ACM 模板

ACM Template

rogeryoungh

2021 年 1 月 6 日 · LaTeX

目录

第 1 章 上号

1.1 头文件

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define _fora(i,a,n) for(ll i=(a);i<=(n);i++)
#define _forz(i,a,n) for(ll i=(a);i>=(n);i--)
#define _forb(i,a) for(ll i=(a);i>0;i-=i&(-i))
#define _dbg(x) cout<<"[Log] "<<#x<<" = "<<x<<endl;
#define _in(i,min,max) ( ((i)-(min)) | ((max)-(i)) )
#define _fore(i,a) for(int i=head[(a)];i;i=edge[i].nxt)
inline ll rr() {
    ll s=0,w=1;char c=getchar();
    while(_in(c,'0','9')<0) { if(c=='-') w*=-1; c=getchar(); }
    while(_in(c,'0','9')>=0) { s=s*10+c-'0'; c=getchar(); }
    return s*w;
}
inline void pr(ll x) { if(x>=10) pr(x/10); putchar(x%10+'0'); }
int main() {
    printf("\n");
    return 0;
}
```

1.2 预编译

头文件引入方式改为如下，可以把头文件放入 `lab.h`，然后使用 `g++ lab.h` 预编译。

实际编译使用 `g++ lab.cpp -D YCYLOCAL` 添加条件编译参数。

```cpp
#ifdef YCYLOCAL
#include "lab.h"
#else  //#include <bits/stdc++.h>
#include <list>      #include <stack>      #include <iostream>
#include <cmath>     #include <cstdio>     #include <algorithm>
#include <queue>     #include <cstring>    #include <functional>
#endif
```

1.3 进制转换

```
void pr_x(ll n,int x) {
    char c = n%x; c += x>9?'A'-10:'0';
    if(n>=x) pr(n/x,x); putchar(c);
}
```

1.4 常见技巧

向上取整 p/q 为 (p-1)/q+1 。

1.5 快速幂

```
ll power(ll a,ll b,ll p) {
    ll rst = 1%p;
    for(;b>0;b>>=1,a=a*a%p)
        if(b&1) rst=a*rst%p;
    return rst;
}
```

1.6 矩阵快速幂

```
const int mod=1e9+7;
ll a[105][105],b,n;
ll ans[105][105]={0};
inline void jzcf1(){
    ll c[105][105]={0};//新矩阵放临时结果
    _fora(k,1,n) _fora(i,1,n) _fora(j,1,n)
        c[i][j]=(c[i][j]+ans[i][k]*a[k][j])%mod;
    _fora(i,1,n) _fora(j,1,n)
        ans[i][j]=c[i][j];
}
inline void jzcf2(){
    ll c[105][105]={0};
    _fora(k,1,n) _fora(i,1,n) _fora(j,1,n)
        c[i][j]=(c[i][j]+ans[i][k]*a[k][j])%mod;
    _fora(i,1,n) _fora(j,1,n)
        a[i][j]=c[i][j];
}
int main(){
    int n; //读入
    _fora(i,1,n) ans[i][i]=1;
```

```
    while(b){
        if(b&1) jzcf1();//矩阵乘法
        jzcf2(); b>>=1;
    }
}
```

1.7 快速排序

```
ll nn[10010];
void q_sort(int l,int r) {
    int i=l,j=r,x = nn[i];
    while(i<j) {
        while(nn[j]>x&&i<j) j--;
        if(i<j) nn[i++] = nn[j];
        while(nn[i]<x&&i<j) i++;
        if(i<j) nn[j--] = nn[i];
    } nn[i] = x;
    if(l<r) { q_sort(l,i-1); q_sort(i+1,r); }
}
```

1.8 第 k 大数

```
ll nn[5000010],k,n;
ll q_sort(ll l,ll r) {
    ll i=l,j=r,x=nn[(l+r)/2];
    while(i<=j) {
        while(nn[j]>x) j--;
        while(nn[i]<x) i++;
        if(i<=j) swap(nn[i++],nn[j--]);
    } //l<=j<=i<=r
    if(k<=j) return q_sort(l,j);
    else if(k>=i) return q_sort(i,r);
    else return nn[k+1];
}
```

1.9 链表

```
struct Node { int val; Node *prev,*next; };
struct List {
    Node *head,*tail; int len;
```

