子的节点数

经过分析可以发现这样一个关键性质：如果一个串 t 在 P[l,r] 中有两个不同的后继，那么 t 的所有后缀都至少有两个不同的后继。所以我们首先套上一个二分的lg，二分了前缀P[mid,r]，然后问题转化为给定一个串P[mid,r]，判断这个串在一段区间[l,r]中是否有两个不同的后继。

对每个结点维护一个线段树，定义为如果这个点对应的串有一个出现位置P[a,b]那么a 在这个点的线段树中，可以通过线段树合并求出（同求sam的right集合的方法）。

在询问的时候，我们首先要在后缀树的lct上找出P[mid,r]。

什么你不会找？直接跳重链，判断重链顶的len长度，然后在轻链的splay上二分就行了吧。（确信

然后如果这个串没有对应节点，即它在后缀树的某条边上，那么就只有一种后缀字符。否则令对应点为p，因为我们无法在p的线段树上搞清楚到底有几个儿子产生了贡献，我们把儿子分为实儿子和虚儿子。

首先在实儿子的线段树上询问是不是有[l,mid-1]，（有的话，令答案为x，则得到一个后缀字符x+|P[mid,r]|），如果没有，那么在虚儿子上一定没有，因为实儿子代表的后缀是刚加入的最长的后缀，其他儿子的后缀更短，条件更严格。如果有，则在p的线段树上询问[l,mid-1]，再减去实儿子上询问的答案。如果大于0，那么说明我们可以找到两种后缀字符。

时间复杂度预处理O(lgn)，查询O(lg2n)，这一步优化到O(lgn)的方法参见《后缀树结点数》命题报告及一类区间问题的优化-陈江伦。

至于这个lct怎么建，应该是在sam建出后缀树，然后在后缀树上做access？

Q3:维护一个字符串s，支持在s的某个位置插入一个字符串，删除s某一段的字符串。每次询问给出一个询问串p，对于每一个1≤i≤|p|，你需要回答p[:i]在s[l:r]中出现了多少次。用块状链表，把询问转化为广义sam上的三位数点，应该很暴力？？

Q4:给出一个字符串s，O(nlogn)预处理，O(1)回答s[l:r]的最大后缀。

* 查询最小后缀

引理:设s[p..n]是s[i..n],(l≤i≤r)中最小者,则minsuf(l,r)等于s[p..r]的最短非空border。

因为最短非空border的长度不超过串长的一半。所以我们先求t=min{s[i:]|l≤i≤r}，minsuf(s[l:r])=min(t,minsuf(s[r-m+1:r]))(m=⌊lg(r-l+1)⌋)，对于所有r,m计算即可。

* 查询最大后缀

模仿最小后缀的方法，构造maxsuf(s[l:r])=max(x,maxsuf(s[r-m+1:r]))(m=⌊lg(r-l+1)⌋)，我们要求出这个x，即s[x..r]=min{s[i..r]| l≤x<r-m+1}。

引理:设s[p1..n]是s[i..n],(l≤i<r-m+1)中最大的。则s[p1..r]是s[x..r]的前缀。

若l=p1，则x之只能是p1。

若l<p1，设s[p2..n]是s[i..n],(l≤i<p1)中最大的。如果s[p1..r]>s[p2..r]则x=p1，因为s[p2..r]比s[p1..r]更长，而s[p1..r]>s[p2..r]说明这两个串的公共前缀不足|s[p1..r]|，且lcp后面那个字符更小，即不能在p1前找到一个x，使得s[p1..r]是s[x..r]的前缀。那么由于 s[p1 :] 是 s[p2 :] 的前缀，s[p2 :] 是 s[x..r] 的前缀因此 s[x..r]有 p2 - p1 的周期。于是只要求 s[1..p2 - 1] 和 s[1..p1 - 1]的最长公共后缀，得出周期延伸至何处即可。

Q5：定义 suf(s) = {s[i :]|1 ≤ i ≤ |s|} 同理 pre(s) = {s[: i]|1 ≤ i ≤ |s|} 。定义 τ (s, t) = suf(s) ∩ pre(t) ，给出 n 个字符串 s1, s2, . . . sn ，回答 si[k :] 是否在 τ (si, sj) 中。需要发现令 τ (s, t) 中最长的字符串为 p1 ，其余的字符串都是 p1 的 border。根据之前的定理 p1 的 border 形成了 O(log n) 个等差数列容易检查。求解p1的话我只会在sam上暴力，求dalao教我原文中的方法。

