Team Notebook

March 13, 2020

Contents

1 Algorithm

1.1 Binary Search

```java
public int upperBound(int[] arr,int length,int value){
int low=0;
int high=length+1;
while (low<high){
    int mid=(low+high)/2;
    if(value>=arr[mid])
        low=mid+1;
    else
        high=mid;
}
return low;
}
public int lowerBound(int[] arr,int length,int value){
int low=0;
int high=length;
while (low<high){
    int mid=(low+high)/2;
    if(value<=arr[mid])
        high=mid;
    else
        low=mid+1;
}
return low;
}
```

1.2 Centroid Decomposition

```java
int getCentroid(int src, bool visited[], int subtree_size[],
    int n){
/* assume the current node to be centroid */
bool is_centroid = true;
/* mark it as visited */
visited[src] = true;
/* track heaviest child of node, to use in case node is not
    centroid */
int heaviest_child = 0;
vector<int>::iterator it;
/* iterate over all adjacent nodes which are children
(not visited) and not marked as centroid to some
subtree */
for (it = tree[src].begin(); it!=tree[src].end(); it++)
  if (!visited[*it] && !centroidMarked[*it]){
  /* If any adjacent node has more than n/2 nodes,
   * current node cannot be centroid */
```

```java
  if (subtree_size[*it]>n/2)
    is_centroid=false;

/* update heaviest child */
  if (heaviest_child==0 ||
  subtree_size[*it]>subtree_size[heaviest_child])
  heaviest_child = *it;
}
/* if current node is a centroid */
if (is_centroid && n-subtree_size[src]<=n/2)
return src;
/* else recur on heaviest child */
return getCentroid(heaviest_child, visited, subtree_size, n)
    ;
}
```

1.3 DSU on tree

```cpp
ll cnt[maxn];
void dfs(ll v, ll p, bool keep){
  ll 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); // bigChild marked as big and not
        cleared from cnt
  for(auto u : g[v])
    if(u != p && u != bigChild)
      for(int p = st[u]; p < ft[u]; p++)
        cnt[ col[ ver[p] ] ]++;
        cnt[ col[v] ]++;
//now cnt[c] is the number of vertices in subtree of
    vertex v that has color c. You can answer the queries
    easily.
  if(keep == 0)
    for(int p = st[v]; p < ft[v]; p++)
      cnt[ col[ ver[p] ] ]--;
}
```

1.4 DSU

```cpp
class DSU{
```

```java
private int[] par;
private int[] size;
public Dsu(int len){
  par=new int[len+1];
  size=new int[len+1];
  for(int i=0;i<len;++i){
    par[i]=i;
    size[i]=1;
  }
}
public int getParent(int u){
  if(par[u]==u)
    return u;
    return par[u]=getParent(par[u]);
}
public void merge(int u,int v){
  u=getParent(u);
  v=getParent(v);
  if(u!=v){
    if(size[u]<size[v]){
      int t=u;u=v;v=t;
    }
    par[v]=u;
    size[u]+=size[v];
  }
}
}
```

1.5 FFT and polynomial multiply

```cpp
typedef complex<double> base;
# define PI        3.14159265358979323846264338327950288 4L
    /* pi */
void fft (vector<base> & a, bool invert) {
 ll n = (ll) a.size(),i,j,len;
 for (i=1, j=0; i<n; ++i) {
  ll bit = n >> 1;
  for (; j>=bit; bit>>=1)
   j -= bit;
  j += bit;
  if (i < j)
   swap (a[i], a[j]);
 }
 for (len=2; len<=n; len<<=1) {
  double ang = 2*PI/len * (invert ? -1 : 1);
  base wlen (cos(ang), sin(ang));
  for (i=0; i<n; i+=len) {
   base w (1);
   for (j=0; j<len/2; ++j) {
```

```cpp
      base u = a[i+j], v = a[i+j+len/2] * w;
      a[i+j] = u + v;
      a[i+j+len/2] = u - v;
      w *= wlen;
    }
  }
}
  if (invert)
   for (int i=0; i<n; ++i)
     a[i] /= n;
}
void multiply (vector<ll> & a,vector<ll> & b,vector<ll> &
      res) {
 vector<base> fa (a.begin(), a.end()), fb (b.begin(), b.end
      ());
 ll n = 1;
 while (n < max (a.size(), b.size())) n <<= 1;
 n <<= 1;
 fa.resize (n), fb.resize (n);
 fft (fa, false), fft (fb, false);
 for (ll i=0; i<n; ++i)
  fa[i] *= fb[i];
 fft (fa, true);
 res.resize (n);
 for (size_t i=0; i<n; ++i)
  res[i] = (ll) (fa[i].real() + 0.5);
}
```

1.6 LIS

```cpp
void LIS(vector<ll>&ar, ll n){
 vector<ll>dp(n+1,mod);
 dp[0]=-mod;
 rep(i,0,n){
  ll j=upper_bound(all(dp),ar[i])-dp.begin();
  if(dp[j-1]<ar[i]&&ar[i]<dp[j]){
   dp[j]=ar[i];}}
 ll x=0;
 rep(i,0,n+1){
  if(dp[i]!=mod){
   x=i;}
 }
 cout<<x;
}
```

1.7 MOs algorithm

```java
public class MOs {
  long ans=0L;
  long[] freq=new long[1000001];
  public void solve() {
    int n=in.ni(),q=in.ni();
    int[] arr=in.intarr(n);
    ArrayList<Query> list=new ArrayList<>();
    for(int i=0;i<q;++i)list.add(new Query(in.ni()-1,in.ni()
        -1,i));
    int sq=(int)Math.sqrt(n);
    list.sort((q1,q2)->{
      if(q1.l/sq==q2.l/sq)
        return q1.r/sq-q2.r/sq;
      return q1.l/sq-q2.l/sq;
    });
    long[] ans_arr=new long[q];
    Arrays.fill(ans_arr,-1);
    int l=0,r=-1;
    for(int i=0;i<q;++i){
      Query query=list.get(i);
      while (l < query.l) remove(arr[l++]);
      while (l > query.l) add(arr[--l]);
      while (r < query.r) add(arr[++r]);
      while (r > query.r) remove(arr[r--]);
      ans_arr[query.idx]=ans;
    }
    for(int i=0;i<q;++i)
      if(ans_arr[i]!=-1)
        out.println(ans_arr[i]);
  }
  void add(int num){
    ans += (long) (num ) * (((freq[num]++)<<1L) + 1L);
  }
  void remove(int num){
    ans -= (long) (num) * (((freq[num]--)<<1L) - 1L);
  }
  class Query{
    int l,r,idx;long ans;
    public Query(int l, int r, int idx) {
      this.l = l;
      this.r = r;
      this.idx = idx;
    }
  }
}
```

