Contents

1 Basic

1.1 Default code

```cpp
#include<bits/stdc++.h>
#include<chrono> // for timing
#pragma GCC optimize("O3,unroll-loops")
#pragma target optimize("avx2,bmi,bmi2,lzcnt,popcnt")
#define IO ios_base::sync_with_stdio(0);cin.tie(0);cout
    .tie(0);
#define pii pair<int,int>
#define ft first
#define sd second
#define int long long
#define double long double
#define PI acos(-1)
#define SZ(x) (int)x.size()
#define all(v) (v).begin(), (v).end()
#define  _for(i,a,b) for(int i=(a);i<(b);++i)
using namespace std;
template<typename T>
ostream& operator<<(ostream& os,const vector<T>& vn){
  for(int i=0;i<vn.size();++i)os<<vn[i]<<" ";
  return os;
}
template<typename T>
ostream& operator<<(ostream& os,const set<T>& vn){
  for(typename set<T>::iterator it=vn.begin();it!=vn.
      end();++it)os<<*it<<" ";
  return os;
}
mt19937 mt(hash<string>()("Mashu_AC_Please")); //mt();
// mt19937 mt(chrono::steady_clock::now().
    time_since_epoch().count());
// g++ a.cpp -Wall -Wshadow -fsanitize=undefined -o a.
    exe
// ./a.exe
const int MXN=2e5+5;
const int INF=INT_MAX;
void sol() {}
signed main() {
    // auto start=chrono::high_resolution_clock::now();
    // #ifdef LOCAL
    // freopen("input.txt","r",stdin);
    // freopen("output.txt","w",stdout);
    // #endif
    IO
    int t=1;
    cin>>t;
    while(t--) {sol();}
    // auto stop = chrono::high_resolution_clock::now()
        ;
    // auto duration = chrono::duration_cast<chrono::
        milliseconds>(stop - start);
    // cerr<<"Time:"<<duration.count()<<" ms\n";
}
```

1.2 Misc

```cpp
iota(vec.begin(),vec.end(),1);// 產生1~size的整數列
stoi(s.begin(),s.end(),k);// 法1,字串轉成k進位int
string s;cin>>s;
int x=stoi(s,0,2); // 法2,2可以改其他進位
__builtin_popcountll // 二進位有幾個1
__builtin_clzll // 左起第一個1前0的個數
__builtin_parityll // 1的個數的奇偶性
__builtin_mul_overflow(a,b,&res) // a*b是否溢位

// double 轉整數 請加 int b=round(a)
// 或是 int b =floor(a+0.5) (floor向下取整)
```

1.3 Fast read & write

```cpp
inline int read() {
    char c = getchar(); int x = 0, f = 1;
    while(c < '0' || c > '9') {if(c == '-') f = -1; c =
        getchar();}
```

```
    while(c >= '0' && c <= '9') x = x * 10 + c - '0', c
         = 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');
}
```

1.4 Sort cmp

```
struct cmp{inline bool operator()(const int a,const int
     b){return a<b;}};//common use
auto cmp=[](vector<int> a, vector<int> b) {return a[1]<
    b[1];};//for set use
set<vector<int>, decltype(cmp)> prepare, done;
```

1.5 Custom unordered_map

```
struct Type{
    int x;
    string y;
    bool operator==(const Type &other) const {
        return (x == other.x && y == other.y);
    }
};
struct hashs{
    size_t operator()(const Type &o) const {
        return ((hash<int>()(o.x)^(hash<string>()(o.y)
            <<1))>>1);
    }
};
//unordered_map<Type,int,hashs> map;
```

1.6 __int128 read

```
//  __int128_t p;
//  lll n=qr(p);
#define lll __int128
template<class type_name> inline type_name qr(type_name
     sample)
{
    type_name ret=0,sgn=1;
    char cur=getchar();
    while(!isdigit(cur))
        sgn=(cur=='-'?-1:1),cur=getchar();
    while(isdigit(cur))
        ret=(ret<<1)+(ret<<3)+cur-'0',cur=getchar();
    return sgn==-1?-ret:ret;
}
```

