OI 模板整理【进阶篇】

YZ_HL

2018 年 11 月 4 日

前言

本文档是《OI 模板整理》系列的第二部分”进阶篇”。在这里我们分成五个板块来对 NOIP 提高组所需要的一些模板来进行讨论，分别为”数据结构”,”图论算法”,”字符串算法”,”数论算法”,”常见骗分手段”。

本文档主要都是以代码的形式给出，因为笔者认为在有了一个中心思路后直接阅读代码比看详细的解释要来的容易（除去数论算法，由于证明篇幅过长因此略去，文档中只提供代码）。

本文档中的部分代码整理于网上的一些题解，在此特表对各位写题解的神犇表示感谢。

本文档完成于 2018-11-13，更多的是笔者对于自己所掌握算法的一次总结，因此可能有很多地方是不详细的（因为笔者认为自己清楚了就略去不写）。如果有改进意见请联系 YZ_HL@oi-liu.com，笔者的网站为 https://www.oi-liu.com。欢迎各位神犇来踩一下。

有人问道为什么笔者这一篇文档不用”轻飏”作为名字了，在这里做出解释。因为笔者当时想的是在我们学校的同学不知情的情况下得到他们对于这个模板集合的评价，于是用了假名和新注册的邮箱。然而由于笔者的码风过于毒瘤一下就被认出来了。所以再写”轻飏”就没什么意义了。

大概就是这样，希望各位读者能在这份模板整理中获益，那就是对笔者最大的肯定。

目录

1 数据结构

1.1 最小生成树 -最小瓶颈路 (Luogu1396)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define maxn 100005
using namespace std;
int n, m, s, t;
int uset[maxn];
struct Edge{
    int u;
    int v;
    int w;
}G1[maxn];
bool cmp(const Edge &A, const Edge &B){
    return A.w < B.w;
}
int find(int x){
    return uset[x] == x ? x : uset[x] = find(uset[x]);
}
void Union(int x, int y){
    int fx = find(x);
    int fy = find(y);
    if(fx == fy)  return;
    uset[fx] = fy;
}
void init(){
    for(int i = 1; i <= n; i++)   uset[i] = i;
}
int main(void)
{
    scanf("%d %d %d %d", &n, &m, &s, &t); init();
    for(int i = 1; i <= m; i++)
        scanf("%d %d %d", &G1[i].u, &G1[i].v, &G1[i].w);
    sort(G1+1, G1+m+1, cmp);
    for(int i = 1, cnt = 0; i <= m && cnt <= n-1; i++)
    {
        if(find(G1[i].u) != find(G1[i].v))
        {
            Union(G1[i].u, G1[i].v);
            cnt++;
        }
        if(find(s) == find(t))
        {
            printf("%d\n", G1[i].w);
            return 0;
        }
    }
    return 0;
}
```

1.2 ST 表 -RMQ 问题 (Luogu3865)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <cmath>
#define maxn 100005
using namespace std;
int n, m, ST[maxn][25];
int Query(int l, int r){
  int k = (int)log2(r−l+1);
  return max(ST[l][k], ST[r−(1<<k)+1][k]);
}
int main(void)
{
  scanf(”%d %d”, &n, &m);
  for(int i = 1; i <= n; i++) scanf(”%d”, &ST[i][0]);
  for(int j = 1; j <= 20; j++)
    for(int i = 1; i+(1<<j)−1 <= n; i++)
      ST[i][j] = max(ST[i][j−1], ST[i+(1<<(j−1))][j−1]);
  for(int i = 1; i <= m; i++)
  {
    int l, r;
    scanf(”%d %d”, &l, &r);
    printf(”%d\n”, Query(l, r));
  }
  return 0;
}
```

1.3 树状数组 -单点修改 + 询问 (Luogu3374)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 1000005
using namespace std;
int n, m, c[maxn];
int lowbit(int k){
  return k&(-k);
}
int Query(int x){
  int ans = 0;
  for(int i = x; i; i -= lowbit(i)) ans += c[i];
  return ans;
}
void UpDate(int x, int k){
  for(int i = x; i <= n; i += lowbit(i))  c[i] += k;
}
int main(void)
{
  scanf("%d %d", &n, &m);
  for(int i = 1; i <= n; i++)
  {
    int tmp;
    scanf("%d", &tmp);  UpDate(i, tmp);
  }
  for(int i = 1; i <= m; i++)
  {
    int type, x, y;
    scanf("%d %d %d", &type, &x, &y);
    if(type == 1) UpDate(x, y);
    else      printf("%d\n", Query(y)-Query(x-1));
  }
  return 0;
}
```

1.4 树状数组 -区间修改 + 询问 (Luogu3368)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 1000005
using namespace std;
int n, m, last, c[maxn];
int lowbit(int k){
  return k&(-k);
}
int Query(int k){
  int ans = 0;
  for(int i = k; i; i -= lowbit(i)) ans += c[i];
  return ans;
}
void UpDate(int x, int k){
  for(int i = x; i <= n; i += lowbit(i))  c[i] += k;
}
int main(void)
{
  scanf("%d %d", &n, &m);
  for(int i = 1; i <= n; i++)
  {
    int tmp;
    scanf("%d", &tmp);
    UpDate(i, tmp-last), last = tmp;
  }
  for(int i = 1; i <= m; i++)
  {
    int type, x, y, k;
    scanf("%d", &type);
    if(type == 1)
    {
      scanf("%d %d %d", &x, &y, &k);
      UpDate(x, k);
      UpDate(y+1, -k);
    }
    else
    {
      scanf("%d", &x);
      printf("%d\n", Query(x));
    }
  }
  return 0;
}
```