```cpp
    List() {
        head = new Node(); tail = new Node();
        head->next = tail; tail->prev - head;
        len = 0;
    } // 在节点后 p 后插入值 v
    void insert(Node *p,int v) {
        Node *q = new Node(); q->val = v;
        p->next->prev = q; q->next = p->next;
        p->next = q; q->prev = p;
        len++;
    } // 删除节点 p
    void erase(Node *p) {
        p->prev->next = p->next; p->next->prev = p->prev;
        delete p; len--;
    } // 清空链表
    ~List() {
        while(head != tail) {
            head = head->next;
            delete head->prev;
        }
        delete tail; len = 0;
    }
};
for(Node *p=l->head->next;p!=l->tail;p=p->next)
    //正序遍历
for(Node *p=l->tail->prev;p!=l->head;p=p->prev)
    //逆序遍历
```

第 2 章 数论

2.1 GCD 和 LCM

```
ll gcd(ll a,ll b) { return a?gcd(b%a,a):b; }
ll lcm(ll a,ll b) { return a/gcd(b%a,a)*b; }
ll gcd(ll a,ll b) { while(b) { ll t=a%b; a=b; b=t; } return a; }
```

2.2 EXGCD

对于方程 $ax+by=c$，令 $g=\gcd(a,b)$，若 $g\nmid c$ 则无解。

因此可通过 $exgcd$ 先求出 $ax+by=g$ 的整数解，再转换回原方程的解。

```
void exgcd(ll a,ll b,ll& x,ll& y) {
    if(!b) { y=0; x=1; return; }
    exgcd(b,a%b,y,x); y-=a/b*x;
}
```

2.3 Eratosthenes 筛

```
bool notn[100000001];
int prime[20000001], cnt;
void init(int n) {
    _fora(i,2,n) { if(!notn[i]) {
        prime[++cnt] = i;
        int tn = n/i;
        _fora(j,i,tn) notn[i*j] = true;
    } } return;
}
```

2.4 Eular 筛

```
bool notn[100000001];
int prime[20000001],cnt;
void init(int n){
    _fora(i,2,n) {
        if(!notn[i]) prime[++cnt] = i;
        int t = n/i;
        _fora(j,1,cnt) {
```

```
            if(prime[j]>t) break;
            notn[i*prime[j]] = true;
            if(i%prime[j]==0) break;
        }
    } return;
}
```

2.5 素性测试

2.5.1 试除法

```
bool isprime(ll n) {
    if(n<3) return n==2;
    if(n&1==0) return false;
    ll sn = (ll)sqrt(n*1.0);
    for(ll i=3;i<=sn;i+=2)
        if(n%i==0) return false;
    return true;
}
```

2.5.2 Miller Rabbin

如果 $n \leqslant 2^{32}$，那么 ppp 取 $2, 7, 61$；如果 ppp 选择 $2, 3, 7, 61, 24251$，那么 10^{16} 内只有唯一的例外。如果莫名 WA 了，就多取点素数吧。

```
bool miller_rabbin(ll n) {
    if (n<3) return n==2;
    if(n&1==0) return false;
    int a=n-1,b=0,j;
    while (1-a&1) a/=2,++b;
    int ppp[10] = {2,7,61};
    _fora(i,0,2) {
        int x = ppp[i];
        if(n==x) return true;
        ll v = power(x,a,n);
        if(v==1||v==n-1) continue;
        for(j=0;j<b;++j) {
            v = v*v%n; if(v==n-1) break;
        }
        if(j>=b) return false;
    } return true;
}
```

第 3 章 图论

3.1 链式前项星

```
const int MN = 10005; int head[MN];
struct Edge {int too,nxt,len;} edge[MN*2];
void add(int frm,int too,int len) {
    static int cnt = 0;
    edge[++cnt] = { too,head[frm],len };
    head[frm] = cnt;
}
void dfs(int x,int fa) {
    _fore(i,x) if(edge[i].too!=fa)
        dfs(edge[i].too,x);
}
```

3.2 Dijkstra

```
int dis[MN];
struct Dis {
    int dis,pos;
    bool operator <(const Dis& x) const
        { return x.dis<dis; }
};
void dijkstra(int ss) {
    memset(dis,0x3f,sizeof(dis));
    dis[ss] = 0;
    priority_queue<Dis> pq; pq.push({0,ss});
    while(!pq.empty()) {
        Dis td = pq.top(); pq.pop();
        int d=td.dis, x=td.pos;
        if(d!=dis[x]) continue;
        _fore(i,x) {
            int y=edge[i].too, z=dis[x]+edge[i].len;
            if(dis[y]>z) dis[y]=z, pq.push({dis[y],y});
        }
    }
}
```

3.3 Bellman-Ford