1.7 字典序 a 嚴格小於 b

```
template<class T> //字典序a嚴格小於b
bool lexicographicallySmaller(const vector<T> &a,const
    vector<T> &b){
    int n=a.size();
    int m=b.size();
    int i;
    for(int i=0;i<n && i<m;++i){
        if(a[i]<b[i])return true;
        else if(b[i]<a[i])return false;
    }
    return (i==n && i<m);
}
```

1.8 Radom

```
mt19937 gen(0x5EED);
int randint(int lb, int ub)
{ return uniform_int_distribution<int>(lb, ub)(gen); }
```

2 對拍

2.1 run.bat

```
@echo off
g++ ac.cpp -o ac.exe
g++ wa.cpp -o wa.exe
g++ gen1.cpp -o gen.exe

:loop
   echo %%x
   gen.exe > input
   ac.exe < input > ac
   wa.exe < input > wa
   fc ac wa
if not errorlevel 1 goto loop
```

2.2 run.sh

```
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

3 Flow & Matching

3.1 Dicnic

```
// flow.init(n,s,t):有n個點(0~n-1)，起點s終點t
// flow.add_edge(u,v,f):建一條邊，從u點到v點流量為f
// flow.solve():回傳網路最大流答案
//時間複雜度: O(V^2*E)
struct Dinic{
    struct Edge{ int v,f,re; };
    int n,s,t,level[MXN];
    vector<Edge> E[MXN];
    void init(int _n, int _s, int _t){
        n = _n; s = _s; t = _t;
        for (int i=0; i<n; i++) E[i].clear();
    }
    void add_edge(int u, int v, int f){
        E[u].push_back({v,f,(int)(E[v]).size()});
        E[v].push_back({u,0,(int)(E[u]).size()-1});
    }
    bool BFS(){
        for (int i=0; i<n; i++) level[i] = -1;
        queue<int> que;
        que.push(s);
        level[s] = 0;
        while (!que.empty()){
            int u = que.front(); que.pop();
            for (auto it : E[u]){
            if (it.f > 0 && level[it.v] == -1){
                level[it.v] = level[u]+1;
                que.push(it.v);
        } } }
        return level[t] != -1;
    }
    int DFS(int u, int nf){
        if (u == t) return nf;
        int res = 0;
        for (auto &it : E[u]){
            if (it.f > 0 && level[it.v] == level[u]+1){
            int tf = DFS(it.v, min(nf,it.f));
            res += tf; nf -= tf; it.f -= tf;
            E[it.v][it.re].f += tf;
            if (nf == 0) return res;
        } }
        if (!res) level[u] = -1;
        return res;
    }
    int solve(int res=0){
    while ( BFS() )
        res += DFS(s,2147483647);
    return res;
```

