**后缀数组**

今天拜读了罗穗骞的神作[《后缀数组——处理字符串的有力工具》](http://d.download.csdn.net/down/1529037/liguoying07)  
这篇主要是模板以及应用,还有一篇[许智磊的《后缀数组》](http://home.ustc.edu.cn/~zhuhcheng/ACM/suffix_array.pdf)主要讲概念和证明  
两者结合疗效好- -  
  
刚刚开始做的时候觉得不过尔尔,模板题而已  
但越做到后边[PKU3693](http://acm.pku.edu.cn/JudgeOnline/problem?id=3693)越发觉有趣,非常巧妙的统计,随便把RMQ也学了  
  
跟着论文做了几道题目,但还不知道以后出现后缀数组的题能不能解出来,呵呵  
刷题途中发现3xian总是排第一—-无限崇拜并馋涎其模板,发现他的最大流,(kth number)图灵树,后缀树模板极其的高效  
  
献上模板,倍增算法nlog(n),基本上时仿照论文里写的,加了点常数优化  
另外一个DC3编程复杂度大,并且效率只是提高一点点,不是严格意义上的线性算法,没写(其实是我偷懒- -)

[?](http://www.ericbess.com/ericblog/2008/03/03/wp-codebox/#examples)[[Copy to clipboard]View Code](javascript:;) CPP

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | //rank从0开始  //sa从1开始,因为最后一个字符(最小的)排在第0位  //high从2开始,因为表示的是sa[i-1]和sa[i]  #define M 220000  int rank[M],sa[M],X[M],Y[M],high[M],init[M];  int buc[M];  void calhigh(int n) {  int i , j , k = 0;  for(i = 1 ; i <= n ; i++) rank[sa[i]] = i;  for(i = 0 ; i < n ; high[rank[i++]] = k)  for(k?k--:0 , j = sa[rank[i]-1] ; init[i+k] == init[j+k] ; k++);  }  bool cmp(int \*r,int a,int b,int l) {  return (r[a] == r[b] && r[a+l] == r[b+l]);  }  void suffix(int n,int m = 128) {  int i , l , p , \*x = X , \*y = Y;  for(i = 0 ; i < m ; i ++) buc[i] = 0;  for(i = 0 ; i < n ; i ++) buc[ x[i] = init[i] ] ++;  for(i = 1 ; i < m ; i ++) buc[i] += buc[i-1];  for(i = n - 1; i >= 0 ; i --) sa[ --buc[ x[i] ]] = i;  for(l = 1,p = 1 ; p < n ; m = p , l \*= 2) {  p = 0;  for(i = n-l ; i < n ; i ++) y[p++] = i;  for(i = 0 ; i < n ; i ++) if(sa[i] >= l) y[p++] = sa[i] - l;  for(i = 0 ; i < m ; i ++) buc[i] = 0;  for(i = 0 ; i < n ; i ++) buc[ x[y[i]] ] ++;  for(i = 1 ; i < m ; i ++) buc[i] += buc[i-1];  for(i = n - 1; i >= 0 ; i --) sa[ --buc[ x[y[i]] ] ] = y[i];  for(swap(x,y) , x[sa[0]] = 0 , i = 1 , p = 1 ; i < n ; i ++)  x[ sa[i] ] = cmp(y,sa[i-1],sa[i],l) ? p-1 : p++;  }  calhigh(n-1);//后缀数组关键是求出high,所以求sa的时候顺便把rank和high求出来  }      //当需要反复询问两个后缀的最长公共前缀时用到RMQ  int Log[M];  int best[20][M];  void initRMQ(int n) {//初始化RMQ  for(int i = 1; i <= n ; i ++) best[0][i] = high[i];  for(int i = 1; i <= Log[n] ; i ++) {  int limit = n - (1<<i) + 1;  for(int j = 1; j <= limit ; j ++) {  best[i][j] = min(best[i-1][j] , best[i-1][j+(1<<i>>1)]);  }  }  }  int lcp(int a,int b) {//询问a,b后缀的最长公共前缀  a = rank[a]; b = rank[b];  if(a > b) swap(a,b);  a ++;  int t = Log[b - a + 1];  return min(best[t][a] , best[t][b - (1<<t) + 1]);  }      int main() {  //预处理每个数字的Log值,常数优化,用于RMQ  Log[0] = -1;  for(int i = 1; i <= M ; i ++) {  Log[i] = (i&(i-1)) ? Log[i-1] : Log[i-1] + 1 ;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // n为数组长度,下标0开始  // 将初始数据,保存在init里,并且保证每个数字都比0大  // m = max{ init[i] } + 1  // 一般情况下大多是字符操作,所以128足够了  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  init[n] = 0;  suffix(n+1,m);    initRMQ(n);  } |

下边收入几道论文上没提的后缀数组题目  
[zoj3199 Longest Repeated Substring](http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3199)  
连续重复至少1次的最长串  
[spoj1811 Longest Common Substring](http://www.spoj.pl/problems/LCS/)  
普通的LCS,同PKU2774,不过数据量超大,上边的模板超时,DC3刚好卡过,据说正解是后缀自动机,Orz  
[hdu2890 Longest Repeated subsequence](http://acm.hdu.edu.cn/showproblem.php?pid=2890)  
不可重叠k次最长重复子串.分层后排序贪心nlog(n)