个人总结

陈铭煊 Max.D.

本周是比较摸的一周，主要学习了一下广义后缀自动机（不敢说自己已经会了），稍微写了下FWT的知识库。

龙鹏宇 Hardict

肖思炀 MountVoom

这周是特别摸的一周，补了补题，没学啥新东西。

摸了一套div.4和一套div.3，等明天再摸一套div.2

爬去写作业了。

本周推荐

陈铭煊 Max.D.

主要介绍一下rope这个容器rope并不在C++标准库中，而是在GNU g++的ext扩展库系列中，和pb_ds是属于平级关系C++98环境也一样可以使用（注意：如果不是g++编译器自然无法使用）。

rope的主要目标，是对一个经过多次拼切、修改、替换而来的超长字符串进行维护。当然也可以用于普通的序列维护（为了体现字符串的地位，`rope<char>`被特化成了`crope`类型）。

rope对于序列的插入、删除、替换、拼接、切片等操作快速到了$\log n$级别。这是vector或者string所不具备的，因为后者的这些操作，基本上都要对元素进行整体移位。可想而知，其内部是使用了某种BST的结构BST的每一个结点，代表了一个序列区间，而子节点则进一步划分成了两个子区间——我们自己手写像SplayTreap这样的BST也能比较轻松地达到rope的要求。

rope的另外一大能力，是进行**可持久化**，或者说“写时复制”——在rope进行复制或者部分复制时BST会尽量利用原序列已有的部分，在对新串进行修改时，才新建部分节点。这种可持久化，能够帮助我们实现一些手写比较困难的数据结构，例如**洛谷3369 可持久化平衡树**，代码详见个人主页。

但rope的缺点也是显而易见的——没有办法对区间添加额外的域进行维护（比如没有办法快速求区间和，没有办法进行快速区间翻转，这里快速是指的$\log n$级别），另外常数也是比手写的平衡树大了不少。这些场合还是建议手写（当然pb_ds中的BST可以弥补这一点，但是额外的代码不必手写简单多少）。

一些常见的操作如下：

```
R.push_back(a) //往后插入，也可以用+=
```

```
R.insert(pos,a)//在pos位置插入a□pos是一个迭代器。
R.erase(pos,n)//在pos位置删除n个元素。
R.replace(pos,x)//从pos开始替换成x
R.substr(pos,x)//从pos开始提取x个。
```

其他还有相当多操作不便于列举。大家可以参看网址SGI Rope（没错□rope是来源于SGI STL的，g++基本上是将其拷贝过来了）。其中实现的原理，参考维基百科

龙鹏宇 Hardict

$平面图边数至多为3|V|-6(V+F-E=2,F至多2|V|-4)$

含递推关系的期望问题

$对递推\sum_{i=0}^{n} a_{i}f_{m-i}=0,(a_{0}一般为0以求解f_{m})
递推的特征多项式为\sum_{i=0}^{n}a_{i}x^{n-i}=0,联接多项式为\sum_{i=0}^{n}a_{i}x^{i}=0$

$P(x)=\sum_{i=0}^{\infty} p_{i}x^{i},p_{i}为概率i,p_{i}具有线性递推关系,那么期望即P^{'}(1)$

$考虑H(x)=P(x)G(x),G(x)为递推的联接多项式,后|G(x)|项都应为0$

$P^{'}(1)=\frac{H^{'}(1)G(1)-H(1)G^{'}(1)}{G(1)^{2}}$

$实际计算时,若P(x)截取2n项,G(x)由BM算法不超过n项,H(x)也应该只截取前2n项$

肖思炀 MountVoom

不知道推荐什么好，就推荐一道数据结构题吧。

题解