|  |
| --- |
|  |
| OI模板整理【入门篇】 |
| 附部分例题及题解 |
|  |
| **轻飏** |
|  |

|  |
| --- |
|  |

前言

本文档是为熟悉OI常用模板以及快速索引到相关例题进行练手而编写的。

本文档预期按照难度分为三个部分，分别为入门模板，进阶模板以及高级模板。入门模板作为基础知识，后面的部分模板是以前面一些入门模板，进阶模板作为铺垫的。例如求解网络流最大流问题的Dinic算法中用到类SPFA的写法。目前本文档只有入门模板部分，关于进阶数据结构（如线段树，树状数组等）以及高级的一些模板（莫比乌斯反演，Splay）正在更新。

本文档力求把相关模板代码写的通俗易懂，同时提供变化多样而又不离本宗的典型例题或对于同一问题的不同写法。

这是本文档的第一版本，只添加了作者认为比较常用有用的模板和例题，[如果有读者有更好的建议的话可以联系](mailto:如果有读者有更好的建议的话可以联系GDOI_qingyang@163.com)GDOI\_qingyang@163.com。

本文档封面设计方案部分由“竹子”提供，特此感谢。

**第一部分：入门模板**

工欲善其事，必先利其器。

PS：不要因为学了高级算法而忘记基本的算法。

第一部分提到了比较重要的排序——归并排序；一些有用的简单数据结构——并查集，最小生成树；以及一些常见的问题以及解决算法：单源最短路，快速幂取模，杨辉三角的应用（以“组合数计算”为例），拓展欧几里得等。

**§1 归并排序-用法1：排序（Luogu1177）**

**一句话评析：归并排序在的复杂度内使序列有序。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 100005

**using** **namespace** std**;**

int num**[**maxn**],** tmp**[**maxn**];**

void MergeArray**(**int left**,** int mid**,** int right**){**

int i **=** left**,** j **=** mid**,** m **=** mid**+**1**,** n **=** right**,** k **=** 0**;**

**while(**i **<=** j **&&** m **<=** n**)**

num**[**i**]** **<=** num**[**m**]** **?** tmp**[**k**++]** **=** num**[**i**++]** **:** tmp**[**k**++]** **=** num**[**m**++];**

**while(**i **<=** j**)**

tmp**[**k**++]** **=** num**[**i**++];**

**while(**m **<=** n**)**

tmp**[**k**++]** **=** num**[**m**++];**

**for(**int i **=** 0**;** i **<** k**;** i**++)** num**[**left**+**i**]** **=** tmp**[**i**];**

**}**

void MergeSort**(**int left**,** int right**){**

**if(**left **>=** right**)** **return;**

int mid **=** **(**left**+**right**)>>**1**;**

MergeSort**(**left**,** mid**);**

MergeSort**(**mid**+**1**,** right**);**

MergeArray**(**left**,** mid**,** right**);**

**}**

int main**(**void**)**

**{**

int n**;** scanf**(**"%d"**,** **&**n**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** scanf**(**"%d"**,** **&**num**[**i**]);**

MergeSort**(**1**,** n**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** printf**(**"%d "**,** num**[**i**]);**

**return** 0**;**

**}**

**§1 归并排序-用法2：有序合并（Luogu1309）**

**一句话评析：归并排序在O(n)的复杂度内合并有序的两个序列。**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <algorithm>

#define maxn 200005

**using** **namespace** std**;**

int N**,** R**,** Q**;**

int Power**[**maxn**];**

struct Athlete**{**

int score**;**

int number**;**

**}**P**[**maxn**],** W**[**maxn**],** L**[**maxn**];**

bool cmp**(**const Athlete **&**A**,** const Athlete **&**B**){**

**if(**A**.**score **==** B**.**score**)**

**return** A**.**number **<** B**.**number**;**

**return** A**.**score **>** B**.**score**;**

**}**

void Merge**(){**

int i **=** 1**,** j **=** 1**,** k **=** 1**;**

**while(**i **<=** N **&&** j **<=** N**)**

**{**

**if(**cmp**(**W**[**i**],** L**[**j**]))**

**{**

P**[**k**].**score **=** W**[**i**].**score**;**

P**[**k**++].**number **=** W**[**i**++].**number**;**

**}**

**else**

**{**

P**[**k**].**score **=** L**[**j**].**score**;**

P**[**k**++].**number **=** L**[**j**++].**number**;**

**}**

**}**

**while(**i **<=** N**)**

**{**

P**[**k**].**score **=** W**[**i**].**score**;**

P**[**k**++].**number **=** W**[**i**++].**number**;**

**}**

**while(**j **<=** N**)**

**{**

P**[**k**].**score **=** L**[**j**].**score**;**

P**[**k**++].**number **=** L**[**j**++].**number**;**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d %d"**,** **&**N**,** **&**R**,** **&**Q**);**

