杂项

二维差分前缀和

```
while(m--){
    int x1,x2,y1,y2;
    cin>>x1>>y1>>x2>>y2;
    a[x1][y1]++;
    a[x1][y2+1]--;
    a[x2+1][y1]--;
    a[x2+1][y2+1]++;
}
for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
        a[i][j]+=(a[i][j-1]+a[i-1][j]-a[i-1][j-1]);
     }
}
```

分组背包

```
for(  组数  ){        //分组背包
    for(  背包大小  ){
          for(  该组物品  )
    }
}
void solve(){
    int n,V;
    cin>>n>>V;
    map<int,vector<pii>>mp;
    while(n--){
        int w,v,p;
        cin>>w>>v>>p;
        mp[p].emplace_back(w,v);
    }
    int ans=0;
    vector<int>dp(V+1,0);
    for(auto _:mp){
        for(int i=V;i>=1;i--){
            for(auto j:_.second){
                int w=j.first,v=j.second;
                if(w>i)continue;
                dp[i]=max(dp[i],dp[i-w]+v);
                ans=max(ans,dp[i]);
            }
        }
    }
    cout<<ans<<endl;
}
```

压位高精

```
const int mod=1e9;
struct BigInt{          // 压位高精加
    int cur;
    int *a;
    void init(){
        a=new long long[20];
        for(int i=0;i<20;i++)a[i]=0;
        cur=0;
    }
    void put(){
        printf("%lld",a[cur]);
        for(int i=cur-1;i>=0;i--) printf("%09lld",a[i]);
    }
    void add(int k){
        a[0]+=k;
        int i=0;
        while(a[i]>=mod)a[i+1]+=a[i]/mod,a[i++]%=mod;
        while(a[cur+1])cur++;
    }
    void Add(const BigInt& o){
        int r=max(cur,o.cur);
        for(int i=0;i<=r;i++){
            a[i]+=o.a[i];
            if(a[i]>=mod)a[i+1]+=a[i]/mod,a[i]%=mod;
        }
        cur=min(r+3,19ll);while(cur && a[cur]==0) cur--;
    }
};
```

手写 Bitset

```
const int V=470;       //   M/64
const unsigned long long MOD=18446744073709551615ull;
struct Bitset{
    unsigned long long bit[V+5]{};
    Bitset(){memset(bit,0,sizeof(bit));}
    void clear(){memset(bit,0,sizeof(bit));}
    void ins(int x){
        bit[x>>6]|=(1ull<<(x&63));
    }
    void del(int x){
        bit[x>>6]&=(MOD^(1ull<<(x&63)));
    }
    Bitset operator|(const Bitset &b)const{
        Bitset ret;
        for(int i=0;i<V;i++)ret.bit[i]=bit[i]|b.bit[i];
        return ret;
    }
    bool get(int x){
        return (bit[x>>6]>>(x&63))&1;
    }
    int mex(){
```

```
        for(int i=0;i<V;i++){
            if(bit[i]^MOD){
                for(int j=0;j<64;j++){
                    if(!((bit[i]>>j)&1)) return i<<6|j;
                }
            }
        }
    }
    int count(){
        int ret=0;
        for(int i=0;i<V;i++)ret+=__builtin_popcountll(bit[i]);
        return ret;
    }
};
```

自定义 set 比较函数

```
class cmp{
    bool operator()(const int&a,const int&b)const{
        return a<b;
    }
};
set<int,cmp>s;
```

pbds

```
#include<bits/extc++.h>
using namespace __gnu_pbds;
__gnu_pbds::priority_queue<int,less<>,__gnu_pbds::pairing_heap_tag>pq;   // 大根堆
| greater<> 为小根堆
// 大根堆遍历顺序为从小到大，小根堆遍历顺序为从大到小
void join(priority_queue &other){}      // 把 other 合并到 *this，然后 other 被清空
```

负数快读

```
inline int read(){
    int x=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9'){
        if(ch=='-')f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9'){
        x=(x<<1)+(x<<3)+(ch^48);
        ch=getchar();
    }
    return x*f;
}
inline void write(int x){
    if(x<0)putchar('-'),x=-x;
    if(x>9)write(x/10);
    putchar(x%10+'0');
}
```