1.5 树状数组 -逆序对 (Luogu1908)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define maxn 1000005
#define int long long
using namespace std;
int n, ans, cnt = 1;
int b[maxn], tree[maxn];
struct Node{
    int val;
    int pos;
}a[maxn];
int calc(int k){
    int res = 0;
    while(k)
    {
        res += tree[k];
        k -= k&(-k);
    }
    return res;
}
void Add(int k, int num){
    while(k <= n)
    {
        tree[k] += num;
        k += k&(-k);
    }
}
bool cmp(const Node &A, const Node &B){
    return A.val < B.val;
}
signed main(void)
{
    scanf("%lld", &n);
    for(int i = 1; i <= n; i++)
        scanf("%lld", &a[i].val), a[i].pos = i;
    sort(a+1, a+n+1, cmp);
    for(int i = 1; i <= n; i++)
    {
        if(i != 1 && a[i].val != a[i-1].val)  cnt++;
        b[a[i].pos] = cnt;
    }
    for(int i = 1; i <= n; i++)
    {
        Add(b[i], 1);
        ans += (i-calc(b[i]));
    }
    printf("%lld\n", ans);
}
```

1.6 线段树 -区间加 + 询问 (Luogu3372)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 400005
#define int long long
using namespace std;
int n, m, num[maxn];
struct SegTree{
  int left;
  int right;
  int val;
  int lazy_sum;
}tree[maxn];
void Build(int x, int left, int right){
  tree[x].left = left;
  tree[x].right = right;
  if(left == right)
  {
    tree[x].val = num[left];
    return;
  }
  int mid = (left+right)>>1;
  Build((x<<1), left, mid);
  Build((x<<1)+1, mid+1, right);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
void PushDown(int x){
  int mid = (tree[x].left+tree[x].right)>>1;
  tree[(x<<1)].val += (mid-tree[x].left+1)*tree[x].lazy_sum;
  tree[(x<<1)+1].val += (tree[x].right-mid)*tree[x].lazy_sum;
  tree[(x<<1)].lazy_sum += tree[x].lazy_sum;
  tree[(x<<1)+1].lazy_sum += tree[x].lazy_sum;
  tree[x].lazy_sum = 0;
}
void Add(int x, int left, int right, int k){
  if(tree[x].left >= left && tree[x].right <= right)
  {
    tree[x].val += (tree[x].right-tree[x].left+1)*k;
    tree[x].lazy_sum += k;
    return;
  }
  if(tree[x].left > right || tree[x].right < left)
    return;
  if(tree[x].lazy_sum)  PushDown(x);
  Add((x<<1), left, right, k);
  Add((x<<1)+1, left, right, k);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
int Query(int x, int left, int right){
  if(tree[x].left >= left && tree[x].right <= right)
    return tree[x].val;
  if(tree[x].left > right || tree[x].right < left)
    return 0;
  if(tree[x].lazy_sum)  PushDown(x);
  return Query((x<<1), left, right)+Query((x<<1)+1, left, right);
}
signed main(void)
{
  scanf("%lld %lld", &n, &m);
  for(int i = 1; i <= n; i++) scanf("%lld", &num[i]);
  Build(1, 1, n);
  for(int i = 1; i <= m; i++)
  {
    int type, x, y, k;
    scanf("%lld", &type);
    if(type == 1)
    {
      scanf("%lld %lld %lld", &x, &y, &k);
      Add(1, x, y, k);
    }
    else
    {
      scanf("%lld %lld", &x, &y);
      printf("%lld\n", Query(1, x, y));
    }
  }
  return 0;
}
```

1.7 线段树 -区间乘 + 询问 (Luogu3373)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 800005
#define int long long
using namespace std;
int n, m, P, num[maxn];
struct SegTree{
  int left;
  int right;
  int val;
  int lazy_sum;
  int lazy_mult;
}tree[maxn];
void Build(int x, int left, int right){
  tree[x].left = left;
  tree[x].right = right;
  tree[x].lazy_mult = 1;
  if(left == right)
  {
    tree[x].val = num[left];
    return;
  }
  int mid = (left+right)>>1;
  Build((x<<1), left, mid);
  Build((x<<1)+1, mid+1, right);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
void PushDown(int x){
  tree[(x<<1)].lazy_sum = (tree[x].lazy_sum+tree[(x<<1)].lazy_sum*tree[x].lazy_mult)%P;
  tree[(x<<1)+1].lazy_sum = (tree[x].lazy_sum+tree[(x<<1)+1].lazy_sum*tree[x].lazy_mult)%P;
  tree[(x<<1)].lazy_mult = (tree[x].lazy_mult*tree[(x<<1)].lazy_mult)%P;
  tree[(x<<1)+1].lazy_mult = (tree[x].lazy_mult*tree[(x<<1)+1].lazy_mult)%P;
  tree[(x<<1)].val = (tree[x].lazy_mult*tree[(x<<1)].val+tree[x].lazy_sum*(tree[(x<<1)].right-tree[(x<<1)
      ].left+1))%P;
  tree[(x<<1)+1].val = (tree[x].lazy_mult*tree[(x<<1)+1].val+tree[x].lazy_sum*(tree[(x<<1)+1].right-tree
      [(x<<1)+1].left+1))%P;
  tree[x].lazy_sum = 0;
  tree[x].lazy_mult = 1;
}
void Add(int x, int left, int right, int k){
  PushDown(x);
  if(tree[x].left >= left && tree[x].right <= right)
  {
    tree[x].val += (tree[x].right-tree[x].left+1)*k;
    tree[x].lazy_sum += k;
    return;
  }
  if(tree[x].left > right || tree[x].right < left)
    return;
  Add((x<<1), left, right, k);
  Add((x<<1)+1, left, right, k);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
void Mult(int x, int left, int right, int k){
  PushDown(x);
  if(tree[x].left >= left && tree[x].right <= right)
  {
    tree[x].lazy_sum = (tree[x].lazy_sum*k)%P;
    tree[x].lazy_mult = (tree[x].lazy_mult*k)%P;
    tree[x].val = (tree[x].val*tree[x].lazy_mult)%P;
    return;
  }
  if(tree[x].left > right || tree[x].right < left)
    return;
  Mult((x<<1), left, right, k);
    Mult((x<<1)+1, left, right, k);
    tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
int Query(int x, int left, int right){
  PushDown(x);
  if(tree[x].left >= left && tree[x].right <= right)
    return tree[x].val%P;
  if(tree[x].left > right || tree[x].right < left)
    return 0;
  return Query((x<<1), left, right)+Query((x<<1)+1, left, right);
}
signed main(void)
{
  scanf(”%lld %lld %lld”, &n, &m, &P);
  for(int i = 1; i <= n; i++) scanf(”%lld”, &num[i]);
  Build(1, 1, n);
  for(int i = 1; i <= m; i++)
  {
    int type, x, y, k;
    scanf(”%lld”, &type);
    if(type == 1)
    {
      scanf(”%lld %lld %lld”, &x, &y, &k);
      Mult(1, x, y, k);
    }
    else if(type == 2)
    {
      scanf(”%lld %lld %lld”, &x, &y, &k);
      Add(1, x, y, k);
    }
    else
    {
      scanf(”%lld %lld”, &x, &y);
      printf(”%lld\n”, Query(1, x, y)%P);
    }
  }
  return 0;
}
```

1.8 线段树 -维护区间最值 (Luogu1083)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 4000005
using namespace std;
int n, m, num[maxn];
struct tree{
  int left;
  int right;
  int minn;
  int lazy_sum;
}tree[maxn];
void Build(int x, int left, int right){
  tree[x].left = left;
  tree[x].right = right;
  if(left == right)
  {
    tree[x].minn = num[left];
    return;
  }
  int mid = (left+right)>>1;
  Build((x<<1), left, mid);
  Build((x<<1)+1, mid+1, right);
  tree[x].minn = min(tree[(x<<1)].minn, tree[(x<<1)+1].minn);
}
void PushDown(int x){
  tree[(x<<1)].minn += tree[x].lazy_sum;
  tree[(x<<1)+1].minn += tree[x].lazy_sum;
  tree[(x<<1)].lazy_sum += tree[x].lazy_sum;
  tree[(x<<1)+1].lazy_sum += tree[x].lazy_sum;
  tree[x].lazy_sum = 0;
}
void UpDate(int x, int left, int right, int k){
  if(tree[x].left >= left && tree[x].right <= right)
  {
    tree[x].minn += k;
    tree[x].lazy_sum += k;
    return;
  }
  if(tree[x].left > right || tree[x].right < left)
    return;
  if(tree[x].lazy_sum)  PushDown(x);
  UpDate((x<<1), left, right, k);
  UpDate((x<<1)+1, left, right, k);
  tree[x].minn = min(tree[(x<<1)].minn, tree[(x<<1)+1].minn);
}
int main(void)
{
  scanf("%d %d", &n, &m);
  for(int i = 1; i <= n; i++)   scanf("%d", &num[i]);
  Build(1, 1, n);
  for(int i = 1; i <= m; i++)
  {
    int x, y, k;
    scanf("%d %d %d", &k, &x, &y);
    UpDate(1, x, y, -k);
    if(tree[1].minn < 0)
    {
      printf("%d\n%d", -1, i);
      return 0;
    }
  }
  printf("%d\n", 0);
  return 0;
}
```

