Team notebook

Relentlessly Outmatched

December 27, 2018

Contents

1 2SAT

```
int n;
vector<vector<int>> g, gt;
vector<bool> used;
vector<int> order, comp;
```

```
vector<bool> assignment;

void dfs1(int v) {
    used[v] = true;
    for (int u : g[v]) {
        if (!used[u])
            dfs1(u);
    }
    order.push_back(v);
}

void dfs2(int v, int cl) {
    comp[v] = cl;
    for (int u : gt[v]) {
        if (comp[u] == -1)
            dfs2(u, cl);
    }
}

bool solve_2SAT() {
    used.assign(n, false);
    for (int i = 0; i < n; ++i) {
        if (!used[i])
            dfs1(i);
    }

    comp.assign(n, -1);
    for (int i = 0, j = 0; i < n; ++i) {
        int v = order[n - i - 1];
        if (comp[v] == -1)
            dfs2(v, j++);
    }

    assignment.assign(n / 2, false);
    for (int i = 0; i < n; i += 2) {
        if (comp[i] == comp[i + 1])
            return false;
        assignment[i / 2] = comp[i] >
            comp[i + 1];
    }
    return true;
}
```

2 BIT

```
// Only ll
typedef struct BIT //ll
{
  vector<ll> bit;
  int n;

  void init(int n)
  {
    this->n = n;
    bit.assign(n,0);
  }

  ll sum(int i)
  {
    ll res = 0;
    for(;i>=0;i = (i&(i+1)) -1)
    {
      res+=bit[i];
    }
    return res;
  }

  void inc(int i, ll delta)
  {
    for(;i<n;i = i|(i+1))
    {
      bit[i]+=delta;
    }
  }

  ll getsum(int l,int r)
  {
    // If l==0, sum(-1) automatically
       returns the default calue of res,
       0
    return sum(r)-sum(l-1);
  }

  void init(vector<ll> v)
  {
    init(v.size());
    for(int i=0;i<v.size();i++)
       inc(i,v[i]);
  }
} BIT;
```

3 Bellman

```
struct edge
{
    int a, b, cost;
};
int n, m, v;
vector<edge> e;
const int INF = 1000000000;
void solve()
```

```
{
    vector<int> d (n, INF);
    d[v] = 0;
    for (;;)
    {
        bool any = false;
        for (int j=0; j<m; ++j)
            if (d[e[j].a] < INF)
                if (d[e[j].b] > d[e[j].a]
                    + e[j].cost)
                {
                    d[e[j].b] = d[e[j].a]
                        + e[j].cost;
                    any = true;
                }
        if (!any) break;
    }
    // display d, for example, on the
        screen
}
```

4 DSU

```
const int N = 100100;
int p[N], sz[N];

void create(int x){
       p[x] = x;
       sz[x] = 1;
       return;
}

int find(int x){
  if(x == p[x]) return x;
  return p[x] = find(p[x]);
}

void merge(int x, int y){
       int x = find(x), y = find(y);
       if(x == y) return;
  if(sz[x] < sz[y]) swap(x, y);
  p[y] = x;
  sz[x] += sz[y];
  return;
}

int main(){
       return 0;
}
```

5 DSUTree

```
int cnt[maxn];
bool big[maxn];
void add(int v, int p, int x){
    cnt[ col[v] ] += x;
    for(auto u: g[v])
        if(u != p && !big[u])
            add(u, v, x)
}
void dfs(int v, int p, bool keep){
    int mx = -1, bigChild = -1;
    for(auto u : g[v])
       if(u != p && sz[u] > mx)
          mx = sz[u], bigChild = u;
    for(auto u : g[v])
        if(u != p && u != bigChild)
            dfs(u, v, 0); // run a dfs on
                small childs and clear
                them from cnt
    if(bigChild != -1)
        dfs(bigChild, v, 1),
            big[bigChild] = 1; //
            bigChild marked as big and
            not cleared from cnt
    add(v, p, 1);
    //now cnt[c] is the number of
        vertices in subtree of vertex v
        that has color c. You can answer
        the queries easily.
    if(bigChild != -1)
        big[bigChild] = 0;
    if(keep == 0)
        add(v, p, -1);
}
```

6 Dijkstra

```
const int INF = 1000000000;
vector<vector<pair<int, int>>> adj;

void dijkstra(int s, vector<int> & d,
    vector<int> & p) {
    int n = adj.size();
    d.assign(n, INF);
    p.assign(n, -1);
    vector<bool> u(n, false);

    d[s] = 0;
```

```
    for (int i = 0; i < n; i++) {
        int v = -1;
        for (int j = 0; j < n; j++) {
            if (!u[j] && (v == -1 || d[j]
                < d[v]))
                v = j;
        }

        if (d[v] == INF)
            break;

        u[v] = true;
        for (auto edge : adj[v]) {
            int to = edge.first;
            int len = edge.second;

            if (d[v] + len < d[to]) {
                d[to] = d[v] + len;
                p[to] = v;
            }
        }
    }
}
vector<int> restore_path(int s, int t,
    vector<int> const& p) {
    vector<int> path;

    for (int v = t; v != s; v = p[v])
        path.push_back(v);
    path.push_back(s);

    reverse(path.begin(), path.end());
    return path;
}
```

7 FloydWarshall

```
// d is the adjacency matrix
for (int k = 0; k < n; ++k) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (d[i][k] < INF && d[k][j]
                < INF)
                d[i][j] = min(d[i][j],
                    d[i][k] + d[k][j]);
        }
    }
}
```

8 GCD

```
int gcd(int a, int b)
{
  if (a == 0)
    return b;
  return gcd(b % a, a);
}
int gcd(int a, int b, int & x, int & y)
    { // Extended
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    int x1, y1;
    int d = gcd(b % a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return d;
}
```