余下的Q6~Q9过于神仙，留坑待填。

#### 简单的连续段数据结构

蒟蒻只能看懂引入中的那道题——Codeforces 997E。

给定一个n的排列，每次询问[l,r]之间的所有子区间有多少个是好区间，即包含连续的len个数。转化好区间的定义为max-min-(r-l)=0的区间，建立线段树，从左到右枚举r，在线段树上叶子结点x表示[x,r]这个区间的最小值以及最小值的个数，非叶子结点为其儿子的信息和，在每次移动r时用单调栈维护出那些区间[l,r]需要修改其最小/大值，在线段树上打标记完成。至此我们对一个询问可以求出[ql..qr,qr]这些区间中有多少好区间。现在询问的是所有子区间，一个O(n^2)的做法是每次移动端点时在线段树上下放所有标记，若最小值为0，将当前的最小值个数记到这个点的总答案里去。但我们发现做这个事情也是可以lazy标记解决的，即用一个标记表示当前的最小值个数要记到这个点的答案中几次。因为最小值的改变和累计到答案中是一起做的，所以没有问题，询问时对[ql,qr]做区间查询即可。

后面的内容emmm，每个字都认识，连起来我怎么就看不懂了呢？

#### 浅谈可追溯化数据结构

定义：对于一个数据结构，维护一个操作序列，支持在某个位置插入一个操作，或删除一个操作，以及对按顺序执行这些操作后的状态 进行一些询问。实现这些功能，就称为原数据结构的部分可追溯化(Partial Retroactivity)。如果还支持对操作序列的任意一个前缀进行询问，就称为原数据结构的完全可追溯化(Full Retroactivity)。

完全可追溯化并查集

LCT维护并查集，边(u,v,w)表示(u,v)最早在w时刻并入同一集合，注意这个并查集不要求删除操作。

部分可追溯化优先队列

 定义的桥是什么？留坑待填。

#### 模拟费用流问题

SPFA的hack方法





Problem 1

定义f[i][j]表示前i个位置，其中有j个洞还没有匹配的最小代价和。将坐标转化为代价，即洞的代价为-x[i]，老鼠代价为x[i]，则匹配的代价就是两者之和。

得到转移——如果 i 为老鼠，则 f[i][j] = f[i - 1][j + 1] + x[i]， 如果 i 为洞，则 f[i][j] = min(f[i - 1][j - 1] - x[i], f[i - 1][j])。因为dp从左往右进行，所以x[i]单调递增，那么对于i-1时刻的状态，不带i选j个一定没有带上i选j个优，第二个转移就变成了确定的式子，用栈维护即可。

Problem 2

此时定义f[i][j]表示前i个位置，其中有-j个老鼠（j>0时可以理解为栈里放的是洞，即缺少j个老鼠）的最小代价和。发现最优情况可以做到任意两对匹配均不相交，那么这个dp在转移的时候仅当i为洞且j=0时需要决策这个洞用不用，其他位置只要对dp数组打整体加和平移的标记即可。

Problem 3

似乎是修改一下代价的计算方式，用平衡树来维护。

堆的应用

伪代码参考讲义第69页（f0表示当前的答案，前面凸函数是堆正确性的证明？）。

此时我们扫到一个点的时候，先为它做一个匹配，但目前的状态可能不是全局最优解，所以要让操作变得可以撤回。

具体来说，对老鼠和洞分别维护两个堆，堆内元素v表示把对应的物品解除配对关系并与当前物品配对后答案的变化值。在插入一个物品的时候与堆顶元素配对，注意洞的配对是有选择的，要判断是否使答案更优。

Problem 4

每个洞有不止一个容量的时候，因为时间复杂度和容量上限有关，或者说拆成多个容量为1的洞，就不能保证时间复杂度。

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，求一次答案 S1。

第二遍，我们强制每只老鼠只能往右，求一次答案 S2。

对于原问题，一定存在一组最优解满足这组解所选取的洞 ⊆ min(S1 + S2, V)，V 表示原来洞的集合。

现在对min(S1 + S2, V)求解原问题，洞的总数就是线性的了。

还有一种简单的方法是在堆内维护流量的时候，使用一个pair，表示增广的代价和流量，在增广的时候对流量进行操作，为0时从堆中弹出，代码参考UOJ455。

Problem 5 |Problem 6

在老鼠有分身或洞有附加费用的情况下依然可以套用堆的做法，因为匹配不交叉，可证复杂度不变。

Problem 7

把模型放到树上，使用可并堆维护。

Problem 8

UOJ455，代码可以参考<http://uoj.ac/submission/321266>，感性理解下计算答案的方法。

Tips:

快递员坐标Xi，餐厅坐标Yi。

在插入餐厅时，因为有容量限制，所以使用while，直到用完所有流量。有配对的餐厅在while时计算代价插入餐厅堆中，没有配对的将-yi+wi插入餐厅堆中。与这个餐厅配上对的快递员在while结束后一并插入快递员堆中，没用上的原样放回快递员堆中。这四种类型对应代码中4个push操作，考虑一下代价即可。

由于本人水平不行，所以后面具体数学，简单数论算法，生成函数的三份讲义先鸽了。

#### 题表 ~~做题是不可能做的，就口胡下这个样子~~

Codeforces 997E Good Subsegments

51nod 1517 递推、平方根与误差

bzoj5342 「CTSC2018」青蕈领主

后缀树节点数

BZOJ 3625 小朋友和二叉树

UOJ455 【UER #8】雪灾与外卖