1.8 Query On Tree

```java
// subtree nodes having 0
int[] BIT = new int[MAX + 1];
int[] start = new int[MAX + 1];
int timer = 0;
int[] idx = new int[MAX + 1];
int[] end = new int[MAX + 1];
public void solve() {
    int n,q;
    int[] arr = new int[n];
    lists = new ArrayList[n];
    dfs(0, -1);
    for (int timer = 1; timer <= n; ++timer) {
      if (arr[idx[timer]] == 0) {
        update(timer, 1);
      }
    }
    while (q-- > 0) {
      char type = in.nc();
      if (type == 'U') {
        int x = in.ni(), y = in.ni();
        x--; int earlier = arr[x];
        arr[x] = y;
        if (arr[x] == 0 && earlier != 0)
          update(start[x], 1);
        else if (earlier == 0 && arr[x] != 0)
          update(start[x], -1);
      }else{
        int x = in.ni();x--;
        int ans=(query(end[x]) - query(start[x] - 1));
      }
    }
}
private void dfs(int s, int p) {
  start[s] = ++timer;
  idx[timer] = s;
  for (int i : lists[s])
    if (i != p)
      dfs(i, s);
  end[s] = timer;
}
```

1.9 Sqrt Decomposition

```java
class SqrtDecomposition {
  public void solve() {
    int n=in.ni();
    int[] arr=in.intarr(n);
    int BUCKET_SIZE=(int)Math.sqrt(n*1.0)+1;
    int[][] preArr=new int[BUCKET_SIZE][BUCKET_SIZE];
```

```
  for(int i=0;i<n;++i){
    preArr[i/BUCKET_SIZE][i%BUCKET_SIZE]=arr[i];
  }
  for(int i=0;i<BUCKET_SIZE;++i)
    sort(preArr[i]);
  int q=in.ni();
  while (q-->0){
    int t=in.ni();
    int ans=0;
    if(t==0){
      int l=in.ni()-1,r=in.ni()-1,c=in.ni();
      int BL=l/BUCKET_SIZE,BR=r/BUCKET_SIZE;
      if(BL==BR){
        for(int i=l;i<=r;++i)
          if(arr[i]>=c) ++ans;
      }else{
        int end=(BL+1)*BUCKET_SIZE-1;
        for(int i=l;i<=Math.min(end,n-1);++i){
          if(arr[i]>=c) ++ans;
        }
        for(int i=BL+1;i<BR;++i){
          ans+=lowerBound(preArr[i],0,BUCKET_SIZE,c);
        }
        for(int i=BR*BUCKET_SIZE;i<=r;++i){
          if(arr[i]>=c) ++ans;
        }
      }
      out.println(ans);
    }else {
      int idx=in.ni()-1,val=in.ni();
      int B=idx/BUCKET_SIZE;
      for(int i=0;i<BUCKET_SIZE;++i)
        if(preArr[B][i]==arr[idx]){
          preArr[B][i]=val;
          break;
        }
      sort(preArr[B]);
      arr[idx]=val;
    }
  }
}
```

1.10   Sum of subset

```
//memory optimized, super easy to code.
for(int i = 0; i<(1<<N); ++i)
 F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<N);
    ++mask){
 if(mask & (1<<i))
  F[mask] += F[mask^(1<<i)];
}
// The above algorithm runs in O(N2N) time.
```

1.11   Ternary$_{search}$

```
double ternary_search(double l, double r) {
  double eps = 1e-9; //error limit
  while (r - l > eps) {
    double m1 = l + (r - l) / 3;
    double m2 = r - (r - l) / 3;
    double f1 = f(m1);//evaluates at m1
    double f2 = f(m2);//evaluates at m2
    if (f1 < f2)
      l = m1;
    else
      r = m2;
  }
  return f(l);//maximum of f(x) in [l, r]
}
```

2   Data Structure

2.1   BIT

```
int BIT[MAX];
void update(int idx,int MAX,int val) {
    while (idx <= MAX) {
        BIT[idx] += val;
        idx += (idx & -idx);
    }
}
int query(int idx) {
    int sum = 0;
    while (idx > 0) {
        sum += BIT[idx];
        idx -= (idx & -idx);
    }
    return sum;
}
int rangeQuery(int x1,int x2){
 return query(x2)-query(x2-1);
}
```

2.2   Merge Sort Tree

```
void build(int idx, int ss, int se,
  vector<int> a[], vector<int> sTree[]){
  if(ss == se){
    sTree[idx] = a[ss];
    return;
  }
  int mid = (ss+se)/2;
  build(2*idx+1, ss, mid, a, sTree);
  build(2*idx+2, mid+1, se, a, sTree);
  merge(sTree[2*idx+1].begin(), sTree[2*idx+1].end(),sTree
      [2*idx+2].begin(), sTree[2*idx+2].end(),back_inserter(
      sTree[idx]));
}
int queryRec(int node, int start, int end, int ss, int se,
  int k, vector<int> a[], vector<int> sTree[]) {
  if (ss > end || start > se) return 0;
  if (ss <= start && se >= end)return upper_bound(sTree[node
      ].begin(),sTree[node].end(), k)-sTree[node].begin();
  int mid = (start+end)/2;
  int p1 = queryRec(2*node+1, start, mid, ss, se, k, a,
      sTree);
  int p2 = queryRec(2*node+2, mid+1, end, ss, se, k, a,
      sTree);
  return p1 + p2;
}
```

