CoU_ Skadooosh

Comilla University

Ahsanul Anam Saboj

Ahnaf Hasan Shifat

Ahmed Jaki

(Allah is the best planner)

1 Contents

1 Important formulas

1.1 Prime

- The number of prime numbers less than or equal to n is approximately $\frac{n}{\ln n}$
- The k-th prime number approximately equals $k \ln k$

1.2 Properties of phi:

1. $If\ gcd(i,n) = d\ ;\ where\ 1 \le i \le n-1\ then, there\ are\ phi\left(\frac{n}{d}\right)$ possible values of i.
2. $phi(p^k) = p^k - p^{k-1}$
3. $phi(ab) = phi(a)phi(b)\frac{d}{phi(d)};\ where\ \ d = \gcd(a,b)$
4. $x^n = x^{\varphi(m)+[n\ mod\ \varphi(m)]}\ mod\ m$;where n>= $\log_2$m
5. Sum of integers that are coprime to n equals to (ϕ(n)×n)/2
6. For a given integer N, the sum of Euler Phi of each of the divisors of N equals to N
7. Given a number N, let d be a divisor of N. Then the number of pairs **a** , N, where 1≤a≤N and gcd(a,N)=d, is ϕ(N/d)

1.3 Properties of mod:

1. $ac\ =\ bc\ (mod\ m), then\ \ a\ =\ b\ mod\frac{m}{g(c,m)}$
2. $mn\ (mod\ n) = 0\ then\ the\ smallest\ number\ m\ is\ equal\ to\ lcm(n,d)$
3. $if\ p\ is\ prime, then\ (x+y)^p\ =\ x^p + y^p\ (mod\ p).$
4. $ab\ (mod\ ac)\ =\ a\ (b\ mod\ c)$

1.4 Properties of Digitsum:

a. $DigitSum(x+y)\ =\ DigitSum(DigitSum(x)\ +\ DigitSum(y))$
b. $DigitSum(x-y)\ =\ DigitSum(DigitSum(x)\ -\ DigitSum(y))$
c. $DigitSum(x*y) = DigitSum(DigitSum(x)*y))$
d. $DigitSum(x*y) = DigitSum(x*DigitSum(y))$
e. $DigitSum(x*y) = DigitSum(DigitSum(x)*DigitSum(y))$
f. $DigitSum(x^y) = DigitSum(DigitSum(x)^y)$
g. $DigitSum(x^y)\ =\ DigitSum(x^{y\%\varepsilon})\ where\ DigitSum(x^{\varepsilon})$=1

1.5 Properties of FLOOR CEIL:

1. $\left\lceil\frac{n}{m}\right\rceil = \left\lfloor\frac{n+m-1}{m}\right\rfloor = \left\lfloor\frac{n-1}{m}\right\rfloor + 1$
2. $\left\lfloor\frac{n}{m}\right\rfloor = \left\lceil\frac{n+m-1}{m}\right\rceil = \left\lceil\frac{n-1}{m}\right\rceil + 1$
3. $\sum_{k=1}^{n-1}\left\lfloor\frac{kn}{m}\right\rfloor = \frac{(m-1)(n-1)+\gcd(m,n)-1}{2}$

1.6 Bit Manipulation Macros

```
#define least_one_pos(x)  __builtin_ffs(x)
#define leading_zeros(x)  __builtin_clz(x)
#define tailing_zeros(x)  __builtin_ctz(x)
#define num_of_one(x)  __builtin_popcount(x)
#define msb(x)  32-leading_zeros(x)
```

1.7 Important Series and properties

1. $\sum_{n\ge 0} a^n z^n = \frac{1}{1-az}$
2. $\sum_{n\ge 0}\binom{m}{n} z^n = (1+z)^m \quad (m \in \mathbb{Z})$
3. $\sum_{n\ge 0}\binom{m+n-1}{n} z^n = \frac{1}{(1-z)^m} \quad (m \in \mathbb{Z})$
4. $\sum_{n\ge 0}\binom{m+n}{n} z^n = \frac{1}{(1-z)^{m+1}} \quad (m \in \mathbb{Z})$
5. $\sum_{n\ge 0}\binom{n}{m} z^n = \frac{z^m}{(1-z)^{m+1}} \quad (m \in \mathbb{N})$
6. $\sum_{n\ge 0}\binom{p+n}{m} z^n = \frac{z^{m-p}}{(1-z)^{m+1}}$
7. $\sum_{n\ge 0}\frac{z^n}{n!} = e^z$
8. $\sum_{n\ge 0} -1^{n-1}\frac{z^n}{n} = \log(1+z)$

9. Vandermonde $\binom{x+y}{n} = \sum_{k=0}^{n}\binom{x}{k}\binom{y}{n-k}$
10. $\sum_{m=0}^{n}\binom{m}{k} = \sum_{k=0}^{n}\binom{i}{k} = \binom{n+1}{k+1}$
11. $\sum_{k=0}^{n}\binom{n}{k} = 2^n$
12. $\sum_{k=0}^{m}\binom{n+k}{k} = \binom{m+n+1}{m}$
13. $\sum_{k=0}^{n}\binom{n}{k}^2 = \binom{2n}{n}$
14. $\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$

Binomial Coefficient (NCR)

```
long long ncr ( long long n,  long long r )
{
   if(n<r)
      return 0;
   ll ans=1;
   ans *= fact[n];
   ll d=fact[r];
   d*=fact[n-r];
   d%=MOD;
   an*=binpow(d,MOD-2,MOD);
   an%=MOD;
   return an;
}
```

Modular Inverse

```
// A and M need to be coprime
int x = bigmod( a, m - 2, m ); //(ax)%m = 1
// When M is not prime
int modInv ( int a, int m ) {
 int x, y;
 ext_gcd( a, m, &x, &y );
 // Process x so that it is between 0 and m-1
 x %= m;
 if ( x < 0 ) x += m;
 return x;
}
```

2 Number Theory

2.1 Sieve

```
vector<long long> prime;
bitset<100000>mark;
inline void sieve( long long n)
{
    mark[0]=mark[1]=1;
    long long i,j,limit=sqrt(n*1.0)+2;
    prime.emplace_back(2);
    for(i=4; i<=n; i+=2)
        mark[i]=1;
    for(i=3; i<=n; i+=2)
    {
        if(!mark[i])
        {
            prime.emplace_back(i);
            if(i<=limit)
            {
                for(j=i*i; j<=n;
j+=i*2)
                    mark[j]=1;
            }
        }
    }
}
```

2.2 Sum of divisor

```
long long SumOfDivisor(long long n)
{
    long long ans=1;
for(long long i=0;prime[i]*prime[i]<=n;
i++)
    {
        long long sum=0,p=1;
        while(n%prime[i]==0)
        {
            n/=prime[i];
            p*=prime[i];
            sum+=p;
        }
        ans*=(sum+1);
    }
    if(n>1)
        ans*=(n+1);
    return ans;
}
```