9 Geo

```
// Circle Line intersection
// For circle circle intersection, C1-C2
    is the eqn for the common chord. That
    inersects with C1 gives the points
double r, a, b, c; // given as input
double x0 = -a*c/(a*a+b*b), y0 =
    -b*c/(a*a+b*b);
if (c*c > r*r*(a*a+b*b)+EPS)
    puts ("no points");
else if (abs (c*c - r*r*(a*a+b*b)) <
    EPS) {
    puts ("1 point");
    cout << x0 << ' ' << y0 << '\n';
}
else {
    double d = r*r - c*c/(a*a+b*b);
    double mult = sqrt (d / (a*a+b*b));
    double ax, ay, bx, by;
    ax = x0 + b * mult;
    bx = x0 - b * mult;
    ay = y0 - a * mult;
    by = y0 + a * mult;
    puts ("2 points");
    cout << ax << ' ' << ay << '\n' <<
        bx << ' ' << by << '\n';
}
```

```
// Intersection point for 2 lines
struct pt {
    double x, y;
};

struct line {
    double a, b, c;
};

const double EPS = 1e-9;

double det(double a, double b, double c,
    double d) {
    return a*d - b*c;
}

bool intersect(line m, line n, pt & res)
    {
    double zn = det(m.a, m.b, n.a, n.b);
    if (abs(zn) < EPS)
        return false;
    res.x = -det(m.c, m.b, n.c, n.b) /
        zn;
    res.y = -det(m.a, m.c, n.a, n.c) /
        zn;
    return true;
}

bool parallel(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b))
        < EPS;
}

bool equivalent(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b))
        < EPS
        && abs(det(m.a, m.c, n.a, n.c)) <
            EPS
        && abs(det(m.b, m.c, n.b, n.c)) <
            EPS;
}
// Area of a polygon
double area(const vector<point>& fig) {
    double res = 0;
    for (unsigned i = 0; i < fig.size();
        i++) {
        point p = i ? fig[i - 1] :
            fig.back();
        point q = fig[i];
        res += (p.x - q.x) * (p.y + q.y);
    }
    return fabs(res) / 2;
}

// Determinant of a matrix
const double EPS = 1E-9;
int n;
vector < vector<double> > a (n,
    vector<double> (n));

double det = 1;
for (int i=0; i<n; ++i) {
    int k = i;
    for (int j=i+1; j<n; ++j)
        if (abs (a[j][i]) > abs (a[k][i]))
            k = j;
    if (abs (a[k][i]) < EPS) {
        det = 0;
        break;
    }
    swap (a[i], a[k]);
    if (i != k)
        det = -det;
    det *= a[i][i];
    for (int j=i+1; j<n; ++j)
        a[i][j] /= a[i][i];
    for (int j=0; j<n; ++j)
        if (j != i && abs (a[j][i]) > EPS)
            for (int k=i+1; k<n; ++k)
                a[j][k] -= a[i][k] *
                    a[j][i];
}

cout << det;
```

10 KMP

```
// Make the preix table first

int main()
{
  // Find a pattern in a string
  int test;
  sc(test);
  while(test--)
  {
    string s,t;
    scr(s);scr(t);
    int n = s.size(), m = t.size();
    s = t+"$"+s;
    vector<int> p(n+m+1); // the pi
        array (dp array)
```

```
    // t(s,m);
    for(int i=1;i<n+m+1;i++)
    {
      int c = p[i-1];
      while(c>0 && s[c]!=s[i]) c = p[c-1];
      if(s[c]==s[i]) c++;
      p[i] = c;
    }

    // Indices of substrings
    vi v;
    for(int i=m+1;i<n+m+1;i++)
    {
      if(p[i]==m) v.pu(i-m-m+1);
    }

    if(v.size()==0) printf("Not
        Found\n");
    else
    {
      printf("%d\n",(int) v.size());
      for(int i=0;i<v.size();i++)
          printf("%d ",v[i]);printf("\n");
    }

    printf("\n");
  }
  return 0;
}
```

11 Kruskal

```
#include<bits/stdc++.h>
#define pu push_back
#define m make_pair
using namespace std;

// Function to compare by the Mth element
template<int M, template<typename> class
    F = std::less>
struct TupleCompare
{
    template<typename T>
    bool operator()(T const &t1, T const
        &t2)
    {
        return F<typename
            tuple_element<M,
            T>::type>()(std::get<M>(t1),
            std::get<M>(t2));
    }
};

void addEdge(vector < pair <int,int> >
    adj[],int u,int v,int key)
{
  adj[u].pu(m(v,key));
  adj[v].pu(m(u,key));
}

int main()
{
  int n;
  int q;
  cin>>n>>q;
  vector < pair <int,int> > adj[n];
  vector < tuple <int,int,int> > v;
  for(int h=0;h<q;h++)
  {
    int a,b,key;
    cin>>a>>b>>key;
    a-=1;b-=1;
    // addEdge(a,b,key,adj);
    v.pu(make_tuple(a,b,key));
  }
  sort(begin(v),end(v),
      TupleCompare<2>());
  int colour[n];
  int number[n];
  list <int> refer[n];
  list <int> :: iterator it;
  int countcolour = n;
  for(int i=0;i<n;i++)
  {
    colour[i] = i;
    number[i] = 1;
    refer[i].pu(i);
  }
  int count = 0;
  for(int i=0;i<v.size()&&
      countcolour;i++)
  {
    int a = get<0>(v[i]), b =
        get<1>(v[i]), c = get<2>(v[i]);
    int d = colour[a],e = colour[b];
    if(d==e) continue;
    else if(number[d]>number[e])
    {
      // for(int j=0;j<number[e];j++)
          colour[refer[e][j]] = d;
      for(it =
          refer[e].begin();it!=refer[e].end();it
          colour[*it] = d;
```

```
      number[d]+=number[e];
      number[e] = 0;
      count+=c;
      addEdge(adj,a,b,c);
      countcolour--;
      // cout<<c<<endl;
      refer[d].splice(refer[d].end(),refer[e]);
    }
    else
    {
      // for(int j=0;j<number[d];j++)
          colour[refer[d][j]] = e;
      for(it =
          refer[d].begin();it!=refer[d].end();it++)
          colour[*it] = e;
      number[e]+=number[d];
      number[d] = 0;
      count+=c;
      addEdge(adj,a,b,c);
      countcolour--;
      // cout<<c<<endl;
      refer[e].splice(refer[e].end(),
          refer[d]);
    }
  }
  for(int i=0;i<n;i++)
  {
    cout<<i<<": ";
    for(int j=0;j<adj[i].size();j++)
        cout<<adj[i][j].first<<" ";
    cout<<endl;
  }
  cout<<"Min Weight: "<<count<<endl;
  return 0;
}
```