2.3   Persistent segtree

```
void build(int id = ir,int l = 0,int r = n){
 s[id] = 0;
 if(r - l < 2)
  return ;
 int mid = (l+r)/2;
 L[id] = NEXT_FREE_INDEX ++;
 R[id] = NEXT_FREE_INDEX ++;
 build(L[id], l, mid);
 build(R[id], mid, r);
 s[id] = s[L[id]] + s[R[id]];
}
// Update function :
int upd(int p, int v,int id,int l = 0,int r = n){
 int ID = NEXT_FREE_INDEX ++; // index of the node in new
      version of segment tree
 s[ID] = s[id] + 1;
 if(r - l < 2)
  return ID;
 int mid = (l+r)/2;
```

```cpp
L[ID] = L[id], R[ID] = R[id]; // in case of not updating
    the interval of left child or right child
if(p < mid)
 L[ID] = upd(p, v, L[ID], l, mid);
else
 R[ID] = upd(p, v, R[ID], mid, r);
return ID;
}
// Ask function (it returns i, so you should print api :4
int ask(int id, int ID, int k, int l = 0,int r = n){// id is
    the index of the node after l-1-th update (or ir) and
    ID will be its index after r-th update
if(r - l < 2) return l;
int mid = (l+r)/2;
if(s[L[ID]] - s[L[id]] >= k)// answer is in the left child'
    s interval
 return ask(L[id], L[ID], k, l, mid);
else
 return ask(R[id], R[ID], k - (s[L[ID]] - s[L[id]] ), mid,
    r);// there are already s[L[ID]] - s[L[id]] 1s in the
    left child's interval
}
```

2.4 segment tree with lazy propagation

```cpp
ll ar1[200000];
ll tree1[500000],lazy1[500000];
void build1(ll node,ll start,ll en){
  if(start==en){
    tree1[node]=ar1[start];}
  else{
    ll mid=(start+en)/2;
    build1(2*node+1,start,mid);
    build1(2*node+2,mid+1,en);
    tree1[node]=tree1[2*node+1]+tree1[2*node+2];
  }
}
void updatelazy1(ll node,ll start,ll en,ll l,ll r,ll val){
  if(lazy1[node]!=0){
    tree1[node]+=(en-start+1)*lazy1[node];
    if(start!=en){
      lazy1[2*node+1]+=lazy1[node];
      lazy1[2*node+2]+=lazy1[node];
    }
    lazy1[node]=0;
  }
  if(start>r||en<l||start>en) return;
  if(start>=l&&en<=r){
    tree1[node]+=(en-start+1)*val;
```

```cpp
  if(start!=en){
    lazy1[2*node+1]+=val;
    lazy1[2*node+2]+=val;
  }
  return;
}
ll mid=(start+en)/2;
updatelazy1(2*node+1,start,mid,l,r,val);
updatelazy1(2*node+2,mid+1,en,l,r,val);
tree1[node]=tree1[2*node+1]+tree1[2*node+2];
}
ll querylazy1(ll node,ll start,ll en,ll l,ll r){
  if(lazy1[node]!=0){
    tree1[node]+=(en-start+1)*lazy1[node];
    if(start!=en){
      lazy1[2*node+1]+=lazy1[node];
      lazy1[2*node+2]+=lazy1[node];
    }
    lazy1[node]=0;
  }
  if(start>r||en<l||start>en) return 0;
  if(start>=l&&en<=r) return tree1[node];
  ll mid=(start+en)/2;
  ll p1=querylazy1(2*node+1,start,mid,l,r);
  ll p2=querylazy1(2*node+2,mid+1,en,l,r);
  return p1+p2;
}
```

2.5 sparse table range-min

```cpp
void process2(int M[MAXN][LOGMAXN], int A[MAXN], int N){
int i, j;
//initialize M for the intervals with length 1
for (i = 0; i < N; i++)
 M[i][0] = i;
//compute values from smaller to bigger intervals
for (j = 1; 1 << j <= N; j++)
 for (i = 0; i + (1 << j) - 1 < N; i++)
  if (A[M[i][j-1]] < A[M[i+(1<<(j-1))][j-1]])
   M[i][j] = M[i][j - 1];
  else
   M[i][j] = M[i + (1 << (j - 1))][j - 1];
}
// query
int j = log[R - L + 1];
int minimum = min(st[L][j], st[R - (1 << j) + 1][j]);
```

3   Geometry

3.1   convex hull

```cpp
struct pt {
 double x, y;
};
bool cmp (pt a, pt b) {
 return a.x <b.x || a.x == b.x && a.y <b.y;
}
bool cw (pt a, pt b, pt c) {
 return a.x * (b.y-c.y) + b.x * (c.y-a.y) + c.x * (a.y-b.y)
     <0;
}
bool ccw (pt a, pt b, pt c) {
 return a.x * (b.y-c.y) + b.x * (c.y-a.y) + c.x * (a.y-b.y)>
     0;
}
void convex_hull (vector <pt> & a) {
 if (a.size () == 1) return;
 sort (a.begin (), a.end (),cmp);
 pt p1 = a [0], p2 = a.back ();
 vector <pt> up, down;
 up.push_back (p1);
 down.push_back (p1);
 for (size_t i = 1; i <a.size (); ++ i) {
  if (i == a.size () - 1 || cw (p1, a [i], p2)) {
   while (up.size () >= 2 &&! cw (up [up.size () - 2], up [
       up.size () - 1], a [i]))
    up.pop_back ();
   up.push_back (a [i]);
  }
  if (i == a.size () - 1 || ccw (p1, a [i], p2)) {
   while (down.size () >= 2 &&! ccw (down [down.size () -
       2], down [down.size () - 1], a [i]))
    down.pop_back ();
   down.push_back (a [i]);
  }
 }
 a.clear ();
 for (size_t i = 0; i <up.size (); ++ i)
  a.push_back (up [i]);
 for (size_t i = down.size () - 2; i> 0; --i)
  a.push_back (down [i]);
}
```

3.2   line sweep

```cpp
ll t,i,j,k,m,n;
cin>>n;ll arx1[n],arx2[n];
for(i=0;i<n;i++)cin>>arx1[i]>>arx2[i];
vector<pair<pair<ll,ll>,ll> >v;
for(i=0;i<n;i++){
  v.push_back(make_pair(make_pair(arx1[i],2),i));
  v.push_back(make_pair(make_pair(arx2[i]+1,-2),i));
}
sort(v.begin(),v.end());
set<ll>ss;ll ans=0;
for(i=0;i<v.size();i++){
  if(v[i].first.second==2){
    ss.insert(v[i].second);}
  if(v[i].first.second==-2){
    ll y=ss.size();ans=max(ans,y);
    ll x=v[i].second;ss.erase (x);
  }}
cout<<ans;
```