Lucas Theorem

```
void init(int n, int mod)
{
//first calucalte i^(-1)
   inv_f[1] = 1;
   for(int i = 2; i <= n; i++)
   {
      inv_f[i] = mod - 1LL * (mod / i) * inv_f[mod % i] % mod;
   }
//Calculate Inverse factorial and factorial
   inv_f[0] = f[0] = f[1] = 1;
   for(int i = 2; i <= n; i++)
   {
      f[i] = ( 1LL * f[i - 1] * i) % mod;
      inv_f[i] = (1LL * inv_f[i] * inv_f[i - 1]) % mod;
   }
}
int nCr(int n, int r, int mod)
{
   if(r > n) return 0;
   return (((1LL * f[n] * inv_f[n - r]) % mod) * inv_f[r]) % mod;
}
//Convert n to some base
vector<int> toBase(ll n, int base)
{
   vector<int> digits;
   while(n)
   {
      digits.push_back(n % base);
      n /= base;
   }
   return digits;
}
int lucas(ll n, ll r, int mod)
{
   if(r > n) return 0;
// convert n and r to base mod
   vector<int> N = toBase(n, mod);
   vector<int> R = toBase(r, mod);
//make lengths equal by filling leading digits of R with zeros
   while(R.size() < N.size())
   {
      R.push_back(0);
   }
//Calculate answer
   int ans = 1;
   for(int i = 0; i < N.size(); i++)
   {
      ans = (1LL * ans * nCr(N[i], R[i], mod)) % mod;
   }
   return ans;
}
```

2.3 Miller Robin

```
///Miller - Rabin primality test starts frome
here
const int N=1e6;
using u64 = uint64_t;
using u128 = __uint128_t;
u64 power(u64 base, u64 e, u64 md) {
   u64 result = 1;
   base %= md;
   while (e) {
      if (e & 1)
         result = (u128)result * base % md;
      base = (u128)base * base % md;
      e >>= 1;
   }
   return result;
}
bool MillerRabin(u64 n, u64 a, u64 d, int s) {
   u64 x = power(a, d, n);
   if (x == 1 || x == n - 1)
      return false;
   for (int r = 1; r < s; r++) {
      x = (u128)x * x % n;
      if (x == n - 1)
         return false;
   }
   return true;
};
bool isPrime(u64 n, int iter=5)
{
 // returns true if n is probably prime, else returns false.
   if (n < 4)
      return n == 2 || n == 3;

   int s = 0;
   u64 d = n - 1;
   while ((d & 1) == 0) {
      d >>= 1;
      s++;
   }

   for (int i = 0; i < iter; i++) {
      int a = 2 + rand() % (n - 3);
      if (MillerRabin(n, a, d, s))
         return false;
   }
   return true;
}
///Miller - Rabin primality test ends here
```

Dynamic Programming

DP (Prefix Sum)

```
void solve(int tc)
{
   cin>>n;
   a.resize(n+1);
   b.resize(n+1);
   for(int i=1; i<=n; i++) cin>>a[i];
   for(int i=1; i<=n; i++) cin>>b[i];

// memset(dp, -1, sizeof(dp));
   for(int i=a[n]; i<=b[n]; i++)
   {
      dp[n][i] = 1;
      csum[n][i] += csum[n][i-1] + 1;
   }
   for(int i=n-1; i>=1; i--)
   {
      for(int j=a[i]; j<=b[i]; j++)
      {
         int l = max(j, a[i+1]);
         int r = b[i+1];
         if(l <= r)
         {
            dp[i][j] += csum[i+1][r] - csum[i+1][l-1];
            dp[i][j] %= mod;
            if(dp[i][j] < 0) dp[i][j] += mod;
         }
         csum[i][j] = csum[i][j-1] + dp[i][j];
         csum[i][j] %= mod;
      }
   }
   ll ans = 0;
   for(int i=a[1]; i<=b[1]; i++)
   {
      ans += dp[1][i];
      ans %= mod;
   }
   cout<<ans<<endl;
}
```

2.4 Number of divisor

```
long long NumberOfDivisor(long long n)
{
 long long ans=1;
 for(long long i=0;prime[i]*prime[i]<=n;i++)
  {
    long long counter=0;
    while(n%prime[i]==0)
    {
            n/=prime[i];
            counter++;
    }
        ans*=(counter+1);
    }
    if(n>1)ans*=2;
    return ans;
}
```

2.5 Euler Phi

```
long long phi(long long n)
{
    long long result = n;
    for (long long p = 2; p * p <= n; ++p)
    {
        if (n % p == 0)
        {
            while (n % p == 0){
                n /= p; }
            result -= result / p;
        }
    }
    if (n > 1)
        result -= result / n;
    return result;
}
```

2.6 Euler phi from 1 to MAX

```
#define MAX 100000
long long phi[MAX + 7];
void generatePhi()
{
phi[1] = 0;
for (long long i = 2; i <= MAX; i++)
{
  if(!phi[i])
  {
  phi[i] = i-1;
  for(long long j=(i<<1);j<=MAX;j+=i)
  {
  if(!phi[j])
  phi[j] = j;
  phi[j] = phi[j] * (i-1) / i;
  }
 }
}
}
```

0-1 Knapsack (log2 trick)

```
vol.pb(0);  profit.pb(0);
for(int i=0; i<N; i++)
{
   ll n, v, p;
   cin>>n>>v>>p;
   ll now = 1, cntSum = 0;
   while(cntSum + now <= n)
   {
// if(v * now > 5000) break;
      vol.pb(v * now);
      profit.pb(p * now);
      cntSum += now;
      now *= 2;
   }
   if(n - cntSum >= 0)
   {
      vol.pb(v * (n - cntSum));
      profit.pb(p * (n - cntSum));
   }
}
```

LIS (nlogn)

```
void solve(int tc)
{
   int n;
   cin>>n;
   vii v(n+1);
   for(int i=1; i<=n; i++) cin>>v[i];
   int inf = 1e9;
   vii ar(n+2, inf);
   v[0] *= -1;
   for(int i=1; i<=n; i++)
   {
      int l = upper_bound( all(ar), v[i]) - ar.begin();
      if(ar[l-1] < v[i] && v[i]<ar[l])
      {
         ar[l] = v[i];
      }
   }
   int ans = 0;
   for(int i=0; i<=n; i++)
   {
      if(ar[i] < inf) ans = i;
   }
   cout<<ans+1<<endl;
}
```

