ICPC Templates

Shanda

April 1, 2021

Contents

1 基础算法

1.1 快速排序

1.1.1 模板

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
#define N 100010
int num[N];
int n;
int quick_sort(int num[], int l, int r){
    if(l >= r) return 0;
    int mid = num[l+r >> 1], i = l-1, j = r + 1;
    while(i < j){
        while(num[++i] < mid);
        while(num[--j] > mid);
        if(i < j) swap(num[i],num[j]);
    }
    quick_sort(num, l, j);
    quick_sort(num, j + 1, r);
}
int main(){
    scanf("%d", &n);
    for(int i =0 ;i < n ; i++) scanf("%d", &num[i]);
    quick_sort(num, 0, n-1);
    for(int i = 0; i < n; i++) printf("%d ", num[i]);
    return 0;
}
```

1.1.2 第 k 个数

```cpp
#include <iostream>
using namespace std;
#define N 100010

int num[N];
int n,k;

int quick_sort(int num[], int l, int r,int k){
    if(l>=r) return 0;
    int x = num[l + r>>1], i = l-1, j= r+1;
    while(i < j){
        while(num[++i] < x);
        while(num[--j] > x);
        if(i < j) swap(num[i],num[j]);
    }

    int sk = j - l + 1;
    if(k <= sk) quick_sort(num, l, j, k);
    else quick_sort(num, j+1, r, k - sk);
}
```

```
21
22 int main(){
23     scanf("%d %d", &n, &k);
24     for(int i = 0; i< n;i++) scanf("%d", &num[i]);
25     quick_sort(num, 0, n-1, k);
26     printf("%d", num[k-1]);
27     return 0;
28 }
```

1.2 归并排序

1.2.1 模板

```
1  #include<iostream>
2  using namespace std;
3  #define N 100010
4
5  int q[N];
6  int n;
7
8
9  int merge_sort(int q[], int l, int r){
10     if(l>=r) return 0;
11     int mid = l+r >>1;
12     merge_sort(q,l,mid);
13     merge_sort(q,mid+1,r);
14     int i=l,j=mid+1,k=0,tmp[r-l+1];
15     while(i<=mid&&j<=r){
16         if(q[i]<=q[j]) tmp[k++] = q[i++];
17         else tmp[k++] = q[j++];
18     }
19     while(i<=mid) tmp[k++] = q[i++];
20     while(j<=r) tmp[k++] = q[j++];
21     for(int i=0,j=l;j<=r;i++,j++) q[j] = tmp[i];
22 }
23
24 int main(){
25     scanf("%d", &n);
26     for(int i = 0 ;i < n;i++) scanf("%d", &q[i]);
27     merge_sort(q,0,n-1);
28     for(int i = 0;i <n ;i++) printf("%d ", q[i]);
29     return 0;
30 }
```

1.2.2 逆序对数量

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4
5  #define N 100010
6
```

```cpp
int q[N];
int n;
long long res=0;

int merge_sort(int q[], int l, int r){
    if(l>=r) return 0;
    int mid = l + r >> 1;
    merge_sort(q,l,mid);
    merge_sort(q,mid+1,r);
    int i=l,j=mid+1,k=0,tmp[r-l+1];
    while(i<=mid&&j<=r){
        if(q[i] <= q[j]) tmp[k++] = q[i++];
        else {
            tmp[k++] = q[j++];
            res += (mid - i + 1);
        }
    }
    while(i<=mid) tmp[k++] = q[i++];
    while(j<=r) tmp[k++] = q[j++];
    for(int i=0,j=l;j<=r;i++,j++) q[j] = tmp[i];
}

int main(){
    scanf("%d", &n);
    for(int i= 0; i< n ; i++) scanf("%d", &q[i]);
    merge_sort(q, 0, n-1);
    printf("%lld",res);
    return 0;
}
```

1.3 二分

1.3.1 数的范围

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
#define N 100010


int n,q,k;
int a[N];

int lower_find(int a[], int l, int r, int k){
    while(l < r){
        int mid = l + r >> 1;
        if(k <= a[mid]) r = mid;
        else l = mid+1;
    }
    return a[l] == k ? l : -1;
}

```

```
int upper_find(int a[], int l, int r, int k){
    while(l < r){
        int mid = l+r+1 >> 1;
        if(k >= a[mid]) l = mid;
        else r = mid -1;
    }
    return a[l] == k ? l : -1;
}

int main(){
    scanf("%d %d",&n, &q);
    for(int i= 0 ;i< n;i++) scanf("%d", &a[i]);
    for(int i =0 ;i < q;i++) {
        scanf("%d", &k);
        int res = lower_find(a, 0, n-1, k);
        if(res == -1) printf("%d %d", -1, -1);
        else{
           printf("%d ", res);
           res = upper_find(a, 0, n-1, k);
           printf("%d", res);
        }
        if(i != q - 1)
            printf("\n");
    }
}
```

1.3.2 数的三次方根

```
#include <iostream>
using namespace std;
# define N 100010

double n;

int main(){
    scanf("%lf", &n);
    double l = -10000, r = 10000, x=0;
    while(r - l >= 10e-8){
        x = (l + r) / 2;
        if(x * x * x >=n) r = x;
        else l = x;
    }
    printf("%lf", x);
}
```

1.4 高精度运算

1.4.1 加法