4 Graph

4.1 LCA in O(logn)

```cpp
void dfss(ll cur, ll prev){
  lev[cur] = lev[prev] + 1;
  parent[cur][0] = prev;
  for (ll i=0; i<v[cur].size(); i++)
    if (v[cur][i] != prev)
      dfss(v[cur][i], cur);
}
void precomputeSparseMatrix(ll n){
  ll kk=20;
  for (ll i=1; i<kk; i++)
    for (ll node = 1; node <= n; node++)
      if (parent[node][i-1] != -1)
        parent[node][i]=parent[parent[node][i-1]][i-1];
}
ll lca(ll u, ll v){
  ll kk=20;
  if (lev[v] < lev[u])
    swap(u, v);
  ll diff = lev[v] - lev[u];
  for (ll i=0; i<kk; i++)
    if ((diff>>i)&1)
      v = parent[v][i];
  if(u == v)
    return u;
```

```cpp
  for(ll i=kk-1; i>=0; i--)
    if(parent[u][i] != parent[v][i]){
      u = parent[u][i];
      v = parent[v][i];
    }
  return parent[u][0];
}
```

4.2 articulation-point-and-bridges

```cpp
#define mod 1000000007
void dfs(int u, vector<vector<int> >& adjList, vector<int>&
    disc, vector<int>& low,
vector<bool>& visited, vector<bool>& ap,
vector<pair<int, int> >& bridges, int parent){
  static int time=1;
  disc[u] = low[u] = time++;
  int child=0;
  visited[u] = true;
  for (auto& ele: adjList[u]){
    if (!visited[ele]){
      child++;
      dfs(ele, adjList, disc, low, visited, ap, bridges, u);
      low[u] = min(low[u], low[ele]);
      if (parent==-1 && child>1)
        ap[u] = true;
      else if (parent!=-1 && low[ele]>=disc[u])
        ap[u] = true;
      if (low[ele]>disc[u])
        bridges.push_back({u, ele});
    }else if (ele != parent){
      low[u] = min(low[u], disc[ele]);
    }
  }
}
int main() {
int n, m;
cin >> n >> m;
vector<vector<int> > adjList(n);
for (int i=0;i<m;i++){
int a, b;
cin >> a >> b;
adjList[a].push_back(b);
adjList[b].push_back(a);}
vector<bool> visited(n, false);
vector<bool> ap(n, false);
vector<pair<int, int> > bridges;
vector<int> disc(n, INT_MAX);
vector<int> low(n, INT_MAX);
```

```cpp
int count=0;
for (int i=0;i<n;i++){
 if (!visited[i])
 dfs(i, adjList, disc, low, visited, ap, bridges, -1);
}
for (int i=0;i<n;i++){
 if (ap[i])
  count++;
}
cout << count << endl;
for (int i=0;i<n;i++){
 if (ap[i]) cout << i << " ";
}cout << endl;
auto compare = [&](auto a, auto b){
return a.first<b.first || (a.first==b.first && a.second<b.
    second);
};
sort(bridges.begin(), bridges.end(), compare);
cout << bridges.size() << endl;
for (auto& ele: bridges){
cout << ele.first << " " << ele.second << endl;}
return 0;}
```

4.3 bellman ford for negative cycle

```cpp
vector<ll>v[10000];
ll dis[10000];
#define inf 1000000000
int main(){
  cin>>n>>m;
  for(i=0;i<m+2;i++){
    v[i].clear();
    dis[i]=inf;// upto n
  }
  for(i=0;i<m;i++){
    cin>>x>>y>>z;
    v[i].push_back(x);
    v[i].push_back(y);
    v[i].push_back(z);
  }
  dis[1]=0;//initiate dis[source]=0
  for(i=0;i<n-1;i++){
    for(j=0;j<m;j++){
      if(dis[v[j][0]]+v[j][2]<dis[v[j][1]])
        dis[v[j][1]]=dis[v[j][0]]+v[j][2];
    }
  }
  for(i=0;i<m;i++){
    if(dis[v[i][0]]!=inf){
```

```cpp
        if(dis[v[i][0]]+v[i][2]<dis[v[i][1]]){
            cout<<"negative cycle\n";
            return 0;
        }
    }
  }
  cout<<"negative cycle not found";
  return 0;
}
```

4.4 bfs

```cpp
ll bfs(ll a,ll b){
    queue<ll>q;q.push(a);q.push(b);ar[a][b]=0;
    ll ax[]={1,0,-1,0}; ll ay[]={0,1,0,-1};
    while(!q.empty()){
        ll u=q.front();q.pop();
        ll v=q.front();q.pop();
        ll i,j;
        for(i=0;i<4;i++){
            ll uu=ax[i]+u;ll vv=ay[i]+v;
            if(check(uu,vv)){
                if(ar[u][v]+1<ar[uu][vv]){
                    ar[uu][vv]=ar[u][v]+1;
                    q.push(uu);
                    q.push(vv);
                }
            }
        }
    }
}
```

4.5 dijkshtra shortest path

```cpp
vector<pair<ll,ll> >v[100005];
map<pair<ll,ll>,ll>mm;
ll dp[100005],visit[100005];
#define inf 1000000000000
ll dij(ll n){
    ll i,j;
    for(i=0;i<=n;i++){
        dp[i]=inf;}
    priority_queue<pair<ll,ll>,vector<pair<ll,ll> >,greater<
        pair<ll,ll> > >q;
    q.push(pair<ll,ll>(0,1));dp[1]=0;
    while(!q.empty()){
        ll u=q.top().second;
```

```cpp
        ll w=q.top().first;q.pop();
        if(visit[u])
            continue;
        for(i=0;i<v[u].size();i++){
            if(!visit[v[u][i].first]&&dp[v[u][i].first]>w+v[u
                ][i].second){
                dp[v[u][i].first]=w+v[u][i].second;
                q.push(pair<ll,ll>(dp[v[u][i].first],v[u][i].
                    first));
            }
        }
        visit[u]=1;
        if(u==n)
            break;
    }
    if(dp[n]==inf)
        return -1;
    return dp[n];
}
```

