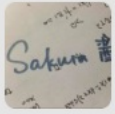
虫逢

范浩强
IIIS, Tsinghua

又是小强阿米巴

虫逢	fanhqme(范…	ioitrai…	智商	10 組



范爷题目的第一句话永远都是：这个题是老子出的，智商低勿入

看到这个提示
感觉我的题均分有望上70辣！

题目描述

给一个M元集的2N个大小为L子集，每个子集有恰好另一个子集和它的交是L/2。数据是随机造的。

$N=M=L^2$。N最大16900。

选手讨论

请最高分、自认为最有意思的解法的同学上来说一下。

正解

时间复杂度O(NL)：和读入一样快

MinHash Algorithm

随机取一个hash函数。计算每个子集中元素的hash值的最小值。

f(S)=min(map(h,S))

分析

考虑两个集合U，V

f(U)==f(V)的概率：|U∩V|/|U∪V|

同源的两个虫：1/3；不同源的虫：1/L

分析

同时取两个hash

f(S)=(min(map(h1,S)),min(map(h2,S)))

同源的虫：1/9；不同源的虫：$1/L^2=1/N$

算法框架

递归

每次，随机取h1，h2，计算出所有集合的hash

把hash相同的集合放在一起，暴力两两判断

判断一轮的时间是O(NL)，可以把N减小1/9

总共的时间复杂度就是O(9NL)=O(NL)

实现细节

离散化，建立倒排索引。使用随机排列作为hash。

可以每次只用一个hash。按照sorted(map(h, S))的字典序排序，然后暴力判断字典序相邻的两个能否配对。

时间复杂度不变，但是能更优雅的处理不同的N和L的关系。

瓶颈

利用倒排索引进行排序会导致大量的cpu缓存缺失(cache miss)，使得常数增大。

解决方法：只考虑hash值前3N/L小的数

两两比较（类似归并排序）时if语句产生大量的cpu分支跳转预测错误，使得常数增大。

解决方法：快速排除掉非同源的虫。两阶段判断。

结论：把除数据读入和倒排索引的所有其他部分的时间复杂度控制在o(NL)

代码

```
#include
#include <algorithm>
using namespace std;
int N,M,L;
int * lines, *lines_old;
const int HMax=1000007;
int hash[HMax],*hash_next,hash_n;
int *hash_key;
int * perm;
int * belong, * todo;
int * wordcnt, * wordlocation;
bool comp(int id1,int id2){
 for (int i=0;i<L;i++){
  if (lines[id1*L+i]!=lines[id2*L+i])
   return lines[id1*L+i]<lines[id2*L+i];
 }
 return false;
}
int getId(int key){
 int h=key%HMax;
 if (h<0)
  h+=HMax;
 for (int i=hash[h];i!=-1;i=hash_next[i])
  if (hash_key[i]==key)
   return i;
 hash_key[hash_n]=key;
 hash_next[hash_n]=hash[h];
 hash[h]=hash_n;
 return hash_n++;
}
```

代码

```
int main(){
 scanf("%d%d%d",&N,&M,&L);N*=2;
 lines=new int[N*L];
 getchar();fread(lines,N*L,sizeof(lines[0]),stdin);
 for (int i=0;i<HMax;i++)hash[i]=-1;
 hash_key=new int[min(N*L,M)];hash_next=new int[min(N*L,M)];
 hash_n=0;
 wordcnt=new int[M+1];wordlocation=new int[N*L];
 for (int i=0;i<M;i++)wordcnt[i]=0;
 for (int i=0;i<N*L;i++)wordcnt[lines[i]=getId(lines[i])]++;
 M=hash_n;
 for (int i=1;i<M;i++)wordcnt[i]+=wordcnt[i-1];
 for (int i=0;i<N;i++)
  for (int j=0;j<L;j++)
   wordlocation[--wordcnt[lines[i*L+j]]]=i;
 wordcnt[M]=N*L;
 perm=new int[M];
 for (int i=0;i<M;i++)perm[i]=i;
 belong=new int[N];
 for (int i=0;i<N;i++)belong[i]=-1;
 todo=new int[N];
 for (int i=0;i<N;i++)todo[i]=i;
 int ntodo=N;
 int * curpos=new int[N];
 for (int i=0;i<N;i++)curpos[i]=i*L;
 lines_old=new int[N*L];
 for (int i=0;i<M;i++)
  for (int j=wordcnt[i];j<wordcnt[i+1];j++){
   int t=wordlocation[j];
   lines_old[curpos[t]++]=i;
```

代码

```
while (ntodo>0){
random_shuffle(perm,perm+M);
for (int i=0;i<N;i++)curpos[i]=i*L;
 for (int j=wordcnt[perm[i]];j<wordcnt[perm[i]+1];j++)
for (int i=0;i<min(max(100,M/L*3),M);i++)
  lines[curpos[wordlocation[j]]++]=perm[i];
sort(todo,todo+ntodo,comp);
int ntodo_new=0;
for (int i=0;i<ntodo;i++){
 int s=0,x1=todo[i],x2=-1;
 if (i+1<ntodo){
  x2=todo[i+1];
  for (int j=0,k=0;j<L && k<L;){
   if (lines_old[x1*L+j]==lines_old[x2*L+k])
    s++,j++,k++;
   else if (lines_old[x1*L+j]<lines_old[x2*L+k])j++;
   else k++;
   if (ntodo>=max(100,N/10) && (j>=5 && s==0))break;
  }
 }
 if (s>=L/2){
  belong[x1]=x2;
  belong[x2]=x1;
  i++;
 }else todo[ntodo_new++]=x1;
}
ntodo=ntodo_new;
}
for (int i=0;i<N;i++)printf("%d\n",belong[i]+1);
return 0;
```