1.9 树链剖分 -LCA(Luogu3379)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 1000005
using namespace std;
int n, m, s;
int k, head[maxn];
int son[maxn], dep[maxn], top[maxn], siz[maxn], fat[maxn];
struct Edge{
  int to;
  int next;
}Edge[maxn];
void Build(int u, int v){
  Edge[k].to = v;
  Edge[k].next = head[u];
  head[u] = k++;
}
void dfs1(int x, int father, int deep){
  siz[x] = 1;
  dep[x] = deep;
  fat[x] = father;
  int maxson = −1;
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(father == Edge[i].to)  continue;
    dfs1(Edge[i].to, x, deep+1);
    siz[x] += siz[Edge[i].to];
    if(siz[Edge[i].to] > maxson)
      son[x] = Edge[i].to, maxson = siz[Edge[i].to];
  }
}
void dfs2(int x, int tp){
  top[x] = tp;
  if(son[x] == 0)   return;
  dfs2(son[x], tp);
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(Edge[i].to == fat[x] || Edge[i].to == son[x])  continue;
    dfs2(Edge[i].to, Edge[i].to);
  }
}
int LCA(int x, int y){
  while(top[x] != top[y])
    dep[top[x]] >= dep[top[y]] ? x = fat[top[x]] : y = fat[top[y]];
  return dep[x] < dep[y] ? x : y;
}
int main(void)
{
  memset(head, −1, sizeof(head));
  scanf(”%d %d %d”, &n, &m, &s);
  for(int i = 1; i <= n−1; i++)
  {
    int u, v;
    scanf(”%d %d”, &u, &v);
    Build(u, v);  Build(v, u);
  }
  dfs1(s, 0, 1);
  dfs2(s, s);
  for(int i = 1; i <= m; i++)
  {
    int p1, p2;
    scanf("%d %d", &p1, &p2);
    printf("%d\n", LCA(p1, p2));
  }
  return 0;
}
```

1.10 树链剖分 -线段树维护 (Luogu3384)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define maxn 800005
#define int long long
using namespace std;
int N, M, R, P;
int k, head[maxn];
int nw[maxn], w[maxn];
int ID[maxn], son[maxn], siz[maxn], top[maxn], fat[maxn], dep[maxn], cnt;
struct Edge{
  int to;
  int next;
}Edge[maxn];
struct STree{
  int left;
  int right;
  int val;
  int lazy;
}tree[maxn];
void Build(int u, int v){
  Edge[k].to = v;
  Edge[k].next = head[u];
  head[u] = k++;
}
void Build_STree(int x, int left, int right){
  tree[x].left = left;
  tree[x].right = right;
  if(left == right)
  {
    tree[x].val = nw[left];
    return;
  }
  int mid = (left+right)>>1;
  Build_STree((x<<1), left, mid);
  Build_STree((x<<1)+1, mid+1, right);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
void PushDown(int x){
  int mid = (tree[x].left+tree[x].right)>>1;
  tree[(x<<1)].val += tree[x].lazy*(mid-tree[x].left+1);
  tree[(x<<1)+1].val += tree[x].lazy*(tree[x].right-mid);
  tree[(x<<1)].lazy += tree[x].lazy;
  tree[(x<<1)+1].lazy += tree[x].lazy;
  tree[x].lazy = 0;
}
void UpDate(int x, int left, int right, int k){
  if(tree[x].left >= left && tree[x].right <= right)
  {
    tree[x].val += (tree[x].right-tree[x].left+1)*k;
    tree[x].lazy += k;
    return;
  }
  if(tree[x].left > right || tree[x].right < left)
```

```
        return;
  if(tree[x].lazy)  PushDown(x);
  UpDate((x<<1), left, right, k);
  UpDate((x<<1)+1, left, right, k);
  tree[x].val = tree[(x<<1)].val+tree[(x<<1)+1].val;
}
int Query(int x, int left, int right){
  if(tree[x].left >= left && tree[x].right <= right)
    return tree[x].val;
  if(tree[x].left > right || tree[x].right < left)
        return 0;
  if(tree[x].lazy)  PushDown(x);
  return Query((x<<1), left, right)+Query((x<<1)+1, left, right);
}
void dfs1(int x, int father, int deep){
  siz[x] = 1;
  dep[x] = deep;
  fat[x] = father;
  int maxson = -1;
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(Edge[i].to == father)  continue;
    dfs1(Edge[i].to, x, deep+1);
    siz[x] += siz[Edge[i].to];
    if(siz[Edge[i].to] > maxson)
      son[x] = Edge[i].to, maxson = siz[Edge[i].to];
  }
}
void dfs2(int x, int tp){
  top[x] = tp;
  ID[x] = ++cnt;
  nw[cnt] = w[x];
  if(son[x] == 0)   return;
  dfs2(son[x], tp);
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(Edge[i].to == fat[x] || Edge[i].to == son[x])  continue;
    dfs2(Edge[i].to, Edge[i].to);
  }
}
void UpRange(int x, int y, int k){
  while(top[x] != top[y])
  {
    if(dep[top[x]] < dep[top[y]]) swap(x, y);
    UpDate(1, ID[top[x]], ID[x], k);
    x = fat[top[x]];
  }
  if(dep[x] > dep[y])   swap(x, y);
  UpDate(1, ID[x], ID[y], k);
}
int QRange(int x, int y){
  int res = 0;
  while(top[x] != top[y])
  {
    if(dep[top[x]] < dep[top[y]]) swap(x, y);
    res += Query(1, ID[top[x]], ID[x]);
    x = fat[top[x]];
  }
  if(dep[x] > dep[y])   swap(x, y);
  res += Query(1, ID[x], ID[y]);
  return res;
}
void UpSon(int x, int k){
  UpDate(1, ID[x], ID[x]+siz[x]-1, k);
}
int QSon(int x){
  return Query(1, ID[x], ID[x]+siz[x]-1);
}
signed main(void)
{
  memset(head, -1, sizeof(head));
  scanf("%lld %lld %lld %lld", &N, &M, &R, &P);
  for(int i = 1; i <= N; i++)   scanf("%lld", &w[i]);
  for(int i = 1; i <= N-1; i++)
  {
    int u, v;
    scanf("%lld %lld", &u, &v);
    Build(u, v);
    Build(v, u);
  }
  dfs1(R, 0, 1);
  dfs2(R, R);
  Build_STree(1, 1, N);
  while(M--)
  {
    int type, x, y, k;
    scanf("%lld", &type);
    switch(type)
    {
      case 1:
      {
        scanf("%lld %lld %lld", &x, &y, &k);
        UpRange(x, y, k);
        break;
      }
      case 2:
      {
        scanf("%lld %lld", &x, &y);
        printf("%lld\n", QRange(x, y)%P);
        break;
      }
      case 3:
      {
        scanf("%lld %lld", &x, &k);
        UpSon(x, k);
        break;
      }
      case 4:
      {
        scanf("%lld", &x);
        printf("%lld\n", QSon(x)%P);
        break;
      }
    }
  }
  return 0;
}
```