4.6 euler circuit and tour

```cpp
// Following are some interesting properties of undirected
    graphs with an Eulerian path and cycle. We can use
    these properties to find whether a graph is Eulerian or
    not.
// Eulerian Cycle
// An undirected graph has Eulerian cycle if following two
    conditions are true.
// (a) All vertices with non-zero degree are connected. We
    dont care about vertices with zero degree because they
    dont belong to Eulerian Cycle or Path (we only consider
    all edges).
// (b) All vertices have even degree.
// Eulerian Path
// An undirected graph has Eulerian Path if following two
    conditions are true.
// (a) Same as condition (a) for Eulerian Cycle
// (b) If two vertices have odd degree and all other
    vertices have even degree. Note that only one vertex
    with odd degree is not possible in an undirected graph
    (sum of all degrees is always even in an undirected
    graph)
// Note that a graph with no edges is considered Eulerian
    because there are no edges to traverse.
find_tour(u):
  for each edge e=(u,v) in E:
    remove e from E
    find_tour(v)
```

```
prepend u to tour
where u is any vertex with a non-zero degree.
```

4.7 floyd warshall for negative cycle

```cpp
for(int k = 1; k <= n; k++){
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            dist[i][j] = min( dist[i][j], dist[i][k] + dist[k
                ][j] );}}}
for (int i = 0; i < V; i++)
        if (dist[i][i] < 0)
            return true;
    return false;
```

4.8 mst kruskal algo

```cpp
const int MAX = 1e4 + 5;
int id[MAX], nodes, edges;
pair <long long, pair<int, int> > p[MAX];
void initialize(){
  for(int i = 0;i < MAX;++i)
    id[i] = i;
}
int root(int x){
  while(id[x] != x){
    id[x] = id[id[x]];
    x = id[x];}
  return x;
}
void union1(int x, int y){
  int p = root(x);
  int q = root(y);
  id[p] = id[q];
}
long long kruskal(pair<long long, pair<int, int> > p[]){
  // p is sorted
  int x, y;long long cost, minimumCost = 0;
  for(int i = 0;i < edges;++i){
    // Selecting edges one by one in increasing order
    x = p[i].second.first;y = p[i].second.second;
    cost = p[i].first;
    // Check if the selected edge is creating a cycle or not
    if(root(x) != root(y)){
      minimumCost += cost;union1(x, y);
    }
  }
```

```
    return minimumCost;
}
```

4.9 strongly connected component

```
vector<ll>v[10000],vv[10000];
ll visit[10000];vector<ll>s;
void dfs(ll a){
  visit[a]=1;
  for(ll i=0;i<v[a].size();i++)
    if(!visit[v[a][i]])
      dfs(v[a][i]);
  s.push_back(a);
}
void dfss(ll a){
  visit[a]=1;cout<<a<<" ";
  for(ll i=0;i<vv[a].size();i++)
    if(!visit[vv[a][i]])
      dfss(vv[a][i]);
}
int main(){
    cin>>n>>m;
    for(i=0;i<m;i++){
        cin>>j>>k;
        v[j].push_back(k);
        vv[k].push_back(j);}
    for(i=1;i<=n;i++){
        if(!visit[i]){
            dfs(i);
}}
memset(visit,0,sizeof(visit));
for(i=s.size()-1;i>=0;i--){
  if(!visit[s[i]]){
    dfss(s[i]);
    cout<<"\n";
  }
}}
```

4.10 topological sorting

```
vector<int>v;vector<int> adj[25];
bool vis[25];
void dfs(int s){
 vis[s] = true;
 for(int i=0;i<adj[s].size();i++){
  int to = adj[s][i];
  if(vis[to]==false){
```

```
    dfs(to);
  }}
 v.push_back(s);
}
void init(){
for(int i=0;i<25;i++) vis[i] = false;}
int main(){
 int n,m;int x,y;
 cin>>n>>m;init();
 for(int i=0;i<m;i++){
  cin>>x>>y;
  adj[x].push_back(y);
 }
 for(int i=1;i<=n;i++){
  sort(adj[i].rbegin(),adj[i].rend());}
 for(int i=n;i>=1;i--){
  if(vis[i]==false) dfs(i);}
 for(int i=v.size()-1;i>=0;i--){
    cout<<v[i]<<" ";}
return 0;}
```

5 Math

5.1 Lucas theorem

```
//Computes C(N,R) modulo P in O(log(n)) time.
LL Lucas(LL N,LL R,int P)
{
 if(R<0||R>N) return 0;
 if(R==0||R==N) return 1;
 if(N>=P) return (Lucas(N/P,R/P,P)*Lucas(N%P,R%P,P))%P;
 return (Fact[N]*(Invfact[N-R]*Invfact[R])%P)%P;
}
```

5.2 catal number

```
// The Catalan number Cn is the solution for
//    Number of correct bracket sequence consisting of n
    opening and n closing
// brackets.
//    The number of rooted full binary trees with n + 1
    leaves (vertices are not
// numbered). A rooted binary tree is full if every vertex
    has either two
// children or no children.
//    The number of ways to completely parenthesize n + 1
    factors.
//    The number of triangulations of a convex polygon with
    n + 2 sides (i.e. the
// number of partitions of polygon into disjoint triangles
    by using the diagonals).
//    The number of ways to connect the 2n points on a
    circle to form n disjoint
// chords.
//    The number of non-isomorphic full binary trees with n
    internal nodes
// (i.e. nodes having at least one son).
//    The number of monotonic lattice paths from point (0,
    0) to point (n, n) in
// a square lattice of size n  n, which do not pass above
    the main diagonal
// (i.e. connecting (0, 0) to (n, n)).
//    Number of permutations of length n that can be stack
    sorted (i.e. it can be
// shown that the rearrangement is stack sorted if and only
    if there is no such
// index i < j < k, such that ak < ai < aj ).
//    The number of non-crossing partitions of a set of n
    elements.
//    The number of ways to cover the ladder 1 . . . n using
    n rectangles (The
// ladder consists of n columns, where i
// th column has a height i).
const int MOD = ....
const int MAX = ....
int catalan[MAX];
void init() {
 catalan[0] = catalan[1] = 1;
 for (int i=2; i<=n; i++) {
  catalan[i] = 0;
  for (int j=0; j < i; j++) {
   catalan[i] += (catalan[j] * catalan[i-j-1]) % MOD;
   if (catalan[i] >= MOD) {
    catalan[i] -= MOD;
   }
  }
 }
}
```

