Rope(不能维护数值相关的可持久化平衡树)

```
#include<bits/stdc++.h>
#include<ext/rope>
#define N 100005
using namespace std;
using namespace __gnu_cxx;
rope<char>t,tmp,A,B,his[N];//rope 从 0 开始存储
rope<int>s;
int main()
{
   int x,pos;
   t=A+B;//A,B 合并为 t
   t.push_back(x);//在末尾插入元素 x
   tmp=t.substr(pos,x);//提取 pos 开始 x 个
   t.replace(pos,x);//pos 开始换成 x
   t.erase(pos,x);//pos 开始删除 x 个
   his[i]=new rope<char>(*his[i-1]);//可持久化 第 i-1 个->第 i 个
   cout<<t<<endl;//输出所有字符（必须开 char）
   for(int i=0;i<n;i++)cout<<s[i]<<' ';cout<<endl;//输出所有数字（int,ll）
   cout<<t[pos]<<endl;//输出第 pos+1 个
   return 0;
}
```

Unordered_map(哈希表)

```
#include<bits/stdc++.h>
#include<tr1/unordered_map>
#define iter unordered_map<int,int>::iterator
using namespace std;
using namespace std::tr1;
unordered_map<int,int>w;
int main()
{
   int a,b;
   for(iter it=w.begin();it!=w.end();++it)
      a=(*it).first,b=(*it).second;
   return 0;
}
```

二分

```
//不用-1
pos=lower_bound(s+1,s+n+1,x)-s;//按从小到大，x 最少能插入到哪个位置
pos=upper_bound(s+1,s+n+1,x)-s;//按从小到大，x 最多能插入到哪个位置
```

离散化

```
sort(s+1,s+n+1);
Tnum=unique(s+1,s+n+1)-s-1;//离散化要-1
```

全排列

next_permutation(s+1,s+n+1);

vector 容器

vector<int>s;
s.push_back(x);
sort(s.begin(),s.end());
for(int i=0;i!=s.size();i++);
lower_bound(t.begin(),t.end(),x) //vector 上二分，x 为数字

pair

pair<int,int>p[N];
p[i]=make_pair(a,b);
sort(p+1,p+n+1); //先按第一维排，后按第二维排
a=p[i].first,b=p[i].second;

堆

priority_queue<int>q;//大根堆
priority_queue<int,vector<int>,greater<int> >q;//小根堆

bitset

bitset<N>s(M);
//N 为 bitset 的长度
//M 为 bitset 中初始的数字（转化成二进制放到每一位）
//如果 M 大小超过 N 的话取二进制后 N 位
//没有（M）等价于（0）

s<<=x; //s 整体左移 x 位 空的位置补 0 >>同理
s|=x s^=x s&=x (x 为数字) //s 搞成二进制数或 x 再搞成数组
s1|=s2 s1^=s2 s1&=s2
//两个 bitset 直接进行位运算
//"|"为求并集 "&"为求交集

s.size() //s 数组的大小
s.reset() //全变 0
s.set() //全变 1
s.flip() //全取反，求补集

for(int i=res._Find_first();i<=n;i=res._Find_next(i)) //遍历 bitset 中所有 1
._Find_fitst 找第一个 1 ._Find_next(i) 找 i 之后的第一个 1