3 Data structure

3.1 Segment tree

```
//p=1 begin=0 end=n-1
vector<long long> tree, arr, lazy;
void build(int p, int begin, int end) {
    if (begin == end) {
        tree[p] = arr[begin];
        return;
    }
    int left = p << 1;
    int right = (p << 1) + 1;
    int mid = (begin + end) >> 1;
    build(left, begin, mid);
    build(right, mid + 1, end);
    tree[p] = min(tree[left], tree[right]);
}
void update_lazy(int p, int begin, int end) {
    tree[p] += lazy[p];
    if (begin != end) {
        int left = p << 1;
        int right = (p << 1) + 1;
        lazy[left] += lazy[p];
        lazy[right] += lazy[p];
    }
    lazy[p] = 0;
}
void update(int p, int begin, int end, int l, int r, long long value) {
    if (lazy[p] != 0)
        update_lazy(p, begin, end);
    if (l > end || r < begin)
        return;
    if (begin >= l && end <= r) {
        lazy[p] += value;
        update_lazy(p, begin, end);
        return;
    }
    int left = p << 1;
    int right = (p << 1) + 1;
    int mid = (begin + end) >> 1;
    update(left, begin, mid, l, r, value);
    update(right, mid + 1, end, l, r, value);
    tree[p] = min(tree[left], tree[right]);
}
long long query(int p, int begin, int end, int l, int r) {
    if (lazy[p] != 0)
        update_lazy(p, begin, end);
    if (l > end || r < begin)
        return LLONG_MAX;
    if (begin >= l && end <= r)
        return tree[p];
    int left = p << 1;
    int right = (p << 1) + 1;
    int mid = (begin + end) >> 1;
    long long a = query(left, begin, mid, l, r);
    long long b = query(right, mid + 1, end, l, r);
    return min(a, b);
}
void segment_tree(vector<long long> temp) {
    arr = temp;
    tree.resize(4 * arr.size());
    build(1, 0, arr.size() - 1);
    lazy.assign(4 * arr.size(), 0LL);
}
```

3.2 Sqrt Decomposition

```
vector<long long> vcr;
vector<vector<long long >> blocks;
long long N, block_size;
void initialize() {
    block_size = sqrt(N);
    long long block_no = -1;
    for (int i = 0; i < N; ++i) {
        if (i % block_size == 0) {
            block_no++;
            vector<long long> s;
            blocks.emplace_back(s);
        }
        blocks[block_no].push_back(vcr[i]);
    }
    for (auto &i: blocks) {
        sort(i.begin(), i.end());
    }

}
void query(int l, int r, long long v, int p, long long u) {
    long long k = 0;
    int l1 = l;
    while (l % block_size && l <= r) {
        if (vcr[l] < v)
            k++;
        l++;
    }
    while (l + block_size <= r) {
        int sz=l / block_size;
        k += lower_bound(blocks[sz].begin(), blocks[sz].end(), v) -
blocks[sz].begin();
        l += block_size;
    }
    while (l <= r) {
        if (vcr[l] < v)
            k++;
        l++;
    }
    int sz=p / block_size;
    int x = lower_bound(blocks[sz].begin(), blocks[sz].end(), vcr[p])-
blocks[sz].begin();
    blocks[sz][x] = (u * k) / (r - l1 + 1);
    vcr[p] = (u * k) / (r - l1 + 1);
    sort(blocks[sz].begin(), blocks[sz].end());
}
void print_array() {
    for (auto &i: vcr) {
        cout << i << endl;
    }
}
void sqrt_Decomposition(vector<long long> &vc) {
    N = vc.size();
    vcr = vc;
    initialize();
}
```

3.3 DSU

```
vector<long long> parent, siz;

void disjointSet(long long n) {
    parent.resize(n), siz.resize(n, 1);
    iota(parent.begin(), parent.end(), 0);
}

long long find_root(long long i) {
    while (parent[i] != i) {
        parent[i] = parent[parent[i]];
        i = parent[i];
    }
    return i;
}

void weighted_union(long long a, long long b) {
    long long root_a = find_root(a);
    long long root_b = find_root(b);
    if (root_a == root_b)
        return;
    if (siz[root_a] >= siz[root_b])
        swap(root_a, root_b);
    parent[root_a] = parent[root_b], siz[root_b] += siz[root_a];
}

bool is_connected(long long a, long long b) {
    return find_root(a) == find_root(b);
}
```

3.4 Trie

```
vector<vector<int>> trie_tree;
int min_val = '0', total_nodes = 0;
vector<int> newnode;
void Trie(int keys) {
    newnode.resize(keys, -1);
    trie_tree.emplace_back(newnode);

}
void push(string &s) {
    int level = 0;
    for (int i = 0; i < s.size(); ++i) {
        if (trie_tree[level][s[i] - min_val] == -1) {
            trie_tree[level][s[i] - min_val] = ++total_nodes;
            trie_tree.emplace_back(newnode);
        }
        level = trie_tree[level][s[i] - min_val];
    }
}
long long search(string &s) {
    long long level = 0,value = 0, j = s.size() - 1;
    for (int i = 0; i < s.size(); ++i, --j) {
        if (trie_tree[level][(s[i] - min_val) ^ 1] == -1)
            level = trie_tree[level][(s[i] - min_val)];
        else {
            value = value | (1LL << j);
            level = trie_tree[level][(s[i] - min_val) ^ 1];
        }
    }
    return value;
}
```

3.5 Ordered Set

```
#include<ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template<typename T> using ordered_set =
tree<T,null_type,less<T>,rb_tree_tag,tree_order_statistics_node_update>;
gp_hash_table<int, int> table;
```

3.6 Sparse Table

```
vector<long long >ara;
vector<vector<long long >>BiT;
long long lim,N;
void compute_ST()
{
    for(int i=0;i<N;i++)BiT[0][i]=i;
    for(long long k=1;(1<<k)<N;k++){
        for(long long i=0;i+(1<<k)<=N;i++){
            long long x=BiT[k-1][i];
            long long y=BiT[k-1][i+(1<<k-1)];
            BiT[k][i]=ara[x]<=ara[y] ? x : y;
        }
    }
}
void Sparse_table(long long N,vector<long long >&ara)
{
    ara=ara;
    N=N;
    lim=64-__builtin_clz(N);
    BiT.resize(lim,vector<long long >(N));
    compute_ST();
}
long long query(long long i,long long j)
{
    long long k=log2(j-i);
    long long x=BiT[k][i];
    long long y=BiT[k][j-(1<<k)+1];
    return ara[x]<=ara[y] ? x: y;
}
```

4 Geometry

4.1 Triangle

Circumcircle	$r = \frac{abc}{\sqrt{(a+b+c)(b+c-a)(c+a-b)(a+b-c)}}$ $r = \frac{abc}{4 \times AreaOfTriangle}$
Incircle Radius	$\frac{1}{2} \times \mathrm{r(a+b+c)} = \mathrm{AreaOfTriangle}$
Excircle Radius (If the circle is tangent to side a of the triangle)	$r = IncircleRadius \times \frac{a+b+c}{(b+c-a)}$ $r = 2 \times \frac{AreaOfTriangle}{b+c-a}$
Heron's Formula	$\sqrt{s(s-a)(s-b)(s-c)}$
Sine & Cosine rule	$\frac{a}{sinA} = \frac{b}{sinB} = \frac{c}{sinC} = 2R$ $a^2 = b^2 + c^2 - 2bcCosA$