```
void bellman_ford(int ss) {
    memset(dis,0x3f,sizeof(dis)); dis[ss]=0;
    _fora(iia,1,n-1) { int flag=1;
        _fora(i,1,n) { int size=ee[i].size();
            _fora(j,0,size-1) {
                int v=ee[i][j].nxt, t=dis[i]+ee[i][j].len;
                if(dis[ee[i][j].nxt]>t) {
                    dis[ee[i][j].nxt] = t;
                    flag = 0;
                }
            }
        } if(flag) return;
    }
}
```

3.4 Floyd

起始条件 $f(i,j) = edge(i,j)$，$f(i,i) = 0$。

```
inline void floyd() {
    _fora(k,1,n) { _fora(i,1,n) {
        if(i==k||f[i][k]==0x3f3f3f3f) continue;
        _fora(j,1,n) f[i][j] = min(f[i][j],f[i][k]+f[k][j]);
    } }
}
```

3.5 最近公共祖先 (LCA)

如果数据小，可以不用求 $\log_2$，直接莽 20。

```
int fa[MN][30],lgb[MN],depth[MN];
void lca_dfs(int now,int fat) {
    _fora(i,1,n) lgb[i]=lgb[i>>1]+1; lgb[1]=0;
    fa[now][0] = fat;
    depth[now] = depth[fat]+1;
    _fora(i,1,lgb[depth[now]])
        fa[now][i] = fa[fa[now][i-1]][i-1];
    _fore(i,now) if(edge[i].too!=fat)
        lca_dfs(edge[i].too,now);
}
```

```
int lca(int x,int y) {
    if(depth[x]<depth[y]) swap(x,y);
    while(depth[x]>depth[y])
        x = fa[x][lgb[depth[x]-depth[y]]];
    if(x==y) return x;
    _forz(k,lgb[depth[x]]-1,0)
        if(fa[x][k]!=fa[y][k])
            { x=fa[x][k]; y=fa[y][k]; }
    return fa[x][0];
}
```

第 4 章 动态规划

4.1 背包

4.1.1 01 背包

给定体积为 v_i，价值 w_i 的 N 个物品，背包容积为 M，每个物品只能取 1 个，求最大价值。

```
_fora(i,1,n) _forz(j,m,v[i])
    dp[j]=max(dp[j],dp[j-v[i]]+w[i]);
_fora(j,1,m) ans = max(ans,dp[j]);
```

4.1.2 完全背包

给定体积为 v_i，价值 w_i 的 n 个物品，背包容积为 v，每个物品任意取，求最大价值。

```
_fora(i,1,n) _fora(j,v[i],m)
    dp[j]=max(dp[j],dp[j-v[i]]+w[i]);
_fora(j,1,m) ans = max(ans,dp[j]);
```

4.1.3 多重背包

给定体积为 v_i，价值 w_i 的 N 个物品，背包容积为 M，每个物品有 c_i 个，求最大价值。

如各种背包组合（如洛谷 P1833 樱花），通常把完全背包转为 99999 个（适当调节）多重背包，再按 01 背包来。

```
int tm=1,vv[],ww[];
_fora(i,1,n) {
    int tc = c[i];
    for(int b=1;b<p;b<<=1,tc-=b,++tm) {
        vv[tm] = v[i]*b;
        ww[tm] = w[i]*b;
    }
    vv[tm] = v[i]*tc;
    ww[tm] = w[i]*tc;
    ++tm;
}
_fora(i,1,n) _forz(j,m,vv[i])
    dp[j]=max(dp[j],dp[j-vv[i]]+ww[i]);
_fora(j,1,m) ans = max(ans,dp[j]);
```

4.2 最长公共上升序列

给出 $1, 2, \ldots, n$ 的两个排列 a 和 b ，求它们的最长公共子序列。

```
int f[MN],ma[MN],b[MN];
int n,len=0; memset(f,0x3f,sizeof(f)); f[0]=0;
_fora(i,1,n) { ma[rr()]=i; } _fora(i,1,n) { b[i]=rr(); }
_fora(i,1,n) {
    int l=0,r=len;
    if(ma[b[i]]>f[len]) f[++len]=ma[b[i]];
    else { while(l<r) {
        int mid=(l+r)/2;
        if(f[mid]>ma[b[i]])r=mid;
        else l=mid+1;
    } } f[l]=min(ma[b[i]],f[l]);
}
```

4.3 数字计数

试计算在区间 1 到 n 的所有整数中，数字 $x(0 \leqslant x \leqslant 9)$ 共出现了多少次？

```
ll addup(ll n, ll x,ll k,ll l){ //k=1,l=0
    ll  a=0; if(n>9) a=addup(n/10,x,k*10,l+1);
    n%=10; sum += ((n>x)+n*a)*k+n*l*k/10-(!x)*k;
    return a+(n==x);
}
```