12 LCA

```
struct LCA {
    vector<int> height, euler, first,
        segtree;
    vector<bool> visited;
    int n;

    LCA(vector<vector<int>> &adj, int
        root = 0) {
        n = adj.size();
        height.resize(n);
        first.resize(n);
        euler.reserve(n * 2);
        visited.assign(n, false);
        dfs(adj, root);
        int m = euler.size();
        segtree.resize(m * 4);
        build(1, 0, m - 1);
    }

    void dfs(vector<vector<int>> &adj,
        int node, int h = 0) {
        visited[node] = true;
        height[node] = h;
        first[node] = euler.size();
        euler.push_back(node);
        for (auto to : adj[node]) {
            if (!visited[to]) {
                dfs(adj, to, h + 1);
                euler.push_back(node);
            }
        }
    }

    void build(int node, int b, int e) {
        if (b == e) {
            segtree[node] = euler[b];
        } else {
            int mid = (b + e) / 2;
            build(node << 1, b, mid);
            build(node << 1 | 1, mid + 1,
                e);
            int l = segtree[node << 1], r
                = segtree[node << 1 | 1];
            segtree[node] = (height[l] <
                height[r]) ? l : r;
        }
    }

    int query(int node, int b, int e,
        int L, int R) {
        if (b > R || e < L)
            return -1;
        if (b >= L && e <= R)
            return segtree[node];
        int mid = (b + e) >> 1;

        int left = query(node << 1, b,
            mid, L, R);
        int right = query(node << 1 | 1,
            mid + 1, e, L, R);
        if (left == -1) return right;
        if (right == -1) return left;
        return height[left] <
            height[right] ? left : right;
    }

    int lca(int u, int v) {
```

```
        int left = first[u], right =
            first[v];
        if (left > right)
            swap(left, right);
        return query(1, 0, euler.size() -
            1, left, right);
    }
};
```

13 LIS

```
int bsearch(int k,int*memo,int n)
{
  int beg = 1,end = k-1,max1 = -1;
  while(beg<=end)
  {
    int mid = (beg+end)/2;
    if(memo[mid]<n) {max1 =
        max(max1,mid);beg = mid+1;}
    else if(memo[mid]>=n) {end = mid-1;}
  }
  return max1;
}

int main()
{
  int n;
  sc(n);
  int*l = new int[n];
  for(int i=0;i<n;i++) sc(l[i]);
  int*dp = new int[n];
  int*memo = new int[n+1];
  for(int i=0;i<n+1;i++) memo[i] = INF;
  // dp[0] = 1;
  if(n==1) printf("1\n");
  else
  {
    dp[0] = 1;
    memo[1] = l[0];
    for(int i=1;i<n;i++)
    {
      int c = bsearch(n,memo,l[i]);
      if(c==-1) {dp[i] = 1;memo[1] =
          min(l[i],memo[1]);}
      else {dp[i] = c+1;memo[c+1] =
          min(l[i],memo[c+1]);}
    }
    int max1 = 0;
    for(int i=0;i<n;i++) max1 =
        max(dp[i],max1);
    printf("%d\n",max1);
  }
  return 0;
}
```

14 Longest Palindrome

```
int rollinghash(int size,string s,int
    max) //This commputes if a palindrome
    of given size exists or not.
{
  int MOD = 4084061;
  int n = s.length();
  long long int hash=0,hashr = 0;
  long long int k = 27,prod=1;
  reverse(s.begin(),s.end());
  string t = s;
  reverse(s.begin(),s.end());
  int boo=-1;
  for(int i=0;i<size;i++)
  {
    hash*=k;
    hash+=(s[i]);hash%=MOD;
    hashr*=k;
    hashr+=(t[i]);hashr%=MOD;
    if(i<size-1)prod*=k;prod%=MOD;
  }
  int* l =
      (int*)calloc(n-size+1,sizeof(int));
  for(int i=0;i<n-size+1;i++)
  {
    l[i] = hashr;
    hashr-=t[i]*prod;
    hashr = ((hashr%MOD)+MOD)%MOD;
    hashr*=k;
    hashr%=MOD;
    if(i<n-size)hashr+=t[i+size];
    hashr%=MOD;
  }
  for(int i=0;i<n-size+1;i++)
  {
    if(hash==l[n-i-size]) {boo =
        i;break;}
    hash-=s[i]*prod;
    hash = ((hash%MOD)+MOD)%MOD;hash*=k;
    if(i<n-size)hash+=s[i+size];
    hash%=MOD;
  }
  if(boo!=-1) {max = size;return max;}
  return 0;
}
```

```
int bsearch(int c,string s)
{
  int max = 0;
  int left = 0,right = s.length();
  int mid = (left+right)/2;
  while(left<=right&&mid<right)
  {
    if(mid%2==c)mid+=1;
    int k = rollinghash(mid,s,0);
    if(k!=0) {if(k>max)max = k;left =
       mid+1;}
    else right = mid-1;
    mid = (left+right)/2;
  }
  return max;
}
int main()
{
  string s;
  cin>>s;
  int n = s.length();
  int left=0,right=n;
  int a = bsearch(1,s);
  int b = bsearch(0,s);
  cout<<a<<" "<<b<<endl;
  cout<<max(a,b)<<endl;
  return 0;
}
```