4.2 Circle

Arc Length	$s = r\theta$ (angle in radian)
Sector Area	$\text{area} = \frac{\theta}{2} \times r^2$ (angle in radian)
Chord length	$d = 2 \times r \times \sin\left(\frac{\theta}{2}\right)$ (angle in radian) $d = 2 \times \sqrt{r^2 - x^2}$ (x = Perpendicular Distance from the Centre to Chord)
Outside one another	$C1C2 > r1 + r2$
Touching externally	$C1C2 = r1 + r2$
Intersecting at 2 points	$\lvert r1 + r2 \rvert < C1C2 < r1 + r2$
Touching internally	$C1C2 = \lvert r1 - r2 \rvert$
One inside the other	$C1C2 < \lvert r1 - r2 \rvert$

4.3 Others

Cube	$area = 6a^2$ $volume = a^3$
Cylinder	$area = 2\pi rh + 2\pi r^2$ $volume = \pi r^2 h$
Cone	$area = \pi rl$ $volume = \frac{1}{3}\pi r^2 h$
sphere	$area = 4\pi r\ 2$ $volume = 4\ 3\ \pi r3$

Operation to Make Palindrome

```
int func(int i, int j)
{
   if(i <= 0 || j<=0) return 0;
   if(dp[i][j] != -1) return dp[i][j];
   int sum = 0;
   if(s[i-1] == ss[j-1])
   {
      sum = max(sum, func(i-1, j-1) + 1);
   }
   else
   {
      sum = max(sum, max(func(i-1, j), func(i, j-1)));
   }
   return dp[i][j] = sum;
}
```

Longest Common Subsequence

```
int func (int i, int j)
{
   if(i <= 0 || j<=0) return 0;
   if(dp[i][j] != -1) return dp[i][j];
   int sum = 0;
   if(s[i-1] == ss[j-1])
   {
      sum = max( sum, func( i-1,  j-1) +  1);
   }
   else
   {
      sum = max( sum, max( func(i-1,  j),  func(i,  j-1)));
   }
   return dp[i][j] = sum;
}
```

5 Graph

5.1 Articulation point

```
bitset<10017> is_visited;
vector<long long> low, dtime;
set<long long>artipoint;
vector<vector<long long>> adjlist;
int minutes;
void articulationpoints(long long u, long long p = -1) {
    ++minutes;
    is_visited[u] = true;
    low[u] = dtime[u] = minutes;
    int child = 0;
    for (auto i:adjlist[u]) {
        if (i == p)
            continue;
        if (is_visited[i]) {
            low[u] = min(low[u], dtime[i]);
        } else {
            articulationpoints(i, u);
            low[u] = min(low[u], low[i]);
            if (dtime[u] <= low[i] && p != -1)
                artipoint.insert(u);
            child++;
        }
    }
    if (p == -1 && child > 1)
        artipoint.insert(u);
}
```

5.2 Dijkstra

```
vector<long long> dis;
vector<int> parent;
vector<vector<pair<int, int>>> adjlist;
void Dijkstra(int node, int source = 0) {
    dis.assign(node, LLONG_MAX);
    parent.assign(node, -1);
    dis[source] = 0;
    priority_queue<pair<long long, int>> pq;
    pq.push({0, source});
    bitset<100007> processed;
    while (!pq.empty()) {
        int cur_node = pq.top().second;
        pq.pop();
        if (processed[cur_node])
            continue;
        processed[cur_node] = 1;
        for (auto &i : adjlist[cur_node]) {
            int x = i.first;
            long long w = i.second;
            if (dis[cur_node] + w < dis[x]) {
                dis[x] = dis[cur_node] + w;
                parent[x] = cur_node;
                pq.push({-dis[x], x});
            }
        }
    }
}
```

5.3 Bellmenford

```
vector<long long>Node[100005],cost[100005];
long long n,m,i,j,cc=0,k;
long long dis[100005],parent[100005];
long long inf=10e9;
void bellmenford(long long s,long long f)
{
    for(i=1;i<=n;i++){
        if(i==s)dis[i]=0;else dis[i]=inf;
        parent[i]=-1;
    }
    for(i=1;i<n;i++){
        bool done=true;
        for(j=1;j<=n;j++){
            for(k=0;k<Node[j].size();k++){
                long long u=j,v=Node[j][k],uv=cost[j][k];
                if(dis[u]+uv<dis[v]){
                    dis[v]=dis[u]+uv;
                    parent[v]=u;
                    done=false;
                }
            }
        }
        if(done)break;/// there was nothing to update ;
    }
    /// Looking for Cycle ;
    bool found=true;
    for(i=1;i<=n;i++){
        for(j=0;j<Node[i].size();j++){
            long long u=i,v=Node[i][j],uv=cost[i][j];
            if(dis[u]+uv<dis[v]){
                cout<<"Found Negative Cycle"<<endl;
                found=false;
                return;
            }
        }
        if(!found)break;
    }
    for(i=1;i<=n;i++)
        cout<<"NODE : "<<i<<" distance : "<<dis[i]<<endl;
}
```

5.4 Floyed Warshal

```
long long n,i,j,cc=0,m,k;
long long adj[100][100];
long long path[100][100];
void floyed_Warshal()
{
    for(k=1;k<=n;k++){
        for(i=1;i<=n;i++){
            for(j=1;j<=n;j++)
                if(adj[i][k]+adj[k][j]<adj[i][j]){
                    adj[i][j]=adj[i][k]+adj[k][j];
                    path[i][j]=path[i][k];
                }
        }
    }
}
```

5.5 Ford Fulkerson

```
const int maX=1e5+5;
typedef vector<vector<long long>>v1;
v1 Graph;
long long capacity[1000][1000];
long long n,m;
void init(int N)
{
    Graph=v1(N+1);
}
long long bfs(long long s,long long t,vector<long long>&parent)
{
    fill(parent.begin(),parent.end(),-1);
    parent[s]=-2;
    queue<pair<long long,long long>>q;
    q.push({s,INT_MAX});
    while(!q.empty()){
        long long u=q.front().first;
        long long flow=q.front().second;
        q.pop();
        for(long long i=0;i<Graph[u].size();i++){
            long long v=Graph[u][i];
            if(parent[v]==-1 && capacity[u][v]){
                parent[v]=u;
                long long new_flow=min(flow,capacity[u][v]);
                cout<<v<<" ";
                if(v==t)return new_flow;
                q.push({v,new_flow});
            }
        }
    }
    return 0;
}
long long max_flow(long long s,long long t)
{
    vector<long long>parent(n+1);
    long long flow=0;
    long long new_flow;
    while( new_flow=bfs(s,t,parent)){
        cout<<endl;
        cout<<new_flow<<endl;
        flow+=new_flow;
        long long u=t;
        while(s != u){
            long long prev=parent[u];
            capacity[prev][u]-=new_flow;
            capacity[u][prev]+=new_flow;
            u=prev;
        }
    }
    return flow;
}
```