2 图论算法

2.1 单源最短路 -Dij+ 堆优化 (Luogu4779)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#include <queue>
#define maxn 200005
using namespace std;
int N, M, S;
int k, dis[maxn], head[maxn];
struct Edge{
  int to;
  int next;
  int weight;
}Edge[maxn];
struct Node{
  int now;
  int dis;
};
bool operator < (const Node &A, const Node &B){
  return A.dis > B.dis;
}
void Build(int u, int v, int w){
  Edge[k].to = v;
  Edge[k].weight = w;
  Edge[k].next = head[u];
  head[u] = k++;
}
priority_queue<Node> q1;
void Dijkstra(int x){
  dis[x] = 0;
  q1.push((Node){x, 0});
  while(!q1.empty())
  {
    Node tmp = q1.top();  q1.pop();
    int now = tmp.now, dist = tmp.dis;
    if(dist != dis[now])  continue;
    for(int i = head[now]; ~i; i = Edge[i].next)
    {
      if(dis[Edge[i].to] > dis[now]+Edge[i].weight)
      {
        dis[Edge[i].to] = dis[now]+Edge[i].weight;
        q1.push((Node){Edge[i].to, dis[Edge[i].to]});
      }
    }
  }
}
int main(void)
{
  memset(dis, 0x3f, sizeof(dis));
  memset(head, −1, sizeof(head));
  scanf(”%d %d %d”, &N, &M, &S);
  for(int i = 1; i <= M; i++)
  {
    int u, v, w;
    scanf("%d %d %d", &u, &v, &w);
    Build(u, v, w);
  }
  Dijkstra(S);
  for(int i = 1; i <= N; i++) printf("%d ", dis[i]);
  return 0;
}
```

2.2 强联通分量 -Tarjan(Luogu2746)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 100005
using namespace std;
int N;
int k, head[maxn];
int top, stk[maxn];
int col, dfn[maxn], low[maxn], color[maxn];
int tot, sumin, sumout, in[maxn], out[maxn];
struct Edge{
    int to;
    int next;
}Edge[maxn];
void Build(int u, int v){
    Edge[k].to = v;
    Edge[k].next = head[u];
    head[u] = k++;
}
void Tarjan(int x){
    dfn[x] = low[x] = ++tot;
    stk[++top] = x;
    for(int i = head[x]; ~i; i = Edge[i].next)
    {
        if(dfn[Edge[i].to] == 0)
        {
            Tarjan(Edge[i].to);
            low[x] = min(low[x], low[Edge[i].to]);
        }
        else if(color[Edge[i].to] == 0)
            low[x] = min(low[x], dfn[Edge[i].to]);
    }
    if(low[x] == dfn[x])
    {
        color[x] = ++col;
        while(stk[top] != x)
        {
            color[stk[top]] = col;
            top--;
        }
        top--;
    }
}
int main(void)
{
    memset(head, -1, sizeof(head));
    scanf("%d", &N);
    for(int i = 1; i <= N; i++)
    {
        int tmp;
        while(scanf("%d", &tmp), tmp != 0)  Build(i, tmp);
    }
    for(int i = 1; i <= N; i++)
        if(dfn[i] == 0)   Tarjan(i);
    for(int i = 1; i <= N; i++)
    {
    for(int j = head[i]; ~j; j = Edge[j].next)
        {
            if(color[i] != color[Edge[j].to])
            {
                in[color[Edge[j].to]]++;
                out[color[i]]++;
            }
        }
  }
    for(int i = 1; i <= col; i++)
    {
        sumin += (in[i] == 0) ? 1 : 0;
        sumout += (out[i] == 0) ? 1 : 0;
    }
    if(col == 1)
    {
        printf("%d\n%d", 1, 0);
        return 0;
    }
    printf("%d\n%d", sumin, max(sumin, sumout));
    return 0;
}
```