15 Mo Algorithm

```
int block;

typedef struct node
{
  int first,second,i;
}node;

bool sorter(const node &a, const node &b)
{
  if(a.fi/block < b.fi/block) return
     true;
  else if(a.fi/block> b.fi/block) return
     false;
  else
  {
    if(a.se<b.se) return true;
    return false;
  }
}

int add(int*counter,int*l,int pos)
{
  int count = 0;
  counter[l[pos]]++;
  if(counter[l[pos]]==1) count++;
  return count;
}

int remove(int*counter,int*l,int pos)
{
  int count =0 ;
  counter[l[pos]]--;
  if(counter[l[pos]]==0) count--;
  return count;
}

int main()
{
  //
     ios_base::sync_with_stdio(0);cin.tie(0);
  int n;
  sc(n);
  int*l = new int[n];
  int*counter =
     (int*)calloc(1000001,sizeof(int));
  for(int i=0;i<n;i++) sc(l[i]);
  int q;
  sc(q);
  int ans[q];
  node *arr = new node[q];
  for(int i=0;i<q;i++)
  {
    sc(arr[i].fi);
    sc(arr[i].se);
    arr[i].fi--;arr[i].se--;
    arr[i].i = i;
  }

  block = int(sqrt(n));
  sort(arr,arr+q,sorter);

  int left = -1,right = -1,count = 0;
  for(int i=0;i<q;i++)
  {
    node p = arr[i];
    int a = p.fi,b = p.se;
    while(left>a)
    {
      left--;
      count+=add(counter,l,left);
    }
```

```
    while(left<a)
    {
      if(left==-1) {left++;continue;}
      count+=remove(counter,l,left);
      left++;
    }
    while(right<b)
    {
      right++;
      count+=add(counter,l,right);
    }
    while(right>b)
    {
      count+=remove(counter,l,right);
      right--;
    }
    ans[p.i] = count;
  }

  for(int i=0;i<q;i++)
  {
    printf("%d\n",ans[i]);
  }
  return 0;
}
```

16 ModularExponentiation

```
ll power(ll x, ll y, ll p)
{
    ll res = 1;     // Initialize result
    x = x % p; // Update x if it is more
        than or
    while (y > 0)
    {
        if (y & 1)
            res = (res*x) % p;
        y = y>>1; // y = y/2
        x = (x*x) % p;
    }
    return res;
}

void multiply(ll**l,ll**m,ll**arr,int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            for(int k=0;k<n;k++)
            {
                arr[i][j]+=(l[i][k]*m[k][j]);
            }
        }
    }
}

void zero(ll **l, int n)
{
    for(int i=0;i<n;i++) for(int
        j=0;j<n;j++) l[i][j] = 0;
}

void copy(ll**arr,ll**m,int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            m[i][j] = arr[i][j];
        }
    }
    zero(arr,n);
}

void pow(ll**l,int n,ll y)
{
    ll**m = new ll*[n];
    for(int i=0;i<n;i++) m[i] = new
        ll[n];
    for(int i=0;i<n;i++) for(int
        j=0;j<n;j++) m[i][j] = l[i][j];
    ll**arr = new ll*[n];
    for(int i=0;i<n;i++) arr[i] = new
        ll[n];
    zero(arr,n);
    while(y)
    {
        if(y&1)
        {
            multiply(l,m,arr,n);
            copy(arr,l,n);
        }

        multiply(m,m,arr,n);
        copy(arr,m,n);
        y>>=1;
    }
}

// pow (l,n,exp)
```

17 Nextsmallest

```
#include<bits/stdc++.h>
#define MAX 1000000001
using namespace std;

void nextsmallest(int l[],int m[],int n)
{
  map<int,int> d;
  for(int i=0;i<n;i++) d[l[i]] = i;
  stack<int> s;
  for(int i=0;i<n;i++)
  {
    if(!s.empty() && l[i]<s.top())
    {
      while(!s.empty() && l[i]<s.top())
      {
        m[d[s.top()]] = i;
        s.pop();
      }
      s.push(l[i]);
    }
    else s.push(l[i]);
  }
}

int main()
{
  int n; cin>>n; int l[n];
  for(int i=0;i<n;i++) cin>>l[i];
  int m[n];
  for(int i=0;i<n;i++) m[i] = MAX;
  nextsmallest(l,m,n); // in m
  return 0;
}
```