**for(**int i **=** 1**;** i **<=** N**\***2**;** i**++)** scanf**(**"%d"**,** **&**P**[**i**].**score**);**

**for(**int i **=** 1**;** i **<=** N**\***2**;** i**++)** scanf**(**"%d"**,** **&**Power**[**i**]);**

**for(**int i **=** 1**;** i **<=** N**\***2**;** i**++)** P**[**i**].**number **=** i**;**

sort**(**P**+**1**,** P**+**N**\***2**+**1**,** cmp**);**

**for(**int k **=** 1**;** k **<=** R**;** k**++)**

**{**

int cnt **=** 1**;**

**for(**int i **=** 1**;** i **<=** N**\***2**;** i**+=**2**)**

**{**

**if(**Power**[**P**[**i**].**number**]** **>** Power**[**P**[**i**+**1**].**number**])**

**{**

W**[**cnt**].**score **=** P**[**i**].**score**+**1**;**

W**[**cnt**].**number **=** P**[**i**].**number**;**

L**[**cnt**].**score **=** P**[**i**+**1**].**score**;**

L**[**cnt**++].**number **=** P**[**i**+**1**].**number**;**

**}**

**else**

**{**

W**[**cnt**].**score **=** P**[**i**+**1**].**score**+**1**;**

W**[**cnt**].**number **=** P**[**i**+**1**].**number**;**

L**[**cnt**].**score **=** P**[**i**].**score**;**

L**[**cnt**++].**number **=** P**[**i**].**number**;**

**}**

**}**

Merge**();**

**}**

printf**(**"%d\n"**,** P**[**Q**].**number**);**

**return** 0**;**

**}**

**§1 归并排序-用法3：逆序对（Luogu1908）**

**一句话评析：归并的过程可以统计出逆序对的数量。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 1000005

#define int long long

**using** **namespace** std**;**

int num**,** ans**;**

int tmp**[**maxn**],** order**[**maxn**];**

void Merge**(**int left**,** int right**){**

**if(**left **==** right**)** **return;**

int mid **=** **(**left**+**right**)>>**1**;**

Merge**(**left**,** mid**);**

Merge**(**mid**+**1**,** right**);**

int i **=** left**,** j **=** mid**+**1**,** k **=** left**;**

**while(**i **<=** mid **&&** j **<=** right**)**

**{**

**if(**order**[**i**]** **>** order**[**j**])**

tmp**[**k**++]** **=** order**[**j**++],** ans **+=** mid**-**i**+**1**;**

**else**

tmp**[**k**++]** **=** order**[**i**++];**

**}**

**while(**i **<=** mid**)** tmp**[**k**++]** **=** order**[**i**++];**

**while(**j **<=** right**)** tmp**[**k**++]** **=** order**[**j**++];**

**for(**int i **=** left**;** i **<=** right**;** i**++)** order**[**i**]** **=** tmp**[**i**];**

**}**

signed main**(**void**)**

**{**

scanf**(**"%lld"**,** **&**num**);**

**for(**int i **=** 1**;** i **<=** num**;** i**++)** scanf**(**"%lld"**,** **&**order**[**i**]);**

Merge**(**1**,** num**);**

printf**(**"%lld\n"**,** ans**);**

**return** 0**;**

**}**

**§2 最小生成树-用法1：连通性判断（Luogu3366）**

**一句话评析：如果不能建成一棵树则说明图不连通，有时可配合二分解题。**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <algorithm>

#define maxn 1000005

**using** **namespace** std**;**

int n**,** m**;**

int ans**,** cnt**;**

int uset**[**maxn**];**

struct Edge**{**

int u**,** v**,** w**;**

**}**E**[**maxn**];**

int find**(**int x**){**

**return** x **==** uset**[**x**]** **?** uset**[**x**]** **:** uset**[**x**]** **=** find**(**uset**[**x**]),** uset**[**x**];**