5.6 Prim

```
const int maX=1e5+5;
long long nodes,edges;
bool visit[maX];
vector<pair<long long,long long>>adj[maX];
long long prim(long long x)
{long long i,j,minimumcost=0,cost;
    priority_queue<pair<long long,long long>,vector<pair<long
    long,long long>>,greater<pair<long long,long long>>> Q;
    pair<long long,long long> p;
    Q.push({0,x});
    while(! Q.empty()){
        p=Q.top();
        Q.pop();
        x=p.second;
        if(visit[x]==true)continue;
        visit[x]=true;
        minimumcost+=p.first;
        for(i=0;i<adj[x].size();i++){
            long long y=adj[x][i].second;
            if(visit[y]==false)Q.push(adj[x][i]);
        }
    }
    return minimumcost;
}
```

5.7 Kruskal

```
const int maX=1e5+5;
long long id[maX],nodes,edges;
pair<long long,pair<long long,long long>>p[maX];
void initialize()
{
    for(int i=1;i<maX;i++)id[i]=i;
}
long long root(long long x)
{
    while(x != id[x])id[x]=id[id[x]],x=id[x];
    return x;
}
void union1(long long x,long long y)
{
    long long p=root(x);long long q=root(y);id[p]=id[q];
}
long long kruskal(pair<long long,pair<long long,long long>>p[])
{
    long long x,y,cost,minimumcost=0,i;
    for(i=0;i<edges;i++){
        x=p[i].second.first; y=p[i].second.second;cost=p[i].first;
        if(root(x) != root(y)){
            minimumcost+=cost;
            union1(x,y);
        }
    }
    return minimumcost;
}
```

5.8 LCA

```
const int LOG = 20;
vii G[N], depth(N, 0);
int up[N][LOG];

void dfs (int a)
{
    for( auto it: G[a])
    {
        depth[it] = 1 + depth[a];

        // Binary Lifting
        up[it][0] = a;
        for (int j=1; j<LOG; j++)
        {
            up[it][j] = up[ up[it][j-1] ][j-1];
        }

        dfs(it);
    }
}

int get_lca( int a, int b)
{
    if(depth[a] < depth[b]) swap(a, b);
    int k = depth[a] - depth[b];
    for ( int  j=LOG-1; j>=0; j--)
    {
        If (checkbit(k, j)) a = up[a][j];
    }
    if(a == b) return a;

    for( int  j=LOG-1; j>=0; j--)
    {
        If ( up[a][j] != up[b][j] )
        {
            a = up[a][j];
            b = up[b][j];
        }
    }
    return up[a][0];
}
```

Centroid Decomposition

```
vector<vector<int>> G;
vector<bool> is_removed;
vector<int> sub, par;

int get_subtree_size(int node, int parent = -1)
{
    sub[node] = 1;
    for (int child : G[node])
    {
        if (child == parent || is_removed[child])
        {
            continue;
        }
        sub[node] += get_subtree_size(child, node);
    }
    return sub[node];
}

int get_centroid(int node, int tree_size, int parent = -1)
{
    for (int child : G[node])
    {
        if (child == parent || is_removed[child])
        {
            continue;
        }
        if (sub[child] * 2 > tree_size)
        {
            return get_centroid(child, tree_size, node);
        }
    }
    return node;
}

void build_centroid_decomp( int node = 0, int parent=-1)
{
    int centroid = get_centroid(node, get_subtree_size(node));
    par[centroid] = parent;
    // do something
    is_removed[centroid] = true;

    for (int child : G[centroid])
    {
        if (is_removed[child])
        {
            continue;
        }
        build_centroid_decomp(child, centroid);
    }
}
```

5.9 SCC

```
const int maX=1e5+5;
vector<long long>Graph[maX],Re_Graph[maX],check[maX];
long long visit[maX];
stack<long long>ans;
void dfs(long long u)
{
    visit[u]=1;
    for(long long i=0;i<Graph[u].size();i++){
        long long v=Graph[u][i];
        if(!visit[v]){
            dfs(v);
        }
    }
    ans.push(u);
}
void dfs2(long long u,long long mark)
{
    check[mark].emplace_back(u);
    visit[u]=1;
    for(long long i=0;i<Re_Graph[u].size();i++){
        long long v=Re_Graph[u][i];
        if(visit[v]==0){
            dfs2(v,mark);
        }
    }
}
```

5.10 Topsort

```
const int maX=1e5+5;
vector<long long>ara[maX],cost[maX];
bitset<maX> visit;
long long start[maX],finish[maX];
long long cnt=0;
vector<long long>ans;
void dfs(long long source)
{
    cnt++;
    visit[source]=1;
    start[source]=cnt;
    for(int i=0;i<ara[source].size();i++){
        long long y=ara[source][i];
        if(visit[y]==0){
            dfs(y);
        }
    }
    cnt++;
    ans.emplace_back(source);
    finish[source]=cnt;
    return;
}
```

6 String

6.1 KMP

```
template<typename T>
class kmp {
    vector<int> indx;
public:
    void lps(T &patt) {
        indx.resize(patt.size(), 0);
        int i = 0, j = 1;
        while (j < patt.size()) {
            if (patt[i] == patt[j])
                indx[j] = ++i, j++;
            else {
                if (i != 0)
                    i = indx[i - 1];
                else
                    indx[j] = 0, j++;
            }
        }
    }

    bool match(T &text, T &patt) {
        int i = 0, j = 0;
        while (j < text.size()) {
            if (patt[i] == text[j])
                i++, j++;
            else {
                if (i != 0)
                    i = indx[i - 1];
                else
                    j++;
            }
            if (i == patt.size())
                return true;
        }
        return false;
    }

    int frequency(T &text, T &patt) {
        int i = 0, j = 0, cnt = 0;
        while (j < text.size()) {
            if (patt[i] == text[j])
                i++, j++;
            else {
                if (i != 0)
                    i = indx[i - 1];
                else
                    j++;
            }
            if (i == patt.size())
                cnt++, i = indx[i -
1];
        }
        return cnt;
    }
};
```

Some Properties/Techniques of Number Theory

1. How many numbers are coprime, from **1 to N*M,** where N and M are coprime ,GCD(N,M)=1
    1. With N but not with M
    2. With M but not with N
    3. With both M and N

    Ans1 = (phi(N)*M)-phi(N*M)
    Ans2 = (phi(M)*N)-phi(N*M)
    Ans3 = phi(M*N)
2. Divisors sum of every number from1 to 2*10^9 ?