18 PaliwalSegementTree

```
const int N = 1 << 17;

struct node{
        int cnt;
        void assign(int value){
                cnt = value;
        }
        void update(int value){
                cnt += value;
        }
        void combine(node &left, node
            &right){
                cnt = left.cnt + right.cnt;
        }
};

int n, a[N], lazy[N];
node tree[2*N];

// [l, r)
void build(int id = 1, int l = 0, int r
    = n)
{
        if(l+1 == r){
                tree[id].assign(a[l]);
                return;
        }
        int left = id<<1, right = left+1,
            mid = (l+r)>>1;
        build(left, l, mid); build(right,
            mid, r);

        tree[id].combine(tree[left],
            tree[right]);
        return;
}

// point update -> update(index, value);
void update(int index, int val, int id =
    1, int l = 0, int r = n)
{
        if(l+1 == r){
                tree[id].assign(val);
                return;
        }
        int left = id<<1, right = left+1,
            mid = (l+r)>>1;

        if(index < mid) update(index,
            val, left, l, mid);
        else update(index, val, right,
            mid, r);

        tree[id].combine(tree[left],
            tree[right]);
}


// range update and utility functions
void upd(int id,int l,int r,int x)
{ //   update the current node and its
    index in the lazy array
        lazy[id] += x;
```

```
        tree[id].update((r - l) * x);
}

void shift(int id,int l,int r)
{ //propogate update information to the
    children
        if(lazy[id] and l+1 < r){
                int mid = (l+r)/2;
                upd(id * 2, l, mid,
                    lazy[id]);
                upd(id * 2 + 1, mid, r,
                    lazy[id]);
                lazy[id] = 0; // passing
                    is done, reset the
                    index in the lazy array
        }
}

// range update -> update(x, y, val);
void update(int x, int y, int val, int
    id = 1, int l = 0, int r = n)
{
        if(x >= r or l >= y) return;
        if(x <= l && r <= y){
                upd(id, l, r, val);
                return;
        }

        shift(id, l, r); // pass the
            updates to the children

        int left = id<<1, right = left+1,
            mid = (l+r)>>1;

        update(x, y, val, left, l, mid);
        update(x, y, val, right, mid, r);

        tree[id].combine(tree[left],
            tree[right]);
        return;
}

// range query -> query(x, y);
// for point query, traverse like in
    point update
int query(int x, int y, int id = 1, int
    l = 0, int r = n)
{
        if(x >= r or l >= y) return 0;
        if(x <= l && r <= y) return
            tree[id].cnt;

        shift(id, l, r);      //use this
            with lazy propogation

        int left = id<<1, right = left+1,
            mid = (l+r)>>1;

        return query(x, y, left, l, mid)
            + query(x, y, right, mid, r);
}

int main()
{
        return 0;
}
```

19 Permutations

```
void recur(set<int> s, vector<int> v,int
    n)
{
  if(s.empty())
  {
    for(int i=0;i<v.size();i++)
        cout<<v[i]<<" ";
    cout<<endl;
    return;
  }

  for(int i=1;i<=n;i++)
  {
    if(s.find(i)!=s.end())
    {
      v.pu(i);
      s.erase(i);
      recur(s,v,n);
      v.pop_back();
      s.insert(i);
    }
  }
}
// recur(set,vector,n)
```

20 Push-ReLabel

```
const int inf = 1000000000;

int n;
vector<vector<int>> capacity, flow;
vector<int> height, excess;
```

```
void push(int u, int v)
{
    int d = min(excess[u],
        capacity[u][v] - flow[u][v]);
    flow[u][v] += d;
    flow[v][u] -= d;
    excess[u] -= d;
    excess[v] += d;
}

void relabel(int u)
{
    int d = inf;
    for (int i = 0; i < n; i++) {
        if (capacity[u][i] - flow[u][i] >
            0)
            d = min(d, height[i]);
    }
    if (d < inf)
        height[u] = d + 1;
}

vector<int> find_max_height_vertices(int
    s, int t) {
    vector<int> max_height;
    for (int i = 0; i < n; i++) {
        if (i != s && i != t && excess[i]
            > 0) {
            if (!max_height.empty() &&
                height[i] >
                height[max_height[0]])
                max_height.clear();
            if (max_height.empty() ||
                height[i] ==
                height[max_height[0]])
                max_height.push_back(i);
        }
    }
    return max_height;
}

int max_flow(int s, int t)
{
    height.assign(n, 0);
    height[s] = n;
    flow.assign(n, vector<int>(n, 0));
    excess.assign(n, 0);
    excess[s] = inf;
    for (int i = 0; i < n; i++) {
        if (i != s)
            push(s, i);
    }

    vector<int> current;
    while (!(current =
        find_max_height_vertices(s,
        t)).empty()) {
        for (int i : current) {
            bool pushed = false;
            for (int j = 0; j < n &&
                excess[i]; j++) {
                if (capacity[i][j] -
                    flow[i][j] > 0 &&
                    height[i] == height[j]
                    + 1) {
                    push(i, j);
                    pushed = true;
                }
            }
            if (!pushed) {
                relabel(i);
                break;
            }
        }
    }

    int max_flow = 0;
    for (int i = 0; i < n; i++)
        max_flow += flow[0][i];
    return max_flow;
}
```

21 Snippets

```
// If you're using unordered set and
    need to hash anything. Here is the
    example for a pair
// Only for pairs of std::hash-able
    types for simplicity.
// You can of course template this
    struct to allow other hash functions
struct pair_hash {
    template <class T1, class T2>
    std::size_t operator () (const
        std::pair<T1,T2> &p) const {
        auto h1 =
            std::hash<T1>{}(p.first);
        auto h2 =
            std::hash<T2>{}(p.second);

        // Mainly for demonstration
            purposes, i.e. works but is
```

```
            overly simple
        // In the real world, use sth.
            like boost.hash_combine
        return h1 ^ h2;
    }
};

// If there's data you wanna frequetly
    wanna access, then use a vector! If
    you don't care about the order, use
    an unordered_map
// before you use a normal map. Maps can
    give you timeouts if you aren't
    careflul


// Let's say you want to compute where
    x> p  . Now by division algorithm
    x=  q  *  ( p - 1 )  + r
// it gets reduced to a^r mod p where r
    is x%(p-1)
```

22 Stress

22.1 check

```
file1 = "A.out"
file2 = "B.out"

s = open(file1, 'r').read().split()
t = open(file2, 'r').read().split()

print s == t
```