**}**

bool cmp**(**const Edge **&**A**,** const Edge **&**B**){**

**return** A**.**w **<** B**.**w**;**

**}**

void Kruskal**(**int n**,** int m**){**

sort**(**E**+**1**,** E**+**m**+**1**,** cmp**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int u **=** find**(**E**[**i**].**u**),** v **=** find**(**E**[**i**].**v**);**

**if(**u **==** v**)** **continue;**

ans **+=** E**[**i**].**w**;**

uset**[**v**]** **=** u**;** cnt**++;**

**if(**cnt **==** n**-**1**)** **break;**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** uset**[**i**]** **=** i**;**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

scanf**(**"%d %d %d"**,** **&**E**[**i**].**u**,** **&**E**[**i**].**v**,** **&**E**[**i**].**w**);**

Kruskal**(**n**,** m**);**

cnt **!=** n**-**1 **?** printf**(**"orz\n"**)** **:** printf**(**"%d\n"**,** ans**);**

**return** 0**;**

**}**

**§2 最小生成树-用法2：最优性判断（Luogu1967）**

**一句话评析：贪心答案一定在最大生成树上，最（大）小生成树相比只是顺序有异。**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <algorithm>

#define maxn 500005

**using** **namespace** std**;**

int n**,** m**,** q**;**

int uset**[**maxn**];**

int head**[**maxn**],** k **=** 0**;**

int pre**[**maxn**],** ID**[**maxn**],** val**[**maxn**],** vis**[**maxn**],** cnt **=** 0**;**

int siz**[**maxn**],** son**[**maxn**],** top**[**maxn**],** fat**[**maxn**],** dep**[**maxn**];**

struct ctree**{**

int u**;**

int v**;**

int w**;**

**}**t**[**maxn**];**

struct Edge**{**

int to**;**

int next**;**

int weight**;**

**}**Edge**[**maxn**];**

void init**(){**

k **=** 0**;**

memset**(**head**,** **-**1**,** **sizeof(**head**));**

**}**

void Build**(**int u**,** int v**,** int w**){**

Edge**[**k**].**to **=** v**;**

Edge**[**k**].**weight **=** w**;**

Edge**[**k**].**next **=** head**[**u**];**

head**[**u**]** **=** k**++;**

**}**

/\*--------------Kruskal--------------\*/

int find**(**int x**){**

**if(**uset**[**x**]** **!=** x**)** uset**[**x**]** **=** find**(**uset**[**x**]);**

**return** uset**[**x**];**

**}**

bool cmp**(**const ctree **&**A**,** const ctree **&**B**){**

**return** A**.**w **>** B**.**w**;**

**}**

void Kruskal**(){**

sort**(**t**+**1**,** t**+**m**+**1**,** cmp**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int x **=** find**(**t**[**i**].**u**);**

int y **=** find**(**t**[**i**].**v**);**

**if(**x **!=** y**)**

**{**

uset**[**x**]** **=** y**;**

Build**(**t**[**i**].**u**,** t**[**i**].**v**,** t**[**i**].**w**);**

Build**(**t**[**i**].**v**,** t**[**i**].**u**,** t**[**i**].**w**);**

**}**

**}**

**}**

/\*--------------SegTree--------------\*/

struct STree**{**

int left**;**

int right**;**

int value**;**

**}**tree**[**maxn**];**

void Build\_STree**(**int x**,** int left**,** int right**){**

tree**[**x**].**left **=** left**;**

tree**[**x**].**right **=** right**;**

**if(**left **==** right**)**

**{**

tree**[**x**].**value **=** val**[**pre**[**left**]];**

**return;**

**}**

int mid **=** **(**left**+**right**)>>**1**;**

Build\_STree**((**x**<<**1**),** left**,** mid**);**

Build\_STree**((**x**<<**1**)+**1**,** mid**+**1**,** right**);**

tree**[**x**].**value **=** min**(**tree**[(**x**<<**1**)].**value**,** tree**[(**x**<<**1**)+**1**].**value**);**

**}**

int Query**(**int x**,** int left**,** int right**){**

**if(**tree**[**x**].**left **==** left **&&** tree**[**x**].**right **==** right**)** **return** tree**[**x**].**value**;**

int mid **=** **(**tree**[**x**].**left**+**tree**[**x**].**right**)>>**1**;**

**if(**mid **<** left**)** **return** Query**((**x**<<**1**)+**1**,** left**,** right**);**

**else** **if(**mid **>=** right**)** **return** Query**((**x**<<**1**),** left**,** right**);**

**else** **return** min**(**Query**((**x**<<**1**),** left**,** mid**),** Query**((**x**<<**1**)+**1**,** mid**+**1**,** right**));**

