Manacher (Palindrome)

```
//0 based
int m[2*lim+1]; //length of the longest palindrome centered at the index

int manacher(string &s)
{
   int len = s.size();
   if(len == 0) return -1;
   mem(m,0);
   m[0] = 0;
   m[1] = 1;
   // "cur" is the current center
   // "r" is the right bound of the palindrome
   // that centered at current center
   int cur, r;
   r = 2;
   cur = 1;
   int ma=1;
   for(int p2=2; p2<2*len+1; p2++)
   {
      int p1 = cur- (p2-cur);
      //if p1 is negative, we need to
      //move "cur" forward
      while(p1 < 0)
      {
         cur++;
         r = m[cur] + cur;
         p1 = cur- (p2-cur);
      }
      //If the first character of t is
      //strictly on the right of the
      // first character of s
      if(m[p1] < r - p2)
         m[p2] = m[p1];
      //otherwise
      else
      {
         //reset "cur"
         cur = p2;
         int k = r-p2;
         if(k<0) k = 0;
         while(1)
         {
            if((p2+k+1)&1)
            {
               if(p2+k+1 < 2*len+1 && p2-k-1 >=0 &&
s[(p2+k)/2] == s[(p2-k-2)/2])
                  k++;

 else break;
            }
            else
            {
               if(p2+k+1 < 2*len+1 && p2-k-1 >=0)
                  k++;
               else break;
            }

         }
         r = p2+k;
         m[p2] = k;
         ma=max(ma,k);
      }
   }
   return ma;
}


int main()
{
   string s;
   while(cin>>s)
   {
      print1(manacher(s));
   }
   return 0;
}
```

2-SAT

```
//0 based
VI adj[2*sz]; //2*sz for true and false
argument(only adj should be cleared)
int col[2*sz],low[2*sz],tim[2*sz],timer;
int
group_id[2*sz],components;//components=number
of components, group_id = which node belongs to
which node
bool ans[sz]; //boolean assignment ans
stack<int>S;
void scc(int u)
{
   int i,v,tem;
   col[u]=1;
   low[u]=tim[u]=timer++;
   S.push(u);
   fr(i,0,SZ(adj[u])-1)
   {
      v=adj[u][i];
      if(col[v]==1)
         low[u]=min(low[u],tim[v]);
      else if(col[v]==0)
      {
         scc(v);
         low[u]=min(low[u],low[v]);
      }
   }
   //SCC checking...
   if(low[u]==tim[u])
   {
      do
      {
         tem=S.top();S.pop();
         group_id[tem]=components;
         col[tem]=2; //Completed...
      }while(tem!=u);
      components++;
   }
}
int TarjanSCC(int n) //n=nodes (some change may
be required here)
{
   int i;
   timer=components=0;
   clr(col,0);
   while(!S.empty()) S.pop();
   fr(i,0,n-1) if(col[i]==0) scc(i);
   return components;
}
//double nodes needed normally
bool TwoSAT(int n) //n=nodes (some change may
be required here)
{
   TarjanSCC(n);
   int i;
   for(i=0;i<n;i+=2)
   {
      if(group_id[i]==group_id[i+1])
         return false;
      if(group_id[i]<group_id[i+1]) //Checking who
is lower in Topological sort
         ans[i/2]=true;
      else ans[i/2]=false;
   }
   return true;
}
void add(int ina,int inb)
{
   adj[ina].pb(inb);
}
int complement(int n)
{
   return n^1;
}
void initialize(int n)
{
   for(int i=0;i<n;i++)
      adj[i].clear();
}
int main()
{
   int n, m, i, u, v;
   while(~scanf("%d %d", &n, &m))
   {
      initialize(n<<1);
      fr(i,0,m-1)
      {
         scanf("%d %d", &u, &v);
         if(u>0) u = 2*u-2;
         else u = -2*u-1;
         if(v>0) v = 2*v-2;
         else v = -2*v-1;
```

```
    clr(col,0);
    while(!S.empty()) S.pop();
    fr(i,0,n-1) if(col[i]==0) scc(i);
    return components;
}
//double nodes needed normally
bool TwoSAT(int n) //n=nodes (some change may
be required here)
{
    TarjanSCC(n);
    int i;
    for(i=0;i<n;i+=2)
    {
        if(group_id[i]==group_id[i+1])
            return false;
        if(group_id[i]<group_id[i+1]) //Checking who is
lower in Topological sort
            ans[i/2]=true;
        else ans[i/2]=false;
    }
    return true;
}
void add(int ina,int inb)
{
    adj[ina].pb(inb);
}
int complement(int n)
{
    return n^1;
}
void initialize(int n)
{
    for(int i=0;i<n;i++)
        adj[i].clear();
}
```

```
int main()
{
    int n, m, i, u, v;
    while(~scanf("%d %d", &n, &m))
    {
        initialize(n<<1);
        fr(i,0,m-1)
        {
            scanf("%d %d", &u, &v);
            if(u>0) u = 2*u-2;
            else u = -2*u-1;
            if(v>0) v = 2*v-2;
            else v = -2*v-1;
  add(complement(u),v);
            add(complement(v),u);
        }
        if(TwoSAT(n<<1)) puts("YES");
        else puts("NO");
    }
    return 0;
}
```