```
} }flow;
```

3.2 ZKW FLow

```
//最大流量上的最小花費
//最大流量優先，相同才是找最小花費，複雜度O(V^2*E^2)
// flow.init(n,s,t):有n個點(0~n-1)，起點s終點t
// flow.add_edge(u,v,f,c):建一條邊，從u點到v點流量為f，每一單位流量的花費為c
// flow.solve():回傳一個pair(maxFlow,minCost)
// 限制: 圖不能有負環
// 網路最大流的add_edge(u,v,f)可以無痛轉成最大流量上的最小花費add_edge(u,v,1,f)即建立一條從u到v的邊流量為1，單位流量花費為f
#define ll long long
struct zkwflow{
    static const int maxN=20000;
    struct Edge{ int v,f,re; ll w;};
    int n,s,t,ptr[maxN]; bool vis[maxN]; ll dis[maxN];
    vector<Edge> E[maxN];
    void init(int _n,int _s,int _t){
        n=_n,s=_s,t=_t;
        for(int i=0;i<n;i++) E[i].clear();
    }
    void add_edge(int u,int v,int f,ll w){
        E[u].push_back({v,f,(int)E[v].size(),w});
        E[v].push_back({u,0,(int)E[u].size()-1,-w});
    }
    bool SPFA() {
        fill_n(dis, n, LLONG_MAX);
        fill_n(vis, n, false);
        queue<int> q;
        q.push(s); dis[s]=0;
        while(!q.empty()) {
            int u = q.front(); q.pop();
            vis[u] = false;
            for(auto &it: E[u]){
                if(it.f>0 && dis[it.v]>dis[u]+it.w){
                    dis[it.v] = dis[u]+it.w;
                    if(!vis[it.v]) {vis[it.v] = true; q
                        .push(it.v);}
                }
            }
        }
        if(dis[t]==LLONG_MAX) return false;
        // 不管流量是多少，花費不能是正數時加上這行 (最小花費可行流)
        // if(dis[t] >= 0) return false;
        return true;
    }
    int DFS(int u, int nf) {
        if(u==t) return nf;
        int res = 0; vis[u] = true;
        for(int &i=ptr[u] ; i<(int)E[u].size() ; i++) {
            auto &it = E[u][i];
            if(it.f>0 && dis[it.v]==dis[u]+it.w && !vis
                [it.v]) {
                int tf = DFS(it.v, min(nf, it.f));
                res += tf;
                nf-=tf;
                it.f-=tf;
                E[it.v][it.re].f += tf;
                if(nf==0) { vis[u]=false; break; }
            }
        }
        return res;
    }
    pair<int,ll> solve(){
        int flow = 0; ll cost = 0;
        while (SPFA()){
            fill_n(ptr, n, 0);
            int f = DFS(s, INT_MAX);
            flow += f;
            cost += dis[t]*f;
        }
        return {flow, cost};
    } // reset: do nothing
} flow;
```

3.3 Hungarian

```
//匈牙利演算法-二分圖最大匹配
//記得每次使用需清空vis數組
//O(nm)
//其中Map為鄰接表(Map[u][v]為u和v是否有連接) S為紀錄這個點與誰匹配(S[i]為答案i和誰匹配)
const int M=505, N=505;
bool Map[M][N] = {0};
int S[N];
bool vis[N];
bool dfs(int u){
    for(int i=0;i<N;i++){
        if(Map[u][i]&&!vis[i]){ //有連通且未拜訪
            vis[i]=1; //紀錄是否走過
            if(S[i]==-1||dfs(S[i])){  //紀錄匹配
                S[i]=u;
                return true;  //反轉匹配邊以及未匹配邊的狀態
            }
        }
    }
    return false;
}
//此二分圖為左邊M個點右邊N個點, 跑匈牙利只要跑1~M就可以了, (S[右邊的點] -> 左邊的點)
memset(S,-1,sizeof(S));
int ans = 0;
for(int i=0;i<M;i++){
    memset(vis,0,sizeof(vis));
    if(dfs(i)) ans++;
    //跑匈牙利
}
cout<<ans<<"\n";
for(int i=0 ; i<N ;i++) {
    if(S[i]!=-1) cout<<"pair: "<<S[i]<<" "<<i<<"\n";
}
```

3.4 KM

```
//二分圖最大權完美匹配
//二分圖左邊的點都要匹配到右邊的點，且每條邊都有權重，求權重最大值，複雜度O(V^3)
// graph.init(n):二分圖左右各n個點
// graph.add_edge(u,v,w):建一條邊，從u點到v點權重為w
// graph.solve():回傳最大權重
struct KM{ // max weight, for min negate the weights
    int n, mx[MXN], my[MXN], pa[MXN];
    ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
    bool vx[MXN], vy[MXN];
    void init(int _n) { // 1-based, N個節點
        n = _n;
        for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0)
            ;
    }
    void add_edge(int x, int y, ll w) {g[x][y] = w;} //
        左邊的集合節點x連邊右邊集合節點y權重為w
    void augment(int y) {
        for(int x, z; y; y = z)
          x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
    }
    void bfs(int st) {
        for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i
            ]=0;
        queue<int> q; q.push(st);
        for(;;) {
            while(q.size()) {
                int x=q.front(); q.pop(); vx[x]=1;
                for(int y=1; y<=n; ++y) if(!vy[y]){
                    ll t = lx[x]+ly[y]-g[x][y];
                    if(t==0){
                        pa[y]=x;
                        if(!my[y]){augment(y);return;}
                        vy[y]=1, q.push(my[y]);
                    }else if(sy[y]>t) pa[y]=x,sy[y]=t;
                }
            }
            ll cut = INF;
```