Ans = sod(1) + sod(2)+sod(3) + sod(4)+…….+sod(n);

```
long long sum_all_divisors(long long num)
{
  long long sum = 0;
  for (long long i = 1; i <= sqrt(num); i++) {
    long long t1 = i * (num / i - i + 1);
    long long t2 = (((num / i) * (num / i + 1)) / 2) - ((i * (i + 1)) / 2);
    sum += t1 + t2;
  }
  return sum;
}
```

3. If gcd(x,n) = 1 then gcd(n-x,n) = 1;

4. $\log_k(number) = \frac{\log_{10}(number)}{\log_{10} k}$ // This is used for base conversion from decimal to k base.

5.

```
int power(long long n, long long k, const int mod) {///Shifater HAshing
  int ans = 1 % mod;
  n %= mod;
  if (n < 0) n += mod;
  while (k) {
    if (k & 1) ans = (long long) ans * n % mod;
    n = (long long) n * n % mod;
    k >>= 1;
  }
  return ans;
}

const int MOD1 = 127657753, MOD2 = 987654319;
const int p1 = 137, p2 = 277;
int ip1, ip2;
pair<int, int> pw[N], ipw[N];

void prec() {
  pw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    pw[i].first = 1LL * pw[i - 1].first * p1 % MOD1;
    pw[i].second = 1LL * pw[i - 1].second * p2 % MOD2;
  }
  ip1 = power(p1, MOD1 - 2, MOD1);
  ip2 = power(p2, MOD2 - 2, MOD2);
  ipw[0] =  {1, 1};
  for (int i = 1; i < N; i++) {
    ipw[i].first = 1LL * ipw[i - 1].first * ip1 % MOD1;
    ipw[i].second = 1LL * ipw[i - 1].second * ip2 % MOD2;
  }

}

struct Hashing {
  int n;
  string s; // 0 - indexed
  vector<pair<int, int>> hs; // 1 - indexed

  Hashing() {}

  Hashing(string _s) {
    n = _s.size();
    s = _s;
    hs.emplace_back(0, 0);
    for (int i = 0; i < n; i++) {
      pair<int, int> p;
      p.first = (hs[i].first + 1LL * pw[i].first * s[i] % MOD1) % MOD1;
      p.second = (hs[i].second + 1LL * pw[i].second * s[i] % MOD2) %
MOD2;
      hs.push_back(p);
    }
  }

  pair<int, int> get_hash(int l, int r) { // 1 - indexed
    assert(1 <= l && l <= r && r <= n);
    pair<int, int> ans;
    ans.first = (hs[r].first - hs[l - 1].first + MOD1) * 1LL * ipw[l - 1].first %
MOD1;
    ans.second = (hs[r].second - hs[l - 1].second + MOD2) * 1LL * ipw[l -
1].second % MOD2;
    return ans;
  }
  pair<int, int> get_hash() {
    return get_hash(1, n);
  }
};
```

```
#include <bits/stdc++.h> /////Jakir Hashing
#define ff first
#define ss second
#define mp make_pair
#define ll long long
using namespace std;
typedef long long LL;
typedef pair<LL, LL> PLL;
///============================================///
const PLL M = mp(1088888881, 1481481481); ///Should be large primes
const LL base = 347;          ///Should be a prime larger than highest value
const int N = 1e6 + 7;        ///Highest length of string
ostream& operator<<(ostream& os, PLL hash)
{
    return os << "(" << hash.ff << ", " << hash.ss << ")";
}
PLL operator+ (PLL a, LL x)
{
    return mp(a.ff + x, a.ss + x);
}
PLL operator- (PLL a, LL x)
{
    return mp(a.ff - x, a.ss - x);
}
PLL operator* (PLL a, LL x)
{
    return mp(a.ff * x, a.ss * x);
}
PLL operator+ (PLL a, PLL x)
{
    return mp(a.ff + x.ff, a.ss + x.ss);
}
PLL operator- (PLL a, PLL x)
{
    return mp(a.ff - x.ff, a.ss - x.ss);
}
PLL operator* (PLL a, PLL x)
{
    return mp(a.ff * x.ff, a.ss * x.ss);
}
PLL operator% (PLL a, PLL m)
{
    return mp(a.ff % m.ff, a.ss % m.ss);
}
PLL power (PLL a, LL p)
{
    if (p == 0)   return mp(1, 1);
    PLL ans = power(a, p / 2);
    ans = (ans * ans) % M;
    if (p % 2)    ans = (ans * a) % M;
    return ans;
}
PLL inverse(PLL a)///Magic!!!!!!
{
    return power(a, (M.ff - 1) * (M.ss - 1) - 1);
}
PLL pb[N];      ///powers of base mod M
PLL invb;
void hashPre()///Call pre before everything
{
    pb[0] = mp(1, 1);
    for (int i = 1; i < N; i++)
        pb[i] = (pb[i - 1] * base) % M;
    invb = inverse(pb[1]);
}
PLL Hash (string s)///Calculates Hash of a string
{
    PLL ans = mp(0, 0);
    for (int i = 0; i < s.size(); i++)
        ans = (ans * base + s[i]) % M;
    return ans;
}
PLL append(PLL cur, char c)///appends c to string
{
    return (cur * base + c) % M;
}
```