**}**

int Getans**(**int x**,** int y**){**

int ans **=** **(**1**<<**30**);**

**while(**top**[**x**]** **!=** top**[**y**])**

**{**

**if(**dep**[**top**[**x**]]** **<** dep**[**top**[**y**]])** swap**(**x**,** y**);**

ans **=** min**(**ans**,** Query**(**1**,** ID**[**top**[**x**]],** ID**[**x**]));**

x **=** fat**[**top**[**x**]];**

**}**

**if(**x **==** y**)** **return** ans**;**

**if(**dep**[**x**]** **>** dep**[**y**])** swap**(**x**,** y**);**

ans **=** min**(**ans**,** Query**(**1**,** ID**[**x**]+**1**,** ID**[**y**]));**

**return** ans**;**

**}**

/\*--------------Heavy-Light Decomposition--------------\*/

void dfs1**(**int x**,** int father**,** int deep**){**

siz**[**x**]** **=** 1**;**

vis**[**x**]** **=** 1**;**

dep**[**x**]** **=** deep**;**

fat**[**x**]** **=** father**;**

int maxson **=** **-**1**;**

**for(**int i **=** head**[**x**];** **~**i**;** i **=** Edge**[**i**].**next**)**

**{**

**if(**Edge**[**i**].**to **==** father**)** **continue;**

dfs1**(**Edge**[**i**].**to**,** x**,** deep**+**1**);**

siz**[**x**]** **+=** siz**[**Edge**[**i**].**to**];**

val**[**Edge**[**i**].**to**]** **=** Edge**[**i**].**weight**;**

**if(**siz**[**Edge**[**i**].**to**]** **>** maxson**)**

**{**

son**[**x**]** **=** Edge**[**i**].**to**;**

maxson **=** siz**[**Edge**[**i**].**to**];**

**}**

**}**

**}**

void dfs2**(**int x**,** int t**){**

top**[**x**]** **=** t**;**

ID**[**x**]** **=** **++**cnt**;**

pre**[**cnt**]** **=** x**;**

**if(!**son**[**x**])** **return;**

dfs2**(**son**[**x**],** t**);**

**for(**int i **=** head**[**x**];** **~**i**;** i **=** Edge**[**i**].**next**)**

**{**

**if(**Edge**[**i**].**to **==** fat**[**x**]** **||** Edge**[**i**].**to **==** son**[**x**])** **continue;**

dfs2**(**Edge**[**i**].**to**,** Edge**[**i**].**to**);**

**}**

**}**

int main**(**void**)**

**{**

init**();**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

scanf**(**"%d %d %d"**,** **&**t**[**i**].**u**,** **&**t**[**i**].**v**,** **&**t**[**i**].**w**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)** uset**[**i**]** **=** i**;**

Kruskal**();**

int tp **=** 1**;**

**while(**tp **<=** n**)**

**{**

**if(!**vis**[**tp**])** val**[**tp**]** **=** **(**1**<<**30**),** dfs1**(**tp**,** 0**,** 1**),** dfs2**(**tp**,** tp**);**

tp**++;**

**}**

Build\_STree**(**1**,** 1**,** n**);**

scanf**(**"%d"**,** **&**q**);**

**for(**int i **=** 1**;** i **<=** q**;** i**++)**

**{**

int s**,** t**;** scanf**(**"%d %d"**,** **&**s**,** **&**t**);**

**if(**find**(**s**)** **!=** find**(**t**))** printf**(**"%d\n"**,** **-**1**);**

**else** printf**(**"%d\n"**,** Getans**(**s**,** t**));**

**}**

**return** 0**;**

**}**

**§3 并查集-用法1：集合从属（Luogu3367）**

**一句话评析：查询两个元素是否在同一集合只需要对比其集合的代表结点即可。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 200005

**using** **namespace** std**;**

int n**,** m**;**

int uset**[**maxn**];**

int find**(**int x**){**

**return** x **==** uset**[**x**]** **?** uset**[**x**]** **:** uset**[**x**]** **=** find**(**uset**[**x**]),** uset**[**x**];**