22.2 gen

```
from random import *

''' Generate a random array of integers
    with elements in the range [L, R] '''
def genRandomArray(N, L, R):
        a = [randrange(L,R+1) for _ in
            xrange(N)]
        return a

''' Generate a random string from
    characters in the range [A, B]'''
def genRandomString(N, A, B):
        l = genRandomArray(N, ord(A),
            ord(B))
        s = ''
        for char in l: s += chr(char)
        return s

''' Generate a random permutation of [1,
    2 ... N] '''
def genRandomPermutation(N):
        permutation = range(1, N+1)
        shuffle(permutation)
        return permutation

''' Generate a random unweighted tree'''
def genRandomTree(N):
        edges = []
        for u in xrange(2,N+1):
                v = randrange(1,u)
                edges.append([u,v])
        permutation =
            genRandomPermutation(N)

        for i in xrange(0,N-1):
                u, v = edges[i]
                u = permutation[u-1]
                v = permutation[v-1]
                edges[i] = (u,v)
        return edges

''' Generate a random weighted tree '''
def genRandomWeightedTree(N, L, R):
        weigths = genRandomArray(N-1, L,
            R)
        tree = genRandomTree(N)
        wtree = []

        for i in xrange(0,N-1):
                u, v, w = tree[i][0],
                    tree[i][1], weigths[i]
                wtree.append((u, v, w))

        return wtree

''' Undirected, no multiedges and no
    self-loops '''
def genRandomGraph(N, E):
        edges = {}

        if N == 1: return []

        for i in xrange(E):
                u = randrange(1,N+1)
                v = u
```

```
            while v == u: v =
                randrange(1,N+1)

            while (u,v) in edges or
                (v,u) in edges:
                    u =
                        randrange(1,N+1)
                    v = u
                    while v == u: v =
                        randrange(1,N+1)

            edges[(u,v)] = 1

    ret = []
    for edge in edges:
        ret.append(edge)

    return ret

''' Undirected, no multiedges, no
    self-loops, connected '''
def genRandomConnectedGraph(N, E):
    E -= N-1
    tree = genRandomTree(N)
    edges = {}
    for edge in tree:
            edges[edge] = 1

    for i in xrange(E):
            u = randrange(1,N+1)
            v = u
            while v == u: v =
                randrange(1,N+1)
            while (u,v) in edges or
                (v,u) in edges:
                    u =
                        randrange(1,N+1)
                    v = u
                    while v == u: v =
                        randrange(1,N+1)

            edges[(u,v)] = 1

    ret = []
    for edge in edges:
        ret.append(edge)

    return ret

''' Undirected, no multiedges, no
    self-loops, can be forced to be
    connected '''
def genRandomWeightedGraph(N, E, L, R,
    connected = False):
    graph = []
    if not connected:
            graph = genRandomGraph(N,
                E)
    else:
            graph =
                genRandomConnectedGraph(N,
                E)

    weights = genRandomArray(E, L, R)

    wgraph = []
    for i in xrange(E):
            u, v, w = graph[i][0],
                graph[i][1], weights[i]
            wgraph.append((u,v,w))
    return wgraph

if __name__ == '__main__':
    n = randint(6,10)
    l = []
    for i in range(n): l.append(i+1)
    shuffle(l)
    print n
    for i in range(n):
            print
                randint(1,3),l[i],randint(1,1
```

22.3 test

```
testcase=0
fail="False"

while [ $testcase -lt 10000 ]
do
    python gen.py > testcase.in
    ./A < testcase.in > A.out
    ./brute < testcase.in > B.out
    result=$(python check.py)

    if [ "$result" == "$fail" ]
    then
        echo "Wrong Answer!"
        break
    fi
    testcase=‘expr $testcase + 1‘
    echo $testcase, $result
```

```
done
```

23 bash

23.1 make

```
#!/bin/bash

alias f="g++ -std=c++14 $1.cpp"
if [ ! -e "$1.cpp" ]; then
  cp ~/IIITB/Questions/template.cpp .
  mv template.cpp $1.cpp
else
  echo "File exists"
fi
```

23.2 run

```
#!/bin/bash
shopt -s expand_aliases
# Script to compile and execute a c
    program in one step.

# Get file name without the .c extension
file_name="$(echo $1|sed
    's/\(.*\)\.cpp/\1/')"

# Compile the program with -o option to
    specify the name of the binary
g++ -std=c++14 "$1"
alias f="g++ -std=c++14 $1 "
alias .="./a.out"
# If there were no compilation errors,
    run the program
#if [[ $? -eq 0 ]]; then
#       ./$file_name.out
#fi
```

24 gen

```
from random import *

''' Generate a random array of integers
    with elements in the range [L, R] '''
def genRandomArray(N, L, R):
        a = [randrange(L,R+1) for _ in
            xrange(N)]
        return a

''' Generate a random string from
    characters in the range [A, B]'''
def genRandomString(N, A, B):
        l = genRandomArray(N, ord(A),
            ord(B))
        s = ''
        for char in l: s += chr(char)
        return s

''' Generate a random permutation of [1,
    2 ... N] '''
def genRandomPermutation(N):
        permutation = range(1, N+1)
        shuffle(permutation)
        return permutation

''' Generate a random unweighted tree'''
def genRandomTree(N):
        edges = []
        for u in xrange(2,N+1):
                v = randrange(1,u)
                edges.append([u,v])

        permutation =
            genRandomPermutation(N)

        for i in xrange(0,N-1):
                u, v = edges[i]
                u = permutation[u-1]
                v = permutation[v-1]
                edges[i] = (u,v)
        return edges

''' Generate a random weighted tree '''
def genRandomWeightedTree(N, L, R):
        weigths = genRandomArray(N-1, L,
            R)
        tree = genRandomTree(N)
        wtree = []

        for i in xrange(0,N-1):
                u, v, w = tree[i][0],
                    tree[i][1], weigths[i]
                wtree.append((u, v, w))

        return wtree

''' Undirected, no multiedges and no
    self-loops '''
def genRandomGraph(N, E):
```

```
        edges = {}

        if N == 1: return []

        for i in xrange(E):
                u = randrange(1,N+1)
                v = u
                while v == u: v =
                    randrange(1,N+1)

                while (u,v) in edges or
                    (v,u) in edges:
                        u =
                            randrange(1,N+1)
                        v = u
                        while v == u: v =
                            randrange(1,N+1)

                edges[(u,v)] = 1

        ret = []
        for edge in edges:
            ret.append(edge)

        return ret

''' Undirected, no multiedges, no
    self-loops, connected '''
def genRandomConnectedGraph(N, E):
        E -= N-1
        tree = genRandomTree(N)
        edges = {}
        for edge in tree:
                edges[edge] = 1
        for i in xrange(E):
                u = randrange(1,N+1)
                v = u
                while v == u: v =
                    randrange(1,N+1)

                while (u,v) in edges or
                    (v,u) in edges:
                        u =
                            randrange(1,N+1)
                        v = u
                        while v == u: v =
                            randrange(1,N+1)

                edges[(u,v)] = 1

        ret = []
        for edge in edges:
            ret.append(edge)

        return ret

''' Undirected, no multiedges, no
    self-loops, can be forced to be
    connected '''
def genRandomWeightedGraph(N, E, L, R,
    connected = False):
        graph = []
        if not connected:
                graph = genRandomGraph(N,
                    E)
        else:
                graph =
                    genRandomConnectedGraph(N,
                    E)
        weights = genRandomArray(E, L, R)

        wgraph = []
        for i in xrange(E):
                u, v, w = graph[i][0],
                    graph[i][1], weights[i]
                wgraph.append((u,v,w))
        return wgraph

if __name__ == '__main__':
```