6.4 Manachar Algorithm to Find longest Palindrome

```
vector<int> manacher_odd(string s)
{
   int n = s.size();
   s = "$" + s + "^";
   vector<int> p(n + 2);
   int l = 1, r = 1;
   for(int i = 1; i <= n; i++)
   {
      p[i] = max(0, min(r - i, p[l + (r - i)]));
      while(s[i - p[i]] == s[i + p[i]])
      {
         p[i]++;
      }
      if(i + p[i] > r)
      {
         l = i - p[i], r = i + p[i];
      }
   }
   return vector<int>(begin(p) + 1, end(p) - 1);
}
```

```
#define olta(a) reverse(a.begin(),a.end())
#define mem(a,b) memset(a,b,sizeof(a))
#define rsrt(v) sort(v.rbegin(),v.rend());
#definegsrt(a)sort(a.begin(), a.end(), greater<ll>())
#define vp vector<pair<ll, ll> >
#define v_min(a) *min_element(a.begin(),a.end())
#define v_max(a) *max_element(a.begin(),a.end())
#define v_mini(v) min_element(v.begin(),v.end()) - v.begin();
#define v_maxi(v) max_element(v.begin(),v.end()) - v.begin();
#define v_sum(a) accumulate(a.begin(),a.end(),0)
#define un(a) a.erase(unique(a.begin(),a.end()),a.end())
#define delete(a) a.erase(a.begin(),a.end())
#define Sort(a) sort(a.begin(),a.end())
#define is(a) is_sorted(a.begin(),a.end())
#define Saboj4632 ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0)
#define lcm(a,b) ((a)*(b))/gcd(a,b)
#define pi 3.141592653589793
```

```
PLL prepend(PLL cur, int k, char c)///prepends c to string with size k
{
   return (pb[k] * c + cur) % M;
}
PLL replace(PLL cur, int i, char a, char b)///replaces the i-th (0-indexed) character from right from a to b;
{
   cur = (cur + pb[i] * (b - a)) % M;
   return (cur + M) % M;
}
PLL pop_back(PLL hash, char c)///Erases c from the back of the string
{
   return (((hash - c) * invb) % M + M) % M;
}
PLL pop_front(PLL hash, int len, char c)///Erases c from front of the string with size len
{
   return ((hash - pb[len - 1] * c) % M + M) % M;
}
PLL concat(PLL left, PLL right, int k)///concatenates two strings where length of the right is k
{
   return (left * pb[k] + right) % M;
}
///Calculates hash of string with size len repeated cnt times
///This is O(log n). For O(1), pre-calculate inverses
PLL repeat(PLL hash, int len, LL cnt)
{
   PLL mul = (pb[len * cnt] - 1) * inverse(pb[len] - 1);
   mul = (mul % M + M) % M;
   PLL ans = (hash * mul) % M;
   if (pb[len].ff == 1)   ans.ff = hash.ff * cnt;
   if (pb[len].ss == 1)   ans.ss = hash.ss * cnt;
   return ans;
}
///Calculates hashes of all prefixes of s including empty prefix
vector<PLL> hashList(string &s)
{
   int n = s.size();
   vector<PLL> ans(n + 1);
   ans[0] = mp(0, 0);
   for (int i = 1; i <= n; i++) ans[i] = (ans[i - 1] * base + s[i - 1]) % M;
   return ans;
}
///Calculates hash of substring s[l..r] (1 indexed)
PLL substringHash(const vector<PLL> &hashlist, int l, int r)
{
   int len = (r - l + 1);
   return ((hashlist[r] - hashlist[l - 1] * pb[len]) % M + M) % M;
}
void solve()
{
   ll n, q;
   cin>>n; string s; cin>>s;
   vector<PLL>Hash_A = hashList(s);
   /// NOW HEAT THE PROBLEM...
}
int main()
{
   hashPre();
   solve();
   return 0;
}
```

#Dijkstra

```
int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    ll tst = 0;
    ll n,k,m,i,j,x,y,w;
    cin>>n>>m;
    ll dis[n+1];
    ll vis[n+1];
    map<ll,ll>path;
    for(i=0; i<=n; i++)
    {
        dis[i] = INF;
        vis[i] = 0;
    }
    vp v[n+1];
    for(i=0; i<m; i++)
    {
        cin>>x>>y>>w;
        v[x].pb(mp(y,w));
        v[y].pb(mp(x,w));
    }
    //  cout<<"Saboj"<<endl;

priority_queue<pair<ll,ll>,vp,greater<pair<l
l,ll>>>s;
    s.push(mp(0,1));
    dis[1] = 0;
    while(!s.empty())
    {
        ll vv,d;
        vv = s.top().S;
        d = s.top().F;
        s.pop();
//          //  cout<<vv<<" "<<d<<endl;
        for(auto child:v[vv])
        {
            x = child.F;
            y = child.S;
            //cout<<x<<" "<<y<<endl;
            if((dis[vv]+y)<dis[x])
            {
                path[x] = vv;
                dis[x] = dis[vv] + y;
                s.push(mp(dis[x],x));
            }
        }
    }
    vecl ans;
    if(dis[n]==INF)cout<<-1<<endl;
    else {
        x = n;
        ans.pb(x);
        while(x>1){
            x = path[x];
            ans.pb(x);
        }
        for(i=ans.size()-1;i>=0;i--
)cout<<ans[i]<<" ";
        cout<<endl;
    }
    return 0;

}
```

Merge Sort Tree (Jaki's)

```
const ll N = 3e4+3;
vector<ll> Tree[4*N], v(N);
ll n;
void mergee(ll node,ll l,ll mid,ll r)
{
   ll n1 = mid-l+1;
   ll n2 = r-mid;
   ll ar1[n1+1], ar2[n2+1];
   for(ll i=1; i<=n1; i++) ar1[i]=v[l+i-1];
   for(ll i=1; i<=n2; i++) ar2[i]=v[mid+i];
   ll i=1, j=1, cnt=1;
   while(i<=n1 && j<=n2)
   {
      if(ar1[i]<ar2[j])
      {
         v[l+cnt-1] = ar1[i];
         i++, cnt++;
      }
      else
      {
         v[l+cnt-1]=ar2[j];
         j++, cnt++;
      }
   }
while(i<=n1){ v[l+cnt-1]=ar1[i]; cnt++; i++;}
while(j<=n2){ v[l+cnt-1]=ar2[j]; cnt++; j++; }
for(ll i=l; i<=r;i++) Tree[node].push_back(v[i]);

}
void mergeSort(ll node,ll l,ll r)
{
if(l==r) {Tree[node].push_back(v[l]); return;}
   if(l<r)
   {
      ll mid = (l+r)/2;
      ll left = 2*node;
      ll right = left+1;
      mergeSort(left, l, mid);
      mergeSort(right, mid+1, r);
      mergee(node, l, mid, r);
   }
}
ll querry(ll node,ll l,ll r,ll i,ll j,ll x)
{
   if(i>r || j<l) return 0;
   if(i<=l && j>=r)
   {
   ll up = upper_bound(Tree[node].begin(),
   Tree[node].end(), x)-Tree[node].begin();
   ll nn = r-l+1;
   return nn-up;
   }
   ll left = node*2;
   ll right = left+1;
   ll mid = (l+r)/2;
   ll a = querry(left, l, mid, i, j, x);
   ll b = querry(right, mid+1, r, i, j, x);
   return a+b;
}
```

```
____//string matching
int pi[200001]; string txt,pat;
void  solve()
{
    int i,j,res=0;
    cin>>pat>>txt;
    string st=pat+"#"+txt;
    for(i=1;i<st.size();i++)
    {
        j=pi[i-1];
     while(j>0 && st[i]!=st[j])
 j=pi[j-1];
        if(st[i]==st[j]) j++;
        pi[i]=j;
    }

for(i=pat.size();i<st.size();i++)
    {
        if(pi[i]==pat.size())
        {
            res++;
            if(res>0) break;
        }
    }
}
//LPS¬_ARRAY
vector<int> lps;
void createLPS(string pat){
    lps.push_back(0);
    int i = 0, j = 1;
    for ( ; pat[j]; ){
        if (pat[i] == pat[j]){
            lps.push_back(i+1);
            i++;j++;
        }
        else{
            if (i != 0) i =
lps[i-1];
            else {
                j++;
                lps.push_back(0);
            }
        }
    }
}
```