2.3 强联通分量 -Kosaraju(Luogu2863)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define maxn 100005
using namespace std;
int n, m, ans;
int cnt, d[maxn], used[maxn];
int k1, k2, head1[maxn], head2[maxn];
struct Edge{
    int to;
    int next;
}G1[maxn], G2[maxn];
void Build(int u, int v){
    G1[k1].to = v;
    G1[k1].next = head1[u];
    head1[u] = k1++;
    G2[k2].to = u;
    G2[k2].next = head2[v];
    head2[v] = k2++;
}
void dfs1(int x){
    used[x] = 1;
    for(int i = head1[x]; ~i; i = G1[i].next)
        if(used[G1[i].to] == 0) dfs1(G1[i].to);
    d[++cnt] = x;
}
void dfs2(int x){
    cnt++;
    used[x] = 1;
    for(int i = head2[x]; ~i; i = G2[i].next)
        if(used[G2[i].to] == 0) dfs2(G2[i].to);
}
void Kosaraju(){
    cnt = 0;
    memset(used, 0, sizeof(used));
    for(int i = 1; i <= n; i++)
        if(used[i] == 0)  dfs1(i);
    memset(used, 0, sizeof(used));
    for(int i = n; i >= 1; i−−)
        if(used[d[i]] == 0) cnt = 0, dfs2(d[i]), cnt != 1 ? ans++ : 0;
}
void init(){
    memset(head1, −1, sizeof(head1));
    memset(head2, −1, sizeof(head2));
}
int main(void)
{
    init();
    scanf(”%d %d”, &n, &m);
    for(int i = 1; i <= m; i++)
    {
        int u, v;
        scanf(”%d %d”, &u, &v);
        Build(u, v);
    }
    Kosaraju();
    printf("%d\n", ans);
    return 0;
}
```

2.4 强联通分量 -稳定婚姻 (Luogu1407)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <map>
#define maxn 10005
using namespace std;
int n, m;
int k, head[maxn];
int idx, cnt, top, dfn[maxn], low[maxn], stk[maxn], ins[maxn], col[maxn];
map<string, int> Name_To_Number;
struct Edge{
  int to;
  int next;
}Edge[maxn];
void Build(int u, int v){
  Edge[k].to = v;
  Edge[k].next = head[u];
  head[u] = k++;
}
void Tarjan(int x){
  dfn[x] = low[x] = ++idx;
  stk[++top] = x;
  ins[x] = 1;
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(!dfn[Edge[i].to])
    {
      Tarjan(Edge[i].to);
      low[x] = min(low[x], low[Edge[i].to]);
    }
    else if(ins[Edge[i].to])
      low[x] = min(low[x], dfn[Edge[i].to]);
  }
  if(low[x] == dfn[x])
  {
    ++cnt;
    do{
      col[stk[top]] = cnt;
      ins[stk[top]] = 0;
    }while(stk[top--] != x);
  }
}
int main(void)
{
  memset(head, -1, sizeof(head));
  scanf("%d", &n);
  string gir, boy;
  for(int i = 1; i <= n; i++)
  {
    cin >> gir >> boy;
    Name_To_Number[gir] = i;
    Name_To_Number[boy] = i+n;
    Build(i, i+n);
  }
  scanf("%d", &m);
  for(int i = 1; i <= m; i++)
  {
    cin >> gir >> boy;
    Build(Name_To_Number[boy], Name_To_Number[gir]);
  }
  for(int i = 1; i <= n*2; i++)
    if(dfn[i] == 0)    Tarjan(i);
  for(int i = 1; i <= n; i++)
    printf(col[i] == col[i+n] ? ”Unsafe\n” : ”Safe\n”);
  return 0;
}
```

2.5 二分图匹配 -匈牙利算法 (Luogu3386)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 400005
using namespace std;
int n, m, e;
int k, head[maxn];
int ans, vis[maxn], link[maxn];
struct Edge{
  int to;
  int next;
}Edge[maxn];
void Build(int u, int v){
  Edge[k].to = v;
  Edge[k].next = head[u];
  head[u] = k++;
}
bool dfs(int x){
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(vis[Edge[i].to] == 0)
    {
      vis[Edge[i].to] = 1;
      if(!link[Edge[i].to] || dfs(link[Edge[i].to]))
      {
        link[Edge[i].to] = x;
        return 1;
      }
    }
  }
  return 0;
}
int main(void)
{
  memset(head, −1, sizeof(head));
  scanf(”%d %d %d”, &n, &m, &e);
  for(int i = 1; i <= e; i++)
  {
    int u, v;
    scanf(”%d %d”, &u, &v);
    if(u > n || v > m)  continue;
    Build(u, v);
  }
  for(int i = 1; i <= n; i++)
  {
    memset(vis, 0, sizeof(vis));
    if(dfs(i))  ans++;
  }
  printf(”%d\n”, ans);
  return 0;
}
```

2.6 二分图匹配 -网络流 (Luogu3386)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <queue>
#define maxn 800005
using namespace std;
int s, t, n, m, e;
int k, head[maxn];
int level[maxn];
struct Edge{
  int to;
  int next;
  int flow;
  int cap;
}Edge[maxn];
void Build(int u, int v, int w){
  Edge[k].to = v;
  Edge[k].cap = w;
  Edge[k].flow = 0;
  Edge[k].next = head[u];
  head[u] = k++;
}
bool bfs(){
  queue<int> q1;
  memset(level, 0, sizeof(level));
  level[s] = 1; q1.push(s);
  while(!q1.empty())
  {
    int tmp = q1.front(); q1.pop();
    for(int i = head[tmp]; ~i; i = Edge[i].next)
      if(Edge[i].cap > Edge[i].flow && !level[Edge[i].to])
        level[Edge[i].to] = level[tmp]+1, q1.push(Edge[i].to);
  }
  return level[t];
}
int dfs(int x, int flow){
  int nowflow = 0;
  if(x == t)  return flow;
  for(int i = head[x]; ~i; i = Edge[i].next)
  {
    if(flow <= nowflow)   break;
    if(level[Edge[i].to] == level[x]+1)
    {
      int mflow = dfs(Edge[i].to, min(Edge[i].cap-Edge[i].flow, flow-nowflow));
      Edge[i].flow += mflow;
      Edge[i^1].flow -= mflow;
      nowflow += mflow;
    }
  }
  return nowflow;
}
int maxflow(){
  int ans = 0;
  while(bfs())  ans += dfs(s, (1<<30));
  return ans;
```

```c
}
int main(void)
{
  memset(head, −1, sizeof(head));
  scanf(”%d %d %d”, &n, &m, &e);
  s = 0, t = n+m+1;
  for(int i = 1; i <= n; i++)
    Build(s, i, 1), Build(i, s, 0);
  for(int i = n+1; i <= n+m; i++)
    Build(i, t, 1), Build(t, i, 0);
  for(int i = 1; i <= e; i++)
  {
    int u, v;
    scanf(”%d %d”, &u, &v);
    if(u > n || v > m)  continue;
    Build(u, v+n, 1), Build(v+n, u, 0);
  }
  printf(”%d\n”, maxflow());
  return 0;
}
```