25 inversion

```
void sort(int* ,int,int);
void merge(int* ,int,int,int);
long long int count = 0;

// sort(array, beg,end);
void sort(int *l,int i,int j)
{
  if(i<j)
  {
    int c = (i+j)/2;
    sort(l,i,c);
    sort(l,c+1,j);
    merge(l,i,c,j);
  }
}

void merge(int*l,int i,int c,int j)
{
```

```
int*m =
    (int*)calloc((j-i+1),sizeof(int));
int x=i,y=c+1;
int k=0;
while(k<j-i+1)
{
  if(y==j+1)
  {
    while(x<=c)
    {
      m[k] = l[x];
      count+=(y-c-1);
      x++;
      k++;
    }
    continue;
  }
  else if(x==c+1)
  {
    while(y<=j)
    {
      m[k] = l[y];
      y++;
      k++;
    }
    continue;
  }
  else if(l[x]<l[y])
  {
    m[k] = l[x];
    count+=(y-c-1);
    x++;
  }
  else if(l[y]<=l[x])
  {
      m[k] = l[y];
      y++;
    }
    k++;
  }
  //Copying
  for(int k=0;k<j-i+1;k++)
  {
    l[k+i] = m[k];
  }
}
```

26 nCr

```
long long int nCr(int n,int k) //Not my
    code
{
    long long int ans=1;
    k=k>n-k?n-k:k;
    int j=1;
    for(;j<=k;j++,n--)
    {
        if(n%j==0)
        {
            ans*=n/j;
        }else
        if(ans%j==0)
        {
            ans=ans/j*n;
        }else
        {
            ans=(ans*n)/j;
        }
    }
    return ans;
}

// Less constraints, but higher usage of
    repeated result=\
];

int dp[max][max];
//Initialise array elements with zero
int nCr(int n, int r)
{
       if(n==r) return dp[n][r] = 1;
           //Base Case
       if(r==0) return dp[n][r] = 1;
           //Base Case
       if(r==1) return dp[n][r] = n;
       if(dp[n][r]) return dp[n][r]; //
           Using Subproblem Result
       return dp[n][r] = nCr(n-1,r) +
           nCr(n-1,r-1);
}
```

27 nCrMODCOMOP

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
#define sl(a) scanf("%lld",&a);
#define rep(n) for(int i=0;i<n;i++)
#define prarr(a,n) rep(n) cout<<a[i]<<"
    ";cout<<endl;
#define vi vector<int>
```

```
#define ff first
#define ss second
#define N 100010

bitset<10000010> bs;
vi primes;
ll _sieve_size;
map<int,int> powers={};//stores powers
    of primes in primefac of mod
map<ll,ll>
    tot={{1000000007ll,1000000006ll}};//
    stores toitent value
ll fac[N];//factorial without prime
    factors of mod
ll ifac[N];//invfac without prime
    factors of mod
ll power(ll x, ll y, ll p)
{
ll res = 1;
// Initialize result
x = x % p; // Update x if it is more
while (y > 0)
{
if (y & 1)
res = (res*x) % p;
y = y>>1; // y = y/2
x = (x*x) % p;
}
return res;
}
void sieve(ll n)
{
  _sieve_size=n+1;
  bs.reset();bs.flip();
  bs.set(0,false);bs.set(1,false);
  for(ll i=2;i<_sieve_size;i++)
  {
    if(bs.test(i))
    {
      for(ll j=i*i;j<_sieve_size;j+=i)
      bs.set(j,false);
    primes.push_back((int)i);
    }
  }
}
vi prfac(int n)
{
  vi ret={};
  if(n<0) ret.push_back(-1);
  n=abs(n);
  int l=0;
while(n&& n%primes[l]==0)
    {ret.push_back(primes[l]);n/=2;}
  l++;
  for(ll j=primes[l];j<=sqrt(n);)
  {
    while(n&& n%j==0)
        {ret.push_back(j);n/=j;}
    l++;
    j=primes[l];
  }
  if(n!=1)
  ret.push_back(n);
return ret;
}
ll toitent(ll m)
{
        if(tot.find(m)!=tot.end()) return
            tot[m];
        if(primes.size()==0)
            sieve(sqrt(m)+2);
        vi fac = prfac(m);
        ll tmp = m;
        for(auto i : fac)
        {
                int cnt = 0;
                while(m%i==0)
                    {m=m/i;cnt++;}
                powers[i] = cnt;
        }
        ll phi = 1;
        for(auto p : powers)
            phi*=power(p.ff,p.ss-1,(ll)(1e9)+7ll
        tot[tmp] = phi;
        return phi;
}
ll modinv(ll a,ll mod)
{
        return
            power(a,toitent(mod)-1,mod);
}
void setfacnp(int n,ll mod)
{
        if(tot.find(mod)==tot.end())
            toitent(mod);
        int cnt=1;
        for(int i = 1;i<=n;i++,cnt++)
        {
                for(auto p : powers)
                    while(i%p.ff==0)
                    i/=p.ff;
                fac[cnt] = i;
                ifac[cnt] = modinv(i,mod);
                i = cnt;
```