**}**

void UnionSet**(**int x**,** int y**){**

x **=** find**(**x**);**

y **=** find**(**y**);**

**if(**x **==** y**)** **return;**

uset**[**x**]** **=** y**;**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** uset**[**i**]** **=** i**;**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int type**,** x**,** y**;**

scanf**(**"%d %d %d"**,** **&**type**,** **&**x**,** **&**y**);**

**if(**type **==** 1**)** UnionSet**(**x**,** y**);**

**else** printf**(**"%c\n"**,** find**(**x**)** **==** find**(**y**)** **?** 'Y' **:** 'N'**);**

**}**

**return** 0**;**

**§3 并查集-用法2：集合统计（Luogu1892）**

**一句话评析：实质就是统计不同的集合，遍历一遍节点即可。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 100005

**using** **namespace** std**;**

int n**,** m**;**

int vis**[**maxn**],** uset**[**maxn**],** Enemy**[**maxn**];**

int find**(**int x**){**

**return** x **==** uset**[**x**]** **?** uset**[**x**]** **:** uset**[**x**]** **=** find**(**uset**[**x**]),** uset**[**x**];**

**}**

void UnionSet**(**int x**,** int y**){**

x **=** find**(**x**);** y **=** find**(**y**);**

**if(**x **==** y**)** **return;**

uset**[**x**]** **=** y**;**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** uset**[**i**]** **=** i**;**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int x**,** y**;** char ch**;**

scanf**(**"%c %d %d\n"**,** **&**ch**,** **&**x**,** **&**y**);**

**if(**ch **==** 'F'**)** UnionSet**(**x**,** y**);**

**else**

**{**

**if(!**Enemy**[**x**])** Enemy**[**x**]** **=** find**(**y**);** **else** UnionSet**(**y**,** Enemy**[**x**]);**

**if(!**Enemy**[**y**])** Enemy**[**y**]** **=** find**(**x**);** **else** UnionSet**(**x**,** Enemy**[**y**]);**

**}**

**}**

int cnt **=** 0**;**

**for(**int i **=** 1**;** i **<=** n**;** i**++)**

**{**

int tmp **=** find**(**i**);**

**if(!**vis**[**tmp**])** vis**[**tmp**]** **=** 1**,** cnt**++;**

**}**

printf**(**"%d\n"**,** cnt**);**

**return** 0**;**

**}**

**§3 并查集-用法3：最小环（Luogu2661）**

**一句话评析：祖先节点相同即构成环，第一个环即最小环，其长度即为答案。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 500005

**using** **namespace** std**;**

int n**,** minn **=** **(**1**<<**30**);**

int dis**[**maxn**],** uset**[**maxn**];**

int find**(**int x**){**

**if(**x **!=** uset**[**x**])**

**{**

int lt **=** uset**[**x**];**

uset**[**x**]** **=** find**(**uset**[**x**]);**

dis**[**x**]** **+=** dis**[**lt**];**

**}**

**return** uset**[**x**];**

**}**

void UnionSet**(**int x**,** int y**){**

int tx **=** find**(**x**),** ty **=** find**(**y**);**

**if(**tx **==** ty**)** minn **=** min**(**minn**,** dis**[**x**]+**dis**[**y**]+**1**);**

**else**

**{**

uset**[**tx**]** **=** ty**;**

dis**[**x**]** **=** dis**[**y**]+**1**;**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d"**,** **&**n**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)** uset**[**i**]** **=** i**;**