```
            for(int y=1; y<=n; ++y)
                if(!vy[y]&&cut>sy[y]) cut=sy[y];
            for(int j=1; j<=n; ++j){
                if(vx[j]) lx[j] -= cut;
                if(vy[j]) ly[j] += cut;
                else sy[j] -= cut;
            }
            for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y
                ]==0){
                if(!my[y]){augment(y);return;}
                vy[y]=1, q.push(my[y]);
            }
        }
    }
    ll solve(){ // 回傳值為完美匹配下的最大總權重
        fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
        fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
        for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
             // 1-base
          lx[x] = max(lx[x], g[x][y]);
        for(int x=1; x<=n; ++x) bfs(x);
        ll ans = 0;
        for(int y=1; y<=n; ++y) ans += g[my[y]][y];
        return ans;
    }
} graph;
```

4 Math

4.1 Formulas

```
//五次方冪次和
a(n) = n^2*(n+1)^2*(2*n^2+2*n-1)/12.
```

4.2 Quick Pow

```
// a^b
const int MOD = 1e9+7;
int qpow(int n, int k,int p) {
    int ret = 1;
    for(;k; k >>= 1, n = n * n % p) if(k & 1) ret = ret
         * n % p;
    return ret;
}
// a^(b^c) = a^(q*(p-1)+r) = a^r so let b^c mod p-1
bc =qpow(b,c,p-1);
ans=qpow(a,bc,p);
```

4.3 Mat quick Pow

```
struct mat{
    long long a[200][200],r,c; // resize
    mat(int _r,int _c){r=_r;c=_c;memset(a,0,sizeof(a))
        ;}
    void build(){for(int i=0;i<r;++i)a[i][i]=1;}
};
mat operator * (mat &x,mat &y){
    mat z(x.r,y.c);
    for(int i=0;i<x.r;++i)for(int j=0;j<x.c;++j)for(int
         k=0;k<y.c;++k)
        z.a[i][j]=(z.a[i][j]+x.a[i][k]*y.a[k][j]%MOD)%
            MOD;
    return z;
}
mat qpow(mat a,int k){
    mat r(a.r,a.r);r.build();while(k){if(k&1)r=r*a;a=a*
        a;k>>=1;}return r;
}
```

4.4 Primes Table

```
int np[MXN];
vector<int> vec;
void sol(){
    np[0]=np[1]=1;
    for(int i=2;i<MXN;++i){
        if(!np[i]){
            for(int j=i;j<MXN;j+=i){
                np[j]=1;
            }
            vec.push_back(i);
        }
    }
}
```

4.5 Factor Table

```
int arr[MXN];
void init(){
    for(int i=1;i<MXN;++i) for(int j=i;j<MXN;j+=i) arr[
        j]++;
}
```

4.6 Catalan Number

```
// O(N)，要記得開long long 跟設定 MOD
cat[0]=1; cat[1]=1;
for(ll i=1 ; i<N ; i++) {
    cat[i+1] = cat[i]*(i*4+2)%MOD*qpow(i+2, MOD-2)%MOD;
}
```

4.7 Miller Rabin

```
#define LL long long
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 : 2, 325, 9375,
    28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
LL magic[]={}; // **<- here**
__int128 mypow(__int128 a, __int128 b, __int128 p) {
    if(b==0) return 1;
    else if(b==1) return a%p;
    else if(b%2==0) {
        __int128 t=mypow(a, b/2, p);
        t = t*t%p;
        return t;
    } else if(b%2==1){
        __int128 t=mypow(a, b/2, p);
        t = t*t%p;
        t = t*a%p;
        return t;
    }
    return 0;
}
__int128 mul(__int128 a, __int128 b, __int128 p) {
    return (a*b)%p;}
__int128 add(__int128 a, __int128 b, __int128 p) {
    return (a+b)%p;}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size);  // **<-here**
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  LL u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
```