2.7 二分图匹配 -KM 算法 (HDU2255)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 1005
using namespace std;
int n, ans;
int WEI[maxn][maxn];
int Lx[maxn], Ly[maxn];
int Left[maxn], S[maxn], T[maxn];
void init(){
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
      scanf("%d", &WEI[i][j]);
  ans = 0;
}
bool match(int x){
  S[x] = 1;
  for(int j = 1; j <= n; j++)
  {
    if(Lx[x]+Ly[j] == WEI[x][j] && !T[j])
    {
      T[j] = 1;
      if(!Left[j] || match(Left[j]))
      {
        Left[j] = x;
        return true;
      }
    }
  }
  return false;
}
void UpDate(){
  int tmp = (1<<30);
  for(int i = 1; i <= n; i++)
  {
    if(S[i])
    {
      for(int j = 1; j <= n; j++)
        if(!T[j]) tmp = min(tmp, Lx[i]+Ly[j]-WEI[i][j]);
    }
  }
  for(int i = 1; i <= n; i++)
  {
    if(S[i])  Lx[i] -= tmp;
    if(T[i])  Ly[i] += tmp;
  }
}
void KM(){
  for(int i = 1; i <= n; i++)
  {
    Left[i] = Lx[i] = Ly[i] = 0;
    for(int j = 1; j <= n; j++)
      Lx[i] = max(Lx[i], WEI[i][j]);
  }
  for(int i = 1; i <= n; i++)
  {
    while(1)
    {
      for(int j = 1; j <= n; j++)   S[j] = T[j] = 0;
      if(match(i))  break;
      else      UpDate();
    }
  }
}
int main(void)
{
  while(~scanf(”%d”, &n))
  {
    init();
    KM();
    for(int i = 1; i <= n; i++) ans += WEI[Left[i]][i];
    printf(”%d\n”, ans);
  }
  return 0;
}
```

3　字符串算法

3.1　字符串匹配 -KMP 算法 (Luogu3375)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 100005
using namespace std;
int len1, len2, next[maxn];
char str1[maxn], str2[maxn];
void GetNext(){
  int now = 0, k = -1; next[0] = -1;
  while(now < len2)
    k == -1 || str2[now] == str2[k] ? next[++now] = ++k : k = next[k];
}
void KMP(){
  int i = 0, j = 0;
  while(i < len1)
  {
    j == -1 || str1[i] == str2[j] ? i++, j++ : j = next[j];
    if(j == len2)
      printf("%d\n", i-len2+1), j = next[j];
  }
}
int main(void)
{
  scanf("%s", str1);
  scanf("%s", str2);
  len1 = strlen(str1);
  len2 = strlen(str2);
  GetNext();
  KMP();
  for(int i = 1; i <= len2; i++)  printf("%d ", next[i]);
  return 0;
}
```

3.2　最小循环节 -KMP 算法 (HDU3476)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 1000005
using namespace std;
int len, next2[maxn];
char string_B[maxn];
void Getnext2(){
  int now = 0, k = −1; next2[0] = −1;
  while(now < len)
    k == −1 || string_B[now] == string_B[k] ? next2[++now] = ++k : k = next2[k];
}
int main(void)
{
  int T;  scanf("%d", &T);
  while(T−−)
  {
    scanf("%s", string_B);
    len = strlen(string_B);
    Getnext2();
    int tmp = len−next2[len];
    if(len%tmp == 0)
      printf("%d\n", len == tmp ? tmp : 0);
    else
      printf("%d\n", tmp−(next2[len]%tmp));
  }
  return 0;
}
```

3.3 Trie-前缀匹配 (Luogu2580)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define maxn 1000005
using namespace std;
int n, m, cnt;
int k, head[maxn];
char str[maxn];
struct Edge{
    int to;
    int next;
}Edge[maxn];
struct Trie{
    char x;
    bool vis, end;
}Trie[maxn];
void Build(int u, int v){
    Edge[k].to = v;
    Edge[k].next = head[u];
    head[u] = k++;
}
void Insert(){
    int flag, pos = 0, len = strlen(str);
    for(int i = 0; i <= len-1; i++)
    {
        flag = 0;
        for(int j = head[pos]; ~j; j = Edge[j].next)
        {
      if(Trie[Edge[j].to].x == str[i])
            {
                pos = Edge[j].to;
                flag = 1;
                break;
            }
      }
      if(flag == 0)
          Build(pos, ++cnt), pos = cnt, Trie[pos].x = str[i];
      if(i == len-1)
          Trie[pos].end = 1;
  }
}
int Query(){
    int flag, pos = 0, len = strlen(str);
    for(int i = 0; i <= len-1; i++)
    {
        flag = 0;
        for(int j = head[pos]; ~j; j = Edge[j].next)
        {
      if(Trie[Edge[j].to].x == str[i])
            {
                pos = Edge[j].to;
                flag = 1;
                break;
            }
      }
        if(flag == 0)   return 3;
        if(i == len-1)
        {
            if(Trie[pos].end)
            {
                if(Trie[pos].vis == 0)
                {
                    Trie[pos].vis = 1;
                    return 1;
                }
                else  return 2;
            }
            else  return 3;
        }
    }
    return 0;
}
int main(void)
{
    memset(head, -1, sizeof(head));
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        scanf("%s", str);
        Insert();
    }
    scanf("%d", &m);
    for(int i = 1; i <= m; i++)
    {
        scanf("%s", str);
        int type = Query();
        if(type == 1)
            printf("OK\n");
        else
            printf(type == 3 ? "WRONG\n" : "REPEAT\n");
    }
    return 0;
}
```

3.4 最长回文串 -manacher 算法 (Luogu3805)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 100000005
using namespace std;
int len, ans, P[maxn];
char str[maxn], nstr[maxn];
void MakeStr(){
    nstr[0] = nstr[1] = '#';
    for(int i = 0; i < len; i++)
        nstr[(i<<1)+2] = str[i], nstr[(i<<1)+3] = '#';
    len = (len<<1)+2;
    nstr[len] = 0;
}
void manacher(){
    int mx = 0, mid = -1;
    for(int i = len; nstr[i] != 0; i++) nstr[i] = 0;
    for(int i = 1; i < len; i++)
    {
        mx > i ? P[i] = min(P[(mid<<1)-i], P[mid]+mid-i) : P[i] = 1;
        while(nstr[i+P[i]] == nstr[i-P[i]])   P[i]++;
        if(P[i]+i > mx)   mx = P[i]+i, mid = i;
    }
}
int main(void)
{
    scanf("%s", str); len = strlen(str);
    MakeStr();
    manacher();
    for(int i = 0; i < len; i++)  ans = max(ans, P[i]);
    printf("%d\n", ans-1);
    return 0;
}
```

4　数论算法

4.1　素数判定 -Miller_Rabin(Luogu3383)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cstdlib>
#define int long long
using namespace std;
int n, r;
int QuickPow(int a, int b, int p){
    if(b == 0)  return 1;
    int tmp = QuickPow(a, b>>1, p);
    tmp = (tmp*tmp)%p;
    if(b&1) tmp = (tmp*a)%p;
    return tmp%p;
}
int QuickMult(int a, int b, int p){
    int ans = 0;
    while(b)
    {
        if(b&1)    ans = (ans+a)%p;
        a = (a+a)%p;
        b >>= 1;
    }
    return ans;
}
bool Miller_Rabin(int x){
    int s = 10;
    int u = x-1, t = 0;
    if(x == 2)         return true;
    if(!(x&1) || x < 2)    return false;
    while(!(u&1))
    {
        t++;
        u >>= 1;
    }
    while(s--)
    {
        int a = rand()%(x-1)+1;
        int b = QuickPow(a, u, x);
        for(int j = 0; j < t; j++)
        {
            int y = QuickMult(b, b, x);
            if(y == 1 && b != 1 && b != x-1)  return false;
            b = y;
        }
        if(b != 1)     return false;
    }
    return true;
}
signed main(void)
{
    scanf("%lld %lld", &r, &n);
    for(int i = 1; i <= n; i++)
    {
        int tmp;  scanf("%lld", &tmp);
        printf(Miller_Rabin(tmp) ? "Yes\n" : "No\n");
    }
    return 0;
}
```