**for(**int i **=** 1**;** i **<=** n**;** i**++)**

**{**

int tmp**;** scanf**(**"%d"**,** **&**tmp**);**

UnionSet**(**i**,** tmp**);**

**}**

printf**(**"%d\n"**,** minn**);**

**return** 0**;**

**}**

**§4 单源最短路-算法1：SPFA（Luogu3371）**

**一句话评析：在无负权的图中最好不用，不卡是出题人失职。**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <queue>

#define maxn 2000005

#define int long long

#define maxx 0x3f3f3f3f

**using** **namespace** std**;**

int n**,** m**,** s**;**

int k**,** head**[**maxn**];**

int inq**[**maxn**],** dis**[**maxn**];**

struct Edge**{**

int to**;**

int next**;**

int weight**;**

**}**Edge**[**maxn**];**

void Build**(**int u**,** int v**,** int w**){**

Edge**[**k**].**to **=** v**;**

Edge**[**k**].**weight **=** w**;**

Edge**[**k**].**next **=** head**[**u**];**

head**[**u**]** **=** k**++;**

**}**

void SPFA**(**int s**){**

memset**(**dis**,** 0x3f**,** **sizeof(**dis**));**

queue**<**int**>** q1**;** q1**.**push**(**s**);**

inq**[**s**]** **=** 1**;** dis**[**s**]** **=** 0**;**

**while(!**q1**.**empty**())**

**{**

int tmp **=** q1**.**front**();**

inq**[**tmp**]** **=** 0**;** q1**.**pop**();**

**for(**int i **=** head**[**tmp**];** **~**i**;** i **=** Edge**[**i**].**next**)**

**{**

**if(**dis**[**Edge**[**i**].**to**]** **>** dis**[**tmp**]+**Edge**[**i**].**weight**)**

**{**

dis**[**Edge**[**i**].**to**]** **=** dis**[**tmp**]+**Edge**[**i**].**weight**;**

**if(!**inq**[**Edge**[**i**].**to**])**

**{**

inq**[**Edge**[**i**].**to**]** **=** 1**;**

q1**.**push**(**Edge**[**i**].**to**);**

**}**

**}**

**}**

**}**

**}**

signed main**(**void**)**

**{**

memset**(**head**,** **-**1**,** **sizeof(**head**));**

scanf**(**"%lld %lld %lld"**,** **&**n**,** **&**m**,** **&**s**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int u**,** v**,** w**;**

scanf**(**"%lld %lld %lld"**,** **&**u**,** **&**v**,** **&**w**);**

Build**(**u**,** v**,** w**);**

**}**

SPFA**(**s**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)**

printf**(**"%lld "**,** dis**[**i**]** **<** maxx **?** dis**[**i**]** **:** 2147483647**);**

**return** 0**;**

**}**

**§4 单源最短路-算法2：Dijkstra（Luogu3371）**

**一句话评析：贪心思想，不得负环。不加堆优化复杂度。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 2000005

#define int long long

#define maxx 0x3f3f3f3f

**using** **namespace** std**;**

int n**,** m**,** s**,** mn**;**

int k**,** head**[**maxn**];**

int vis**[**maxn**],** dis**[**maxn**];**

struct Edge**{**

int to**;**

int next**;**

int weight**;**

**}**Edge**[**maxn**];**

void Build**(**int u**,** int v**,** int w**){**

Edge**[**k**].**to **=** v**;**

Edge**[**k**].**weight **=** w**;**

Edge**[**k**].**next **=** head**[**u**];**

head**[**u**]** **=** k**++;**

**}**

void Dijkstra**(**int s**){**

memset**(**dis**,** 0x3f**,** **sizeof(**dis**));**

dis**[**s**]** **=** 0**;**

int now **=** s**;**

**while(!**vis**[**now**])**

**{**

vis**[**now**]** **=** 1**;**

**for(**int i **=** head**[**now**];** **~**i**;** i **=** Edge**[**i**].**next**)**

**if(!**vis**[**Edge**[**i**].**to**]** **&&** dis**[**Edge**[**i**].**to**]** **>** dis**[**now**]+**Edge**[**i**].**weight**)**

dis**[**Edge**[**i**].**to**]** **=** dis**[**now**]+**Edge**[**i**].**weight**;**

mn **=** **(**1**<<**30**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)**

**{**

**if(!**vis**[**i**]** **&&** mn **>** dis**[**i**])**

**{**

now **=** i**;**

mn **=** dis**[**i**];**

**}**

**}**

**}**

**}**

signed main**(**void**)**

**{**

memset**(**head**,** **-**1**,** **sizeof(**head**));**

scanf**(**"%lld %lld %lld"**,** **&**n**,** **&**m**,** **&**s**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int u**,** v**,** w**;**

scanf**(**"%lld %lld %lld"**,** **&**u**,** **&**v**,** **&**w**);**

Build**(**u**,** v**,** w**);**

**}**

Dijkstra**(**s**);**

**for(**int i **=** 1**;** i **<=** n**;** i**++)**

printf**(**"%lld "**,** dis**[**i**]** **<** maxx **?** dis**[**i**]** **:** 2147483647**);**