```
  }
  return 1;
}
// does not work when n is prime  O(n^(1/4))
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
LL pollard_rho(LL n) {
  if(!(n&1)) return 2;
    while(true){
        LL y=2, x=rand()%(n-1)+1, res=1;
        for(int sz=2; res==1; sz*=2) {
            for(int i=0; i<sz && res<=1; i++) {
                x = f(x, n);
                res = __gcd(abs(x-y), n);
            }
            y = x;
        }
        if (res!=0 && res!=n) return res;
    }
}
vector<int> factor;
void get_factor(int x) {
    if(x==1) return;
    if(prime[x]) {
        factor.push_back(x);
        return;
    }
    int fac = pollard_rho(x);
    get_factor(fac); get_factor(x/fac);
}
```

4.8 Josephus Problem

```
//base1 n people count k find lastone O(n)
int jo(int n, int k){return n>1?(jo(n-1,k)+k-1)%n+1:1;}
//base0 when k<n O(klogn)
int jo(int n, int k) {
    if (n == 1) return 0;
    if (k == 1) return n - 1;
    if (k > n) return (jo(n - 1, k) + k) % n;
    int f = jo(n - n / k, k) - n % k;
    return f + (f < 0 ? n : (f / (k - 1)));
}
//base1 when k=2 fast find mth
int jo2(int n, int m, int f=0){
    if(n == 1) return 1;
    int kill = (n + f) / 2;
    if(m <= kill) return 2 * m - f;
    return 2 * jo2(n - kill, m - kill, (n ^ f) & 1) -
        (1 ^ f);
}
```

4.9 Harmonic Sum

```
struct Harmonic{
    const double gamma = 0.5772156649;
    //求第N個調和級數
    double nthHarmonic(int n){
        double result = log(n)+gamma;
        return result;
    }
    //求項數n的Sn>k
    int findNearstN(int k){
        int n = exp(k-gamma)+0.5;
        return n;
    }
    // 16n
    // n/1 + n/2 + n/3 + ... + n/n
    //就是這東西
        [20,10,6,5,4,3,2,2,2,2,1,1,1,1,1,1,1,1,1]
    //這是N以下的全因數和
    int nthHarmonicSum9(int n){
        int inv2=qpow(2,MOD-2,MOD),ans=0;
        for(int i=1;i<=n;){
            int v = n/i; int j = n/v;
            int area=(((j-i+1)%MOD)*((j+i)%MOD))%MOD*
                inv2%MOD; //梯形
            ans=(ans+v*area%MOD)%MOD;
            i=j+1;
        }
        return ans;
    }
};
```

5 Python

5.1 Decimal

```
from decimal import Decimal, getcontext, ROUND_FLOOR
getcontext().prec = 250 # set precision (MAX_PREC)
getcontext().Emax = 250 # set exponent limit (MAX_EMAX)
getcontext().rounding = ROUND_FLOOR # set round floor
itwo,two,N = Decimal(0.5),Decimal(2),200
pi = angle(Decimal(-1))
```

5.2 Fraction

```
from fractions import Fraction
import math
"""專門用來表示和操作有理數，可以進行算"""
frac1 = Fraction(1)    # 1/1
frac2 = Fraction(1, 3) # 1/3
frac3 = Fraction(0.5)  # 1/2
frac4 = Fraction('22/7') # 22/7
frac5 = Fraction(8, 16) # 自動約分為 1/2
frac9 = Fraction(22, 7)
frac9.numerator   # 22
frac9.denominator # 7
x = Fraction(math.pi)
y2 = x.limit_denominator(100) # 分母限制為 100
print(y2)  # 311/99
float(x) #轉換為浮點數
```

5.3 Misc

```
# 轉為高精度整數比，(分子, 分母)
x=0.2
x.as_integer_ratio()  # (8106479329266893,
    9007199254740992)
x.is_integer()        # 判斷是否為整數
x.__round__()         # 四捨五入
int(eval(num.replace("/","//"))) # parser string num
```