4.2 逆元 -扩展欧几里德 (Luogu1082)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define int long long
using namespace std;
void exgcd(int a, int b, int &x, int &y){
    if(b == 0)    x = 1, y  = 0;
    else
    {
        exgcd(b, a%b, y, x);
        y −= a/b*x;
    }
}
signed main(void)
{
    int a, b, x, y;
    scanf(”%lld %lld %lld %lld”, &a, &b, &x, &y);
    exgcd(a, b, x, y);
    while(x < 0)  x += b;
    printf(”%lld\n”, x);
    return 0;
}
```

4.3 逆元 -线性求逆 (Luogu3811)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 3000005
using namespace std;
int inv[maxn], n, p;
int main(void)
{
    scanf("%d %d", &n, &p);
    inv[1] = 1; printf("%d\n", inv[1]);
    for(int i = 2; i <= n; i++)
    {
        inv[i] = (long long)(p-p/i)*inv[p%i]%p;
        printf("%d\n", inv[i]);
    }
    return 0;
}
```

4.4 逆元 -欧拉函数 (Luogu1082)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define int long long
using namespace std;
int Euler(int x){
    int res = x;
    for(int i = 2; i*i <= x; i++)
    {
        if(x%i == 0)    res = res/i*(i−1);
        while(x%i == 0)   x /= i;
    }
    if(x > 1) res = res/x*(x−1);
    return res;
}
int SuperPow(int a, int b, int p){
    if(b == 0)  return 1;
    int tmp = SuperPow(a, b>>1, p);
    tmp = (tmp*tmp)%p;
    if(b&1)   tmp = (tmp*a)%p;
    return tmp%p;
}
signed main(void)
{
    int a, b;
    cin >> a >> b;
    int _pow = Euler(b)−1;
    cout << SuperPow(a, _pow, b) << endl;
    return 0;
}
```

4.5 CRT&Lucas(Luogu2480)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#define maxn 40005
#define int long long
#define MOD 999911658
using namespace std;
int N, G, val;
int res[maxn], fact[maxn];
const int Prie[5] = {0, 2, 3, 4679, 35617};
void init(int P){
    fact[0] = 1;
    for(int i = 1; i <= P; i++)
        fact[i] = (fact[i-1]*i)%P;
}
int QuickPow(int a, int b, int p){
    if(b == 0)  return 1;
    int tmp = QuickPow(a, b>>1, p);
    tmp = (tmp*tmp)%p;
    if(b&1) tmp = (tmp*a)%p;
    return tmp%p;
}
int Com(int n, int m, int p){
    if(n < m) return 0;
    return ((fact[n]*QuickPow(fact[m], p-2, p))%p*(QuickPow(fact[n-m], p-2, p))%p)%p;
}
int Lucas(int n, int m, int p){
    if(n < m) return 0;
    if(n == 0)  return 1;
    return Lucas(n/p, m/p, p)*Com(n%p, m%p, p)%p;
}
void CRT(){
    for(int k = 1; k <= 4; k++)
        val = (val+res[k]*(MOD/Prie[k])%MOD*QuickPow(MOD/Prie[k], Prie[k]-2, Prie[k]))%MOD;
}
signed main(void)
{
    scanf("%lld %lld", &N, &G);
    if(G%(MOD+1) == 0)
    {
        printf("%d\n", 0);
        return 0;
    }
    for(int k = 1; k <= 4; k++)
    {
        init(Prie[k]);
        for(int i = 1; i*i <= N; i++)
        {
            if(N%i == 0)
            {
                res[k] = (res[k]+Lucas(N, i, Prie[k]))%Prie[k];
                if(i*i != N)
                    res[k] = (res[k]+Lucas(N, N/i, Prie[k]))%Prie[k];
            }
        }
    }
    CRT();
    printf(”%lld\n”, QuickPow(G, val, MOD+1));
    return 0;
}
```

4.6 矩阵乘法 + 快速幂 (Luogu3390)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 105
#define int long long
#define MOD 1000000007
using namespace std;
int n, k;
struct Matrix{
    int Board[maxn][maxn];
    Matrix(){
      memset(Board, 0, sizeof(Board));
    }
    void Build(){
        for(int i = 1; i <= n; i++) Board[i][i] = 1;
    }
}NowMatrix;
Matrix operator *(const Matrix &A, const Matrix &B){
    Matrix res;
    for(int i = 1; i <= n; i++)
      for(int j = 1; j <= n; j++)
        for(int k = 1; k <= n; k++)
                res.Board[i][j] = (res.Board[i][j]+A.Board[i][k]*B.Board[k][j])%MOD;
    return res;
}
Matrix QuickPow(Matrix A, int k){
    Matrix res; res.Build();
    while(k)
    {
        if(k&1)   res = res*A;
        A = A*A;
        k >>= 1;
    }
    return res;
}
signed main(void)
{
    scanf(”%lld %lld”, &n, &k);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            scanf(”%lld”, &NowMatrix.Board[i][j]);
    NowMatrix = QuickPow(NowMatrix, k);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            printf(j == n ? ”%lld\n” : ”%lld ”, NowMatrix.Board[i][j]);
    return 0;
}
```

5 卡常骗分

5.1 手写 STL+ 快速读入 (Luogu2827)