5.3 discrete log

```
int solve (int a, int b, int m) {
 int n = (int) sqrt (m + .0) + 1;
 int an = 1;
```

```cpp
for (int i=0; i<n; ++i)
 an = (an * a) % m;
map<int,int> vals;
for (int i=1, cur=an; i<=n; ++i) {
 if (!vals.count(cur))
  vals[cur] = i;
 cur = (cur * an) % m;
}
for (int i=0, cur=b; i<=n; ++i) {
 if (vals.count(cur)) {
  int ans = vals[cur] * n - i;
  if (ans < m)
   return ans;
 }
 cur = (cur * a) % m;
}
return -1;
}
```

5.4 extended eucledian

```cpp
ll x,y,d;
void extende(ll a,ll b){
 if(b==0){
    d=a;x=1;y=0;
 }else{
  extende(b,a%b);
  ll temp=x;
  x=y;
  y=temp-(a/b)*y;
}}
extende(16,10);
cout<<d<<"\n";
cout<<x<<" "<<y;
```

5.5 factorial Mod

```cpp
ll factmod(ll n,ll p) {
 ll res = 1;
 while (n > 1) {
  res = (res*((n/p)%2?p-1:1)) % p;
  for(int i = 2; i <= n%p; ++i)
   res = (res * i) % p;
  n /= p;
 }
 return res % p;
}
```

```cpp
// This implementation works in O(p*logp(n))
```

5.6 matrix exponentiation

```cpp
void mul(lo g[2][2],lo h[2][2]){
  lo u=(g[0][0]*h[0][0])%mm+(g[0][1]*h[1][0])%mm;
  lo v=(g[0][0]*h[0][1])%mm+(g[0][1]*h[1][1])%mm;
  lo w=(g[1][0]*h[0][0])%mm+(g[1][1]*h[1][0])%mm;
  lo x=(g[1][0]*h[0][1])%mm+(g[1][1]*h[1][1])%mm;
  g[0][0]=u%mm;g[0][1]=v%mm;
  g[1][0]=w%mm;g[1][1]=x%mm;
}
void pr(lo g[2][2],lo n,lo p){
  if(n==0||n==1){
 return;}
  lo h[2][2]={{p,-1+mm},{1,0}};
  pr(g,n/2,p);
  mul(g,g);
  if(n%2!=0){
    mul(g,h);
  }
}
```

5.7 mul ab

```cpp
ll mul(ll a,ll b,ll mod){
  ll res=0;
  while(b>0){
    if(b%2!=0)
      res=(res+a)%mod;
    a=(a*2)%mod;
    b/=2;
  }
  return res;
}
```

5.8 primality-test-milankarp

```cpp
bool miller(ll n,ll d,ll a){
 srand(time(0));
 //ll a=rand()%(n-4) + 2;
 ll x=me(a,d,n);
 if(x==1||x==n-1)
  return true;
 while(d!=n-1){
```

```cpp
  //x=(x*x)%n;
  x=mulmod(x,x,n);
  d*=2;
  if(x==1){return false;}
  if(x==n-1){return true;}
 }
 return false;
}
bool isPrime(ll n){
 if(n==1){return false;}
 if(n<=3){return true;}
 if(n%2==0){return false;}
 ll d=n-1;
 while(d%2==0){
  d/=2;}
 vector<ll>a = {2,3,5,7,11,13,17,19,23,29,31,37,41};
 ll k=a.size();
 ll i=-1;
 while(k--){
  i++;
  if(a[i]<=(n-2)){
  if(!miller(n,d,a[i])){
   //cout<<a[i]<<" "<<d<<"\n";
   return false;
  }}
 }
 return true;
}
```

5.9 segmented sieve

```cpp
vector<ll>f;
void ss(ll l,ll r){
 bool isPrime[r - l + 1];
  memset(isPrime,true,sizeof(isPrime));
 for (ll i = 2; i * i <= r; ++i)
  for (ll j = max(i*i, (l+(i-1))/i*i); j<=r;j+= i)
   isPrime[j - l] = false;
 ll k=2;
 for (ll i = max(l, k); i <= r; ++i){
  if (isPrime[i - l])
   f.push_back(i);
 }
}
////
// 1 <= l <= r <= ??
vector<bool> seg_sieve(ll l,ll r){
 ll i,j;
 vector<bool> V(r-l+1,true);
```

```
for(i=2;i*i<=r;++i){
 for(j=max(2LL,(l+i-1)/i)*i;j<=r;j+=i)
  V[j-l]=false;
 }
 if(l==1) V[1-l]=0;
 return V;
}
```

5.10   totient in O(sqrt)

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

5.11   totient(nlogn)

```
ll tot[200005];
void totient(){
 ll i,j;tot[1]=1;
 for(i=2;i<=200000;i++){
 if(tot[i]==0){
   tot[i]=i-1;
  for(j=i*2;j<=200000;j+=i){
   if(tot[j]==0)
    tot[j]=j;
   tot[j]=(tot[j]/i)*(i-1);
  }
 }
}}
```

6   String

6.1   KMP

```
public int[] getLPSArray(char s[]) {
  int n = s.length;
```

```
  int lps[] = new int[n];
  lps[0] = 0;
  int len = 0;
  for (int i = 1; i < n;) {
    if (s[i] == s[len]) {
      lps[i++] = ++len;
    }else{
      if (len == 0) {
        lps[i++] = 0;
      }else{
        len = lps[len - 1];
      }
    }
  }
  return lps;
}
void KMPSearch(String pat, String txt) {
  int M = pat.length();
  int N = txt.length();
  int lps[] =getLPSArray(pat.toCharArray());
  int j = 0,i = 0;
  while(i<N){
    if (pat.charAt(j) == txt.charAt(i)) {
      j++; i++;
    }
    if (j == M) {
      out.println("Found at index " + (i - j));
      j = lps[j - 1];
    }
    else if (i < N && pat.charAt(j)!=txt.charAt(i)) {
      if (j != 0)
        j = lps[j - 1];
      else
        i = i + 1;
    }
  }
}
```