正整数快读

```
inline int read(){
    int x=0;
    char ch=getchar();
    while(ch<'0'||ch>'9'){
        ch=getchar();
    }
    while(ch>='0'&&ch<='9'){
        x=(x<<1)+(x<<3)+(ch^48);
        ch=getchar();
    }
    return x;
}
inline void write(int x){
    if(x>9)write(x/10);
    putchar(x%10+'0');
}
```

手写Map

```
struct Hash_Map{
    static const int MOD=1999997;
    int Hash[MOD],V[MOD],stk[MOD],top;
    //Hash_table() {memset(Hash,-1,sizeof(Hash));}
    inline void Insert(int val,int mi){
        int h=val%MOD;
        while(Hash[h]&&Hash[h]!=val) h++;
        Hash[h]=val;V[h]=mi;
        stk[++top]=h;
        return;
    }
    inline int find(int val){
        int h=val%MOD;
        while(Hash[h]&&Hash[h]!=val) h++;
        return Hash[h]==val?V[h]:-1;
    }
}H;
```

自定义umap（防卡Hash）

```
struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        x+=0x9e3779b97f4a7c15;
        x=(x^(x>>30))*0xbf58476d1ce4e5b9;
        x=(x^(x>>27))*0x94d049bb133111eb;
        return x^(x>>31);
    }
    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM =
chrono::steady_clock::now().time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
```

```
    }
};

unordered_map<int, int, custom_hash> mp;
```

匿名函数

```
function<void(int,int)>dfs=[&](int x,int y){
    int a,b;
    dfs(a,b);
    return;
};
```

StringStream 分隔字符

```
for(int i=1;i<=m;i++){
    string s;
    int x;
    getline(cin,s);
    stringstream ss;
    ss<<s;
    while(ss>>x){
        v[i].push_back(x);
    }
}
```

Vector嵌套Array

```
vector<array<int,5>>dp(n+1,array<int,5>{});
vector<int>().swap(a);          // vector 空间释放
```

指令集优化

```
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
```

开栈

```
int main() {
    int size(256<<20); //256M
    __asm__ ( "movq %0, %%rsp\n"::"r"((char*)malloc(size)+size));
    // YOUR CODE
    //...
    exit(0);
}
```

Windows 对拍

```
// run.bat
@echo off
:start
gen.exe
std.exe<data.out
my.exe<data.out
fc std.out my.out
if not errorlevel 1 goto start
pause
go start

// gen.cpp  ->  data.out
freopen("data.out","w",stdout);

// my.cpp  ->  my.out
freopen("my.out","w",stdout);

// std.cpp  ->  std.out
freopen("my.out","w",stdout);
```

随机数

```
mt19937_64 Rnd(random_device{}());
mt19937 Rnd(chrono::steady_clock::now().time_since_epoch().count());
uniform_int_distribution<long long>dist1(0,10000);
uniform_real_distribution<double>dist2(-10,10);
Cout<<Rnd()<<endl;      //完全随机
cout<<dist1(Rnd)<<endl;          //限定范围
cout<<dist2(Rnd)<<endl;
```

随机质数

979345007，986854057502126921

935359631，949054338673679153

check

1. 多测清空，调用初始化，清空时数组大小，没读入完就 break
2. 数组是否开大，数组是否越界，set, map 调用时是否越界
3. 取模是否漏取，取模取到正数
4. 复制过的代码对应位置注意正确修改
5. 几何: 共线, sqrt(-0.0), nan / inf, 重点, 除零 det/dot, 旋转方向, 求得的是否是所求
6. int/LL 溢出, INF/-1 大小，浮点数 eps 和 = 0, __builtin_popcountll
7. 模数
8. n=0/1 特殊情况
9. 答案初值是否设对

OIES

$$1^2+2^2+\ldots+n^2=\frac{n(n+1)(2n+1)}{6} \qquad =\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n$$

$$1^3+2^3+\ldots+n^3=(\frac{n(n+1)}{2})^2 \qquad =\frac{1}{4}n^4+\frac{1}{2}n^3+\frac{1}{4}n^2$$

$$1^4+2^4+\ldots+n^4=\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} \qquad =\frac{1}{5}n^5+\frac{1}{2}n^4+\frac{1}{3}n^3-\frac{1}{30}n$$

$$1^5+2^5+\ldots+n^5=\frac{n^2(n+1)^2(2n^2+2n-1)}{12} \qquad =\frac{1}{6}n^6+\frac{1}{2}n^5+\frac{5}{12}n^4-\frac{1}{12}n^2$$