//KMP

```
char
txt[1000009],pat[1000009];
void lps_ar(char *pat,int
M,int *lps)
{
 int len=0;lps[0]=0; int i=1;
    while(i<M)
    {
   if(pat[i]==pat[len])
        len++,lps[i]=len,i++;
        else
        {
        if(len!=0)
len=lps[len-1];
         else lps[i]=0,i++;
        }
    }
}
void KMPsearch(char *txt,char
*pat)
{
    int N=strlen(txt);
    int M=strlen(pat);
    int lps[M];
    lps_ar(pat,M,lps);
    int i=0,j=0; total=0;
    while(i<N)
    {
        //cout<<"kmp"<<endl;
 if(pat[j]==txt[i]) i++,j++;
        if(j==M)
        {
          //to print how many
times match
        total++;
//cout<<"found pattern at
index: "<<i-j<<endl;
            j=lps[j-1];
        }
else if(i<N &&
pat[j]!=txt[i])
        {
     if(j!=0) j=lps[j-1];
       else i++;
        }
    }
}
```

//STRING MULTIPLACATION

```
string multyply(string a,int b)

{
    int carry = 0;
    ans = "";
for(int i=0;i<a.size();i++)
{
carry=((a[i]-'0')*b+carry);
 ans += carry % 10 + '0';
     carry /= 10;
    }
  while(carry != 0){
 ans += carry % 10  + '0';
        carry /= 10;
    }
    return ans;
}
```

```
//find nCr
ll nCr(ll n,ll r)
{
    ll p=1,q=1;
    r=min(r,n-r);
    if(r!=0)
    {
        while(r)
        {
            p*=n;q*=r;
            ll x=__gcd(p,q);
            p/=x;q/=x;
            n--;r--;
        }
    }
    else p=1;
    return p;
}
///print power
ll power(ll x,ll n)
{
    ll res=1;
    while(n)
    {
        if(n&1) res*=x;
        x*=x;
        n>>=1;
    }
    return res;
}
///print power_mod
ll power_mod(ll a,ll b)
{
    ll res=1;
    while(b)
    {
        if(b&1)
res=(res*1LL*a)%MOD;
        a=(a*1LL*a)%MOD;
        b>>=1;
    }
    return res;
}
```

Farmat Little Theorem

Power_mod(n,m-2)

Here we should pass the mod as power.

```
//GCD
ll gcd(ll a,ll b){
    if(b==0 || a==0) return 0;
    if(b%a==0) return a;
    else  return gcd(b%a,a);
}
```

```
//o_1 bfs
void zeronebfs(ll x,ll
y,ll r,ll c)
{
    for(i=0; i<=r; i++)
for(j=0; j<=c; j++)
dis[i][j]=INT_MAX;
    dis[0][0]=0;
    deque<pair<ll,ll>>q;
    q.push_back({x,y});
    while(!q.empty())
    {
        auto
it=q.front();
        q.pop_front();
        ll a=it.F;
        ll b=it.S;
        for(i=0; i<4;
i++)
        {

      ll  e=a+fx[i];
       ll d=b+fy[i];
if(e>=0 && e<r && d>=0 &&
d<c)
   {
           ll z=0;

if(ar[a][b]!=ar[e][d])
z=1;

if(dis[a][b]+z<dis[e][d])
                {

dis[e][d]=dis[a][b]+z;

if(z==0)
q.push_front({e,d});
      else
q.push_back({e,d});
                }
            }
        }
    }
        cout<<dis[r-1][c-
1]<<endl;

}
```

//EULET TOTIENT (1 to N) (nlog(n))

```
void EulerTotient()
{
    phi[1] = 1;
    for (int i=2; i<MAX; i++)
    {
        if (!phi[i])
        {
            phi[i] = i-1;
 for (int j = i*2;j<MAX; j+=i)
        {
      if (!phi[ j]) phi[j] = j;
   phi[ j] = (phi [ j]/i)*(i-1);
         }   }   } }
```

// EULET TOTIENT (sqrt(n) * log(n))

```
int phi(int n) {
    int result = n;
    for(int i=2;i*i<= n; i++) {
        if (n % i == 0) {
      while (n % i == 0) n /= i;
        result -= result / i;
        }   }
    if (n > 1)
      result -= result / n;
    return result;
}
```

Fibonacci of nth Using Matrix multiplication

```
ll fib(ll n){
    if(n==1)return 0;
    if(n==2)return 1;
    ll b = n-2;
    ll x,y,z,w;
    ll f[2][2] = {{1,1},{1,0}};
    ll r[2][2] = {{1,0},{0,1}};
    if(b<0){
        return 0;
    }
    while(b>0){
        if(b&1){
            x = ((r[0][0]*f[0][0])%MAX + (r[0][1]*f[1][0])%MAX)%MAX;
            y = ((r[0][0]*f[0][1])%MAX + (r[0][1]*f[1][1])%MAX)%MAX;
            w = ((r[1][0]*f[0][0])%MAX + (r[1][1]*f[1][0])%MAX)%MAX;
            z  = ((r[1][0]*f[0][1])%MAX + (r[1][1]*f[1][1])%MAX)%MAX;
            r[0][0] = x;
            r[0][1] = y;
            r[1][0] = w;
            r[1][1] = z;
            //cout<<r[0][0]<<" r"<<endl;

        }
        // cout<<" b "<<b<<endl;

        x = ((f[0][0]*f[0][0])%MAX + (f[0][1]*f[1][0])%MAX)%MAX;
        y = ((f[0][0]*f[0][1])%MAX + (f[0][1]*f[1][1])%MAX)%MAX;
        w = ((f[1][0]*f[0][0])%MAX + (f[1][1]*f[1][0])%MAX)%MAX;
        z = ((f[1][0]*f[0][1])%MAX + (f[1][1]*f[1][1])%MAX)%MAX;
        // cout<<"X "<<x<<" y "<<y<<" w "<<w<<" "<<z<<endl;
        f[0][0] = x;
        f[0][1] = y;
        f[1][0] = w;
        f[1][1] = z;
        //  cout<<"f[0][0] "<<f[0][0]<< " "<<f[0][1]<<endl;
        b>>=1;
    }
    return r[0][0];
}
```

```
const  int  fx[]={+1,-1,+0,+0};//graph move
const  int  fy[]={+0,+0,+1,-1};//graph move
const  int  fx[]={+0,+0,+1,-1,-1,+1,-1,+1};//kings move
const  int  fy[]={-1,+1,+0,+0,+1,+1,-1,-1};//kings move
const  int  fx[]={-2,-2,-1,-1,+1,+1,+2,+2};//knight's move
const  int  fx[]={-1,+1,-2,+2,-2,+2,-1,+1};//knight's move
```