6.2   Manaker Algorithm palindrome

```
// string s = "aabaabaa";
// d1: [1, 1, 3, 2, 1, 3, 2, 1]
// d2: [0, 1, 0, 0, 4, 0, 0, 1]
vector<ll> d1 (n);
 l=0, r=-1;
for ( i=0; i<n; ++i) {
 ll k = (i>r ? 0 : min (d1[l+r-i], r-i)) + 1;
 while (i+k < n && i-k >= 0 && s[i+k] == s[i-k]) ++k;
 d1[i] = k--;
```

```
 if (i+k > r)
  l = i-k, r = i+k;
}
vector<ll> d2 (n);
 l=0, r=-1;
for ( i=0; i<n; ++i) {
 ll k = (i>r ? 0 : min (d2[l+r-i+1], r-i+1LL)) + 1;
 while (i+k-1 < n && i-k >= 0 && s[i+k-1] == s[i-k]) ++k;
 d2[i] = --k;
 if (i+k-1 > r)
  l = i-k, r = i+k-1;
}
```

6.3   prefix function

```
vector<int> prefix_function(string s) {
  int n = (int)s.length();
  vector<int> pi(n);
  for (int i = 1; i < n; i++) {
    int j = pi[i-1];
    while (j > 0 && s[i] != s[j])
      j = pi[j-1];
    if (s[i] == s[j]) j++;
    pi[i] = j;
  }
  return pi;
}
```

6.4   suffix array

```
// add with $ in the back
vector<ll> suffix(string &s){
  ll n = s.size(),i,j,alphabet=267;
  vector<ll> p(n,0),cnt(alphabet,0),c(n,0),pn(n,0),cn(n,0);
  vector<pair<char,ll> > Fi;
  for(i=0;i<n;++i)Fi.push_back({s[i],i});
  sort(Fi.begin(),Fi.end());
  for(i=0;i<n;++i){p[i]=Fi[i].second;++cnt[s[i]];}
  c[p[0]] = 0;
  ll classes = 1;
  for (i=1;i<n;++i) {
   if (s[p[i]] != s[p[i-1]]) ++classes;
   c[p[i]] = classes-1;
  }
  for (ll h=0; (1<<h)<n; ++h) {
   for (i=0; i<n; ++i) {
    pn[i] = p[i] - (1<<h);
```

```cpp
    if (pn[i] < 0) pn[i] += n;
 }
 cnt.clear();
 cnt.resize(classes,0);
 for (i=0; i<n; ++i)
  ++cnt[c[pn[i]]];
 for (i=1; i<classes; ++i)
  cnt[i] += cnt[i-1];
 for (i=n-1; i>=0; --i)
  p[--cnt[c[pn[i]]]] = pn[i];
 cn[p[0]] = 0;
 classes = 1;
 for (i=1; i<n; ++i) {
  ll mid1 = (p[i] + (1<<h)) % n, mid2 = (p[i-1] + (1<<h)) %
         n;
  if (c[p[i]] != c[p[i-1]] || c[mid1] != c[mid2])
   ++classes;
  cn[p[i]] = classes-1;
 }
 c = cn;
 }
 return p;
}
```

6.5   trie using pointer

```cpp
struct trie{
 trie *children[26];
 bool endofword;
 ll weight;
};
trie *newnode(){
 struct trie *node=new trie();
 for(ll i=0;i<26;i++){
  node->children[i]=NULL;
 }
 nodode->endofword=false;
 node->weight=-1;
 return node;
}
void inser(trie *root,string s,ll weight){
 struct trie *node=new trie();
 node=root;
 for(ll i=0;i<s.length();i++){
  ll ind=s[i]-'a';
  if(!node->children[ind]){
   node->children[ind]=newnode();
  }
  node=node->children[ind];
```

```cpp
  if(node->weight<weight){
   node->weight=weight;
  }
  node->endofword=true;
 }
}
void searc(trie *root,string s){
 struct trie *node=new trie();
 node = root;
 for(ll i=0;i<s.length();i++){
  ll ind=s[i]-'a';
  if(!node->children[ind]){
   cout<<"-1\n";
   return;
  }
  node=node->children[ind];
 }
 cout<<node->weight<<"\n";
return;}
```

6.6   trie without using pointer

```cpp
ll trie[100005][26],finish[100005],nodeweight[100005];
ll nxt=1;
void add(string s,ll weight){
 ll node=0;ll i,j,k;
 for(i=0;i<s.length();i++){
  if(trie[node][s[i]-'a']==0){
   trie[node][s[i]-'a']=nxt;
   node=nxt;
   nxt++;
  }else{
   node=trie[node][s[i]-'a'];
  }
  if(nodeweight[node]<weight){
   nodeweight[node]=weight;
  }
  finish[nxt-1]=1;
 }
}
void findd(string s){
 ll node=0;ll i,j;
 for(i=0;i<s.length();i++){
  if(trie[node][s[i]-'a']==0){
   cout<<"-1\n";return ;}
  node=trie[node][s[i]-'a'];
 }
 cout<<nodeweight[node]<<"\n";
}
```

6.7   z algo prefix match

```cpp
// string s = "aaabaab";
// debug : [0, 2, 1, 0, 2, 1, 0]
vector<ll> z_function (string s) {
 ll n = (ll) s.size();
 vector<ll> z (n);
 for (ll i=1, l=0, r=0; i<n; ++i) {
  if (i <= r)
   z[i] = min (r-i+1LL, z[i-l]);
  while (i+z[i] < n && s[z[i]] == s[i+z[i]])
   ++z[i];
  if (i+z[i]-1 > r)
   l = i, r = i+z[i]-1;
 }
 return z;
}
```

7   c++ sublime build linux

```json
{
 "shell_cmd": "g++ ${file} -o ${file_path}/${file_base_name
    }&&gnome-terminal -- bash -c '${file_path}/${
    file_base_name};read -p \"\nPlease press any key to
    exit...\"'",
 "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?:? (.*)$",
  "working_dir": "${file_path}",
  "selector": "source.c, source.c++",
  "variants":
  [
   {
    "name":"Run with more Memory",
    "shell":true,
    "shell_cmd":"g++ ${file} -o ${file_path}/${
       file_base_name}&&gnome-terminal -- bash -c 'ulimit
       -s 1000000;${file_path}/${file_base_name};read -p
       \"\nPlease press any key to exit...\"'"
   },
   {
    "name":"Run with More More Memory",
    "shell":true,
    "shell_cmd":"g++ ${file} -o ${file_path}/${
       file_base_name}&&gnome-terminal -- bash -c 'ulimit
       -s 2000000;${file_path}/${file_base_name};read -p
       \"\nPlease press any key to exit...\"'"
```

```
        },
        {
          "name":"Run with Very More More Memory",
          "shell":true,
          "shell_cmd":"g++ ${file} -o ${file_path}/${
                file_base_name}&&gnome-terminal -- bash -c 'ulimit
                -s 4000000;${file_path}/${file_base_name};read -p
                \"\nPlease press any key to exit...\"'"
        }
      ]
}
```