```
        }
        fac[0] = 1;
        ifac[0] = 1;
        for(int i = 1;i<=n;i++)
        {
                fac[i] =
                    (fac[i-1]*fac[i])%mod;
                ifac[i] =
                    (ifac[i-1]*ifac[i])%mod;
        }


}
ll C(int n,int r,ll mod)
{
        setfacnp(n,mod);
        ll ret = 1;
        ret = (fac[n]*ifac[n-r])%mod;
        ret=(ret*ifac[r])%mod;
        map<int,int> m={}; vi v =
            {n,n-r,r};
        int i = n,cnt = 0,pi;;
        //now take the prime factors of
            mod and find their exponents
            in nCr and store in ans
        for(auto p : powers)
            {cnt=0;pi=p.ff; while(i/pi)
            {cnt+=i/pi;pi*=p.ff;}m[p.ff]+=cnt;}
        i = n-r;
        for(auto p : powers)
            {cnt=0;pi=p.ff; while(i/pi)
            {cnt+=i/pi;pi*=p.ff;}m[p.ff]-=cnt;}
        i = r;
        for(auto p : powers)
            {cnt=0;pi=p.ff; while(i/pi)
            {cnt+=i/pi;pi*=p.ff;}m[p.ff]-=cnt;}
        ll ans = 1;
        for(auto p :m){
                ans =
                    ans*(power(p.ff,p.ss,mod));ans%=mod;
        }
        ret*=ans;
        ret%=mod;
        return ret;
}
int main(int argc, char const *argv[])
{


        int t,n =15,r = 4,mod =
            1800000092;
        cout<<C(n,r,mod)<<endl;
        return 0;
        cin>>t;
        while(t--)
        {
                // cin>>n>>r>>mod;
        }
        return 0;
}
```

28 rabinkarp

```
#include<stdio.h>
#include<string.h>
int bruteforce(char*,char*,int,int);
void rabinkarp(char*,char*,int,int);

int main()
{
  int n,m;
  scanf("%d",&n);
  char s[n];
  scanf("%s",s);
  scanf("%d",&m);
  char t[m];
  scanf("%s",t);
  printf("%d\n",bruteforce(s,t,n,m));
  //rabinkarp(s,t,n,m);
  return 0;
}
int bruteforce(char* s,char* t,int n,int
    m)
{
  int count = 0,j=0;
  for(j=0;j<n-m+1;j++)
  {
    count = 0;
    for(int i=0;i<m;i++)
    {
      if (s[i+j]==t[i]) count++;
      else break;
    }
    if(count==m)return 1;
  }
  return 0;
}

void rabinkarp(char* s,char*t,int n,int
    m)
{
  int* hash[1000000];
  int d = 26;
  int hasht=0,hashs=0;
  int p=1001;
```

```
  int prod=1;
  for(int i=0;i<m;i++)
  {
    hashs+=(s[i]-'0')*prod;
    hasht+=(t[i]-'0')*prod;
    prod*=d;hashs%=p;hasht%=p;prod%=p;
  }
}

typedef struct
{
  char* data;
  s1 *next;
} s1;
```

29 template

```
#include<bits/stdc++.h>
#define mt make_tuple
#define mp make_pair
#define pu push_back
#define INF 1000000001
#define MOD 1000000007
#define N 1000010
#define ll long long int
#define ld long double
#define vi vector<int>
#define vll vector<long long int>
#define fi first
#define se second
#define sc(n) scanf("%d",&n);
#define scll(n) scanf("%lld",&n);
#define scld(n) scanf("%Lf",&n);
#define scr(s) {char
    temp[1000000];scanf("%s",temp);s =
    temp;}
#define pr(v) { for(int
    i=0;i<v.size();i++) { v[i]==INF?
    cout<<"INF " : cout<<v[i]<<" "; }
    cout<<endl;}
#define rep(n) for(int i=0;i<n;i++)
#define t1(x)              cerr<<#x<<" :
    "<<x<<endl
#define t2(x, y)           cerr<<#x<<" :
    "<<x<<" "<<#y<<" : "<<y<<endl
#define t3(x, y, z)        cerr<<#x<<" :"
    <<x<<" "<<#y<<" : "<<y<<" "<<#z<<" :
    "<<z<<endl
#define t4(a,b,c,d)        cerr<<#a<<" :
    "<<a<<" "<<#b<<" : "<<b<<" "<<#c<<" :
    "<<c<<" "<<#d<<" : "<<d<<endl
#define t5(a,b,c,d,e)        cerr<<#a<<"
    : "<<a<<" "<<#b<<" : "<<b<<" "<<#c<<"
    : "<<c<<" "<<#d<<" : "<<d<<" "<<#e<<"
    : "<<e<<endl
#define
    GET_MACRO(_1,_2,_3,_4,_5,NAME,...)
    NAME
#define t(...) GET_MACRO(__VA_ARGS__,t5,
    t4, t3, t2, t1)(__VA_ARGS__)
#define _ cout<<"here"<<endl;
#define __ {ios::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);}


using namespace std;

int main()
{

  return 0;
}
```