**return** 0**;**

**}**

**§5 简单数学-算法1：快速幂取模（Luogu1965）**

**一句话评析：同余+二进制分解拆解幂运算，复杂度为**

#include <iostream>

#include <cstring>

#include <cstdio>

**using** **namespace** std**;**

int n**,** m**,** k**,** x**,** calc**;**

int QuickPow**(**int a**,** int b**,** int p**){**

**if(**b **==** 0**)** **return** 1**;**

int tmp **=** QuickPow**(**a**,** b**>>**1**,** p**);**

tmp **=** **(**tmp**\***tmp**)%**p**;**

**if(**b**&**1**)** tmp **=** **(**tmp**\***a**)%**p**;**

**return** tmp**%**p**;**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d %d %d"**,** **&**n**,** **&**m**,** **&**k**,** **&**x**);**

calc **=** QuickPow**(**10**,** k**,** n**);**

printf**(**"%d\n"**,** **(**x**+**m**\***calc**)%**n**);**

**return** 0**;**

**}**

**§5 简单数学-算法2：拓展欧几里得（Luogu1516）**

**一句话评析：可求解线性同余方程，不定方程或者模意义下的逆元**

#include <iostream>

#include <cstring>

#include <cstdio>

#define int long long

**using** **namespace** std**;**

int n**,** m**,** x**,** y**,** l**;**

int a**,** b**,** sx**,** sy**,** ans**;**

int exgcd**(**int a**,** int b**,** int **&**x**,** int **&**y**){**

**if(**b **==** 0**)** **{**x **=** 1**;** y **=** 0**;** **return** a**;}**

ans **=** exgcd**(**b**,** a**%**b**,** x**,** y**);**

int tmp **=** x**;**

x **=** y**;** y **=** tmp**-(**a**/**b**)\***y**;**

**return** ans**;**

**}**

signed main**(**void**)**

**{**

scanf**(**"%lld %lld %lld %lld %lld"**,** **&**x**,** **&**y**,** **&**m**,** **&**n**,** **&**l**);**

a **=** x**-**y**;**

b **=** n**-**m**;**

**if(**b **<** 0**)** a **=** **-**a**,** b **=** **-**b**;**

exgcd**(**b**,** l**,** sx**,** sy**);**

**if(**a**%**ans**)**

printf**(**"Impossible\n"**);**

**else**

printf**(**"%lld\n"**,** **(**sx**\*(**a**/**ans**)%(**l**/**ans**)+(**l**/**ans**))%(**l**/**ans**));**

**return** 0**;**

**}**

**§5 简单数学-算法3：杨辉三角的前缀和处理（Luogu2822）**

**一句话评析：简单情况下，杨辉三角套二维前缀和，加上题目的约束条件即可。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 2005

**using** **namespace** std**;**

int t**,** k**;**

int tri**[**maxn**][**maxn**];**

int sum**[**maxn**][**maxn**];**

void solve**(){**

/\*MakeTriangleTable\*/

tri**[**1**][**1**]** **=** 1**;**

**for(**int i **=** 0**;** i **<=** 2000**;** i**++)** tri**[**i**][**0**]** **=** 1**;**

**for(**int i **=** 2**;** i **<=** 2000**;** i**++)**

**for(**int j **=** 1**;** j **<=** i**;** j**++)**

tri**[**i**][**j**]** **=** **(**tri**[**i**-**1**][**j**-**1**]+**tri**[**i**-**1**][**j**])%**k**;**

/\*SolveSumTable\*/

**for(**int i **=** 2**;** i **<=** 2000**;** i**++)**

**{**

**for(**int j **=** 1**;** j **<=** i**;** j**++)**

**{**

sum**[**i**][**j**]** **=** sum**[**i**-**1**][**j**]+**sum**[**i**][**j**-**1**]-**sum**[**i**-**1**][**j**-**1**];**

**if(**tri**[**i**][**j**]** **==** 0**)** sum**[**i**][**j**]++;**

**}**

sum**[**i**][**i**+**1**]** **=** sum**[**i**][**i**];**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**t**,** **&**k**);**