```cpp
#include <iostream>
#include <cstring>
#include <cstdio>
#include <algorithm>
#define re register
#define maxn 7000005
//将多次调用的一些变量扔到寄存器以加速
using namespace std;
int n, m, q, u, v, t, tmp[200010];
//手写队列，不用STL
struct queue{
    int left, right, q[maxn];
    void pop()      {left++;}
    int front()     {return q[left];}
    void push(int x)  {q[++right] = x;}
    bool empty()    {return left > right;}
}p[3];
//最简单的一种快速读入
//快读不仅可以快速读入，还对一些不方便格式化读入的数据有奇效
int read(){
    int x = 0,f = 1;  char ch = getchar();
    while (ch < '0' || ch > '9')    {if(ch == '-') f = -1; ch = getchar();}
    while (ch >= '0' && ch <= '9')    {x = x*10+ch-'0';      ch = getchar();}
    return x*f;
}
//利用缓冲区一段段读入的写法，上面是一个个字符读入的写法
/*
char buf[maxn], *p1 = buf, *p2 = buf;
inline char fred(){
    return p1 == p2 && (p2 = (p1 = buf)+fread(buf, 1, 100, stdin), p1 == p2) ? EOF : *p1++;
}
inline int read(){
    char ch = fred(); int sum = 0;
    while(ch < '0' || ch > '9')   ch = fred();
    while(ch >= '0' && ch <= '9') sum = (sum<<3)+(sum<<1)+ch-'0', ch = fred();
    return sum;
}
*/
bool cmp(const int &A, const int &B){
    return A > B;
}
//内联函数，使得其加速
//相当于将这块代码粘贴到调用的地方，对于太长的代码和递归太深的代码效用较小
inline int Choose(){
    int res = -(1<<30), pos = -1;
    if(!p[0].empty() && res < p[0].front()) res = p[0].front(), pos = 0;
    if(!p[1].empty() && res < p[1].front()) res = p[1].front(), pos = 1;
    if(!p[2].empty() && res < p[2].front()) res = p[2].front(), pos = 2;
    p[pos].pop();
    return res;
}
int main(void)
```

```
{
    n = read(); m = read(); q = read();
    u = read(); v = read(); t = read();
    for(re int i = 1; i <= n; i++)    tmp[i] = read();
    sort(tmp+1, tmp+n+1, cmp);
    for(re int i = 1; i <= n; i++)    p[0].push(tmp[i]);
    p[0].left = p[1].left = p[2].left = 1;
    int lazy = 0;
    for(re int i = 1; i <= m; i++)
    {
        int maxx = Choose();
        if(i%t == 0)  printf(i+t>m ? "%d" : "%d ", maxx+lazy);
        p[1].push((long long)(maxx+lazy)*u/v-lazy-q);
        p[2].push((maxx+lazy)-(long long)(maxx+lazy)*u/v-lazy-q);
        lazy += q;
    }
    putchar('\n');
    int cnt = 0;
    while(!p[0].empty() || !p[1].empty() || !p[2].empty())
    {
        cnt++;
        int maxx = Choose();
        if(cnt%t == 0)  printf(cnt+t>m+n ? "%d" : "%d ", maxx+lazy);
    }
    return 0;
}
```

5.2 手写 max(Luogu2722)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 10005
#define max(x, y) x > y ? x : y;
//对于这道题这样卡一下总共能快300ms
//注意命名不要和敏感字冲突，比如time—>time2
using namespace std;
int dp[maxn];
int time2, num;
int cost[maxn], score[maxn];
int main(void)
{
    memset(dp, 0xcf, sizeof(dp));
    scanf("%d %d", &time2, &num);
    for(int i = 1; i <= num; i++)
        scanf("%d %d", &score[i], &cost[i]);
    dp[0] = 0;
    for(int i = 1; i <= num; i++)
        for(int j = cost[i]; j <= time2; j++)
            dp[j] = max(dp[j], dp[j—cost[i]]+score[i]);
    int maxx = —(1<<30);
    for(int i = 1; i <= time2; i++)   maxx = max(maxx, dp[i]);
    printf("%d\n", maxx);
    return 0;
}
```

5.3 循环置换 (Luogu3390)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#define maxn 105
#define int long long
#define MOD 1000000007
using namespace std;
int n, k;
struct Matrix{
    int Board[maxn][maxn];
    Matrix(){
      memset(Board, 0, sizeof(Board));
    }
    void Build(){
        for(int i = 1; i <= n; i++) Board[i][i] = 1;
    }
}NowMatrix;
//利用缓冲改变循环顺序可以加快速度，请读者自己试一下
Matrix operator *(const Matrix &A, const Matrix &B){
    Matrix res;
    for(int i = 1; i <= n; i++)
        for(int k = 1; k <= n; k++)
            for(int j = 1; j <= n; j++)
                res.Board[i][j] = (res.Board[i][j]+A.Board[i][k]*B.Board[k][j])%MOD;
    return res;
}
Matrix QuickPow(Matrix A, int k){
    Matrix res; res.Build();
    while(k)
    {
        if(k&1)   res = res*A;
        A = A*A;
        k >>= 1;
    }
    return res;
}
signed main(void)
{
    scanf("%lld %lld", &n, &k);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            scanf("%lld", &NowMatrix.Board[i][j]);
    NowMatrix = QuickPow(NowMatrix, k);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            printf(j == n ? "%lld\n" : "%lld ", NowMatrix.Board[i][j]);
    return 0;
}
```

5.4 模拟退火 (Luogu1337)

```
#include <iostream>
#include <cstring>
#include <cstdio>
#include <cmath>
#include <ctime>
#include <cstdlib>
#define maxn 1005
#define Ra() ((rand()<<1)-RAND_MAX)*T
using namespace std;
int n, sx, sy;
double tx, ty;
double ans = 1e18;
const double delta = 0.993;
struct Node{
    int x;
    int y;
    int w;
}P[maxn];
double calc(double x, double y){
    double res = 0;
    for(int i = 1; i <= n; i++)
    {
        double tmpx = x-P[i].x;
        double tmpy = y-P[i].y;
        res += sqrt(tmpx*tmpx+tmpy*tmpy)*P[i].w;
    }
    return res;
}
void SA(){
    double T = 2500;
    double nowx = tx, nowy = ty;
    while(T > 1e-14)
    {
        double tmpx = nowx+Ra();
        double tmpy = nowy+Ra();
        double now = calc(tmpx, tmpy);
        double div = now-ans;
        if(div < 0)
        {
            ans = now;
            tx = tmpx;
            ty = tmpy;
        }
        else if(exp(-div/T)*RAND_MAX > rand())
        {
            nowx = tmpx;
            nowy = tmpy;
        }
        T *= delta;
    }
}
int main(void)
{
    srand(19260817);
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
    {
        scanf(”%d %d %d”, &P[i].x, &P[i].y, &P[i].w);
        sx += P[i].x; sy += P[i].y;
    }
    tx = (double)sx/n;
    ty = (double)sy/n;
    while((double)clock()/CLOCKS_PER_SEC < 0.83)  SA();
    printf(”%.3lf %.3lf”, tx, ty);
    return 0;
}
```

6 后记

这是作者第一次使用 $LaTex$ 进行文档编辑，在此首先对 $LaTex$ 的开发者以及维护的同志们表示敬意。顺便希望各位 $OIer$ 不要死记模板，模板是有用的，但不是绝对的。就像本文档中写的考场骗分的各种操作，虽然有用，但是如果过于依赖骗分而忽视了真正的算法就得不偿失了。

最后，祝各位 $OIerRP++$!

如果标算太难请坚定信念，

不如回头再看一眼题面，

以那暴力模拟向正解吊唁。

——《膜你抄》