8   c++ sublime build windows

```
{
    "cmd": [
        "g++", "${file}", "-DDARKKEKS", "-std=c++14", "-Wl,--
            stack=268435456", "-o", "${file_path}/${
            file_base_name}",
        "&&",
        "start", "cmd.exe", "@cmd", "/c",
            "${file_path}/${file_base_name} && echo. && pause
                "
    ],
    "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?:? (.*)$",
    "working_dir": "${file_path}",
    "selector": "source.c, source.c++",
    "shell": "true",
    "variants": [{
        "name": "Run",
        "cmd": ["start", "cmd.exe", "@cmd", "/c", "${
            file_path}/${file_base_name} && echo. && pause"]
    }]
}
```

9   policy-based-ds

```
#include<bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp>
#include <functional> // for less
#include <iostream>
using namespace std;
using namespace __gnu_pbds;
#define ll long long int
```

```
typedef tree<int, null_type, less_equal<int>, rb_tree_tag,
            tree_order_statistics_node_update>
    new_data_set;
int main(){
 new_data_set p;
 p.insert(8LL);
 *p.find_by_order(3);//element
 p.order_of_key(6);//index
 return 0;
}
```

10   template Arjunwa

```
#include<bits/stdc++.h>
#define pb push_back
#define all(v) (v).begin(),(v).end()
#define ll long long int
#define rep(i,x,y) for(ll i=a;i<b;i++)
#define sz(a) (ll)(a.size())
#define mod 1000000007
#define fi first
#define se second
#define pii pair<ll,ll>
using namespace std;
int main(){
 ll t,i,j,k,m,n;
 while(t--){
 }
 return 0;
}
```

11   template Avinash

```
#include <bits/stdc++.h>
#define ll long long int
#define MOD 1000000007

using namespace std;

template <typename T, typename S>
ostream& operator<<(ostream& os, const pair<T, S>& v){
    os<<"("<<v.first<<","<<v.second<<")";
    return os;
}
template <typename T>
```

```
ostream& operator<<(ostream& os, const set<T>& v){
    os << "debug : [";
    for (auto it:v){os << it;
        if (it != *v.rbegin())
            os << ", ";
    }
    os << "]\n\n";
    return os;
}
template <typename T>
ostream& operator<<(ostream& os, const vector<T>& v)
{
    os << "debug : [";
    for (int i = 0; i < v.size(); ++i) {
        os << v[i];
        if (i != v.size() - 1)
            os << ", ";
    }
    os << "]\n\n";
    return os;
}
template <typename T, typename S>
ostream& operator<<(ostream& os, const map<T, S>& v)
{
 os << "debug : \n";
    for (auto it : v)
        os << it.first << " : "
            << it.second << "\n\n";

    return os;
}
int main(){
 ll t,i,j,k,l,n,m,x,y,a,b,c,r,q;
 // ios_base::sync_with_stdio(false);
 // cin.tie(NULL);
 // fopen()
 return 0;
}
```

12   template Punee

```
class Template implements Runnable{
  public void solve(){}
  InputReader in;PrintWriter out;
  @Override
  public void run() {
    InputStream inputStream = System.in;
    OutputStream outputStream = System.out;
```

```java
    try {
      if (System.getProperty("user.name").equals("puneet")) {
        outputStream = new FileOutputStream("path/output.txt"
            );
        inputStream = new FileInputStream("path/input.txt");
      }
    } catch (Exception ignored) {}
    out = new PrintWriter(outputStream);
    in = new InputReader(inputStream);
    solve();
    out.flush();
  }
  public static void main(String[] args) {
    new Thread(null,new Template(),"Main",1<<27).start();
  }
  class InputReader {
    InputStream obj;
    public InputReader(InputStream obj) {
      this.obj = obj;
    }
    byte inbuffer[] = new byte[1024];
    int lenbuffer = 0, ptrbuffer = 0;
    int readByte() {
      if (lenbuffer == -1) throw new InputMismatchException()
          ;
      if (ptrbuffer >= lenbuffer) {
        ptrbuffer = 0;
        try {
          lenbuffer = obj.read(inbuffer);
        } catch (IOException e) {
          throw new InputMismatchException();
        }
      }
```

```java
    }
    if (lenbuffer <= 0) return -1;
    return inbuffer[ptrbuffer++];
  }
  String ns() {
    int b = skip();
    StringBuilder sb = new StringBuilder();
    while (!(isSpaceChar(b))) // when nextLine, (
        isSpaceChar(b) && b!=' ')
    {
      sb.appendCodePoint(b);
      b = readByte();
    }
    return sb.toString();
  }
  int ni() {
    int num = 0, b;
    boolean minus = false;
    while ((b = readByte()) != -1 && !((b >= '0' && b <= '9
        ') || b == '-')) ;
    if (b == '-') {
      minus = true;
      b = readByte();
    }
    while (true) {
      if (b >= '0' && b <= '9') {
        num = num * 10 + (b - '0');
      }else {
        return minus ? -num : num;
      }
      b = readByte();
    }
```

```java
  }
  long nl() {
    long num = 0;
    int b;
    boolean minus = false;
    while ((b = readByte()) != -1 && !((b >= '0' && b <=
        '9') || b == '-')) ;
    if (b == '-') {
      minus = true;
      b = readByte();
    }
    while (true) {
      if (b >= '0' && b <= '9') {
        num = num * 10L + (b - '0');
      }else {
        return minus ? -num : num;
      }
      b = readByte();
    }
  }
  boolean isSpaceChar(int c) {
    return (!(c >= 33 && c <= 126));
  }
  int skip() {
    int b;
    while ((b = readByte()) != -1 && isSpaceChar(b)) ;
    return b;
  }
  float nf() {return Float.parseFloat(ns());}
  double nd() {return Double.parseDouble(ns());}
  char nc() {return (char) skip();}}}
```