solve**();**

**for(**int i **=** 1**;** i **<=** t**;** i**++)**

**{**

int n**,** m**;**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);**

**if(**m **>** n**)** m **=** n**;**

printf**(**"%d\n"**,** sum**[**n**][**m**]);**

**}**

**return** 0**;**

**}**

**§5 简单数学-算法4：枚举因子【暴力】（JZOJ5773）**

**一句话评析：设为的因子且不大于，则有因子为除以；由此可推。**

#include <iostream>

#include <cstring>

#include <cstdio>

#include <cmath>

#include <algorithm>

#define maxn 2000005

#define int long long

**using** **namespace** std**;**

int N**,** k**;**

int ans**[**maxn**];**

signed main**(**void**)**

**{**

scanf**(**"%lld"**,** **&**N**);**

**for(**int i **=** 1**;** i **<=** sqrt**(**N**);** i**++)**

**{**

**if(**N**%**i **!=** 0**)** **continue;**

**if(**i**%**2 **==** 1 **&&** i **!=** 1**)** ans**[**k**++]** **=** i**;**

**if((**N**/**i**)%**2 **==** 1 **&&** i**\***i **!=** N**)** ans**[**k**++]** **=** N**/**i**;**

**}**

printf**(**"%lld "**,** k**);**

sort**(**ans**,** ans**+**k**);**

**for(**int i **=** 0**;** i **<** k**;** i**++)** printf**(**"%lld "**,** N**/**ans**[**i**]\*(**ans**[**i**]-**1**));**

**return** 0**;**

**}**

**§6 素数判定-算法1：暴力（Luogu3383）**

**一句话评析：同理上一篇。在内就可以枚举出所有因子，从2开始枚举即可。**

#include <iostream>

#include <cstring>

#include <cstdio>

**using** **namespace** std**;**

int N**,** M**;**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**N**,** **&**M**);**

**for(**int i **=** 1**;** i **<=** M**;** i**++)**

**{**

int tmp**;** scanf**(**"%d"**,** **&**tmp**);**

**if(**tmp **==** 1 **||** tmp **==** 0**)** **{**printf**(**"No\n"**);** **continue;}**

bool isPrime **=** **true;**

**for(**int j **=** 2**;** isPrime **&&** j**\***j **<=** tmp**;** j**++)**

**if(**tmp**%**j **==** 0**)** **{**isPrime **=** **false;** **break;}**

printf**(**isPrime **?** "Yes\n" **:** "No\n"**);**

**}**

**return** 0**;**

**}**

**§6 素数判定-算法2：埃拉托斯特尼筛法（Luogu3383）**

**一句话评析：基于素数的倍数一定不是素数的思想去枚举所有的倍数来筛选素数。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 10000005

**using** **namespace** std**;**

int N**,** M**;**

bool isPrime**[**maxn**];**

void MakeTable**(**int limit**){**

memset**(**isPrime**,** **true,** **sizeof(**isPrime**));**

isPrime**[**0**]** **=** isPrime**[**1**]** **=** **false;**

**for(**int i **=** 2**;** i **<=** limit**;** i**++)**

**{**

**if(!**isPrime**[**i**])** **continue;**

**for(**int j **=** i**\***2**;** j **<=** limit**;** j **+=** i**)** isPrime**[**j**]** **=** 0**;**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**N**,** **&**M**);**

MakeTable**(**N**);**

**for(**int i **=** 1**;** i **<=** M**;** i**++)**

**{**

int tmp**;** scanf**(**"%d"**,** **&**tmp**);**

printf**(**isPrime**[**tmp**]** **?** "Yes\n" **:** "No\n"**);**

**}**

**return** 0**;**

**}**

**§6 素数判定-算法3：欧拉筛法/线性筛（Luogu3383）**

**一句话评析：优化了埃氏筛重复筛选而浪费的时间，达到线性筛选素数。**

#include <iostream>

#include <cstring>

#include <cstdio>

#define maxn 10000005

**using** **namespace** std**;**

int n**,** m**;**

int Prime**[**maxn**];**

bool isPrime**[**maxn**];**

void MakeTable**(**int limit**){**

memset**(**isPrime**,** **true,** **sizeof(**isPrime**));**

int cnt **=** 0**,** j**;**

isPrime**[**0**]** **=** isPrime**[**1**]** **=** **false;**

**for(**int i **=** 2**;** i **<=** limit**;** i**++)**

**{**

**if(**isPrime**[**i**])** Prime**[**cnt**++]** **=** i**;**

**for(**j **=** 0**;** j **<** cnt **&&** i**\***Prime**[**j**]** **<=** limit**;** j**++)**

**{**

isPrime**[**i**\***Prime**[**j**]]** **=** **false;**

**if(**i**%**Prime**[**j**]** **==** 0**)** **break;**

**}**

**}**

**}**

int main**(**void**)**

**{**

scanf**(**"%d %d"**,** **&**n**,** **&**m**);** MakeTable**(**n**);**

**for(**int i **=** 1**;** i **<=** m**;** i**++)**

**{**

int tmp**;** scanf**(**"%d"**,** **&**tmp**);**

printf**(**isPrime**[**tmp**]** **?** "Yes\n" **:** "No\n"**);**

**}**

**return** 0**;**

**}**