```
#include <iostream>
#include <algorithm>
#include <string>
```

```cpp
#include <vector>
using namespace std;

vector<int> add(vector<int> &A, vector<int> &B){
    vector<int> C;
    int t = 0;
    for(int i=0; i < A.size() || i < B.size() ; i++){
        if(i < A.size()) t += A[i];
        if(i < B.size()) t += B[i];
        C.push_back(t % 10);
        t /= 10;
    }
    if(t) C.push_back(1);
    return C;
}

int main(){
    vector<int> A,B;
    string a,b;
    cin >> a >> b;
    for(int i= a.size() - 1; i>=0; i--) A.push_back(a[i] - '0');
    for(int i= b.size() -1; i>=0;i--) B.push_back(b[i] - '0');
    auto C = add(A,B);
    for(int i = C.size() -1; i>=0;i--) printf("%d", C[i]);
    return 0;
}
```

1.4.2 减法

```cpp
#include<iostream>
#include<string>
#include<algorithm>
#include<vector>
using namespace std;


// A >= B
bool cmp(vector<int> &A, vector<int> &B){
    if(A.size() > B.size()) return true;
    else if(A.size() < B.size()) return false;
    else{
        for(int i = A.size() - 1; i >= 0; i--)
            if(A[i] != B[i]) return A[i] > B[i];
        return true;
    }
}

vector<int> sub(vector<int> &A, vector<int> &B){
    vector<int> C;
    int t = 0;
    for(int i = 0; i < A.size(); i++){
        t = A[i] -t; // 减掉进位
```

```
        if(i < B.size()) t -= B[i];
        C.push_back((t+10)%10);
        if(t<0) t=1;
        else t=0;
    }
    while(C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

int main(){
    string a, b;
    vector<int> A, B;
    cin >> a >> b;
    for(int i = a.size() - 1;i>=0; i--) A.push_back(a[i] -'0');
    for(int i= b.size() -1; i>=0;i--) B.push_back(b[i] - '0');
    vector<int> C;
    if(cmp(A,B)){
        C = sub(A,B);
    }
    else{
        printf("%c", '-');
        C = sub(B,A);
    }
    for(int i = C.size() -1; i>=0 ; i--) printf("%d", C[i]);
    return 0;
}
```

1.4.3 乘法

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<string>
using namespace std;


vector<int> mul(vector<int> &A, int b){
    vector<int> C;
    int t = 0;
    for(int i = 0; i < A.size() || t; i++){
        if(i < A.size()) t += A[i] * b;
        C.push_back(t % 10);
        t /= 10;
    }
    while(C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}


int main(){
    string a;
    int b;
```

```
    vector<int> A,B;
    cin>>a>>b;
    for(int i = a.size() - 1 ;i >= 0 ; i --) A.push_back(a[i] - '0');
    // for(int i = b.size() - 1 ;i >= 0 ; i ++) B.push_back(b[i] - '0');
    auto C = mul(A,b);
    for(int i = C.size() - 1; i>=0; i--) printf("%d" ,C[i]);
}
```

1.4.4 除法

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;


vector<int> div(vector<int> &A, int b, int &r){
    vector<int> C;
    int t = 0;
    for(int i = A.size() -1 ; i >=0 ;i --){
        r = r * 10 + A[i];
        C.push_back(r / b);
        r %= b;
    }
    reverse(C.begin(),C.end());
    while(C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}


int main(){
    string a;
    int b, r=0;
    vector<int> A;
    cin >> a >> b;
    for(int i = a.size() - 1 ; i>=0; i--) A.push_back(a[i] - '0');
    auto C = div(A,b,r);
    for(int i =C.size() -1 ; i>=0; i--) printf("%d", C[i]);
    printf("\n%d", r);
    return 0;
}
```

1.5 前缀与差分

1.5.1 前缀和

```
#include<iostream>
using namespace std;
#define N 100010
int a[N];
int S[N];
```

```
int n,m,l,r;
int main(){
    scanf("%d %d",&n,&m);
    for(int i = 1; i<=n; i++) scanf("%d", &a[i]);
    for(int i =1 ;i<=n;i++) S[i] = S[i-1] + a[i];
    while(m--){
        scanf("%d %d", &l, &r);
        printf("%d\n" ,S[r] - S[l-1]);
    }
}
```

1.5.2 子矩阵的和

```
#include<iostream>
using namespace std;

#define N 1010

int a[N][N];
int S[N][N];

int n,m,q,x1,x2,y1,y2;

int main(){
    scanf("%d %d %d",&n,&m, &q);
    for(int i = 1; i<=n;i++)
        for(int j=1; j<=m; j++)
          scanf("%d", &a[i][j]);

    for(int i = 1;i<=n;i++)
        for(int j = 1; j<=m;j++)
           S[i][j] = S[i-1][j] + S[i][j-1] - S[i-1][j-1] + a[i][j];


    while(q--){
        scanf("%d %d %d %d", &x1,&y1, &x2, &y2);
        printf("%d\n", S[x2][y2] - S[x1-1][y2] - S[x2][y1-1] + S[x1-1][y1-1]);
    }
    return 0;
}
```

1.5.3 差分

```
#include <iostream>
using namespace std;

#define N 100010

int a[N], b[N];
int n,m,l,r,c;

```

```
int insert(int b[],int l, int r , int c){
    b[l] += c;
    b[r+1] -= c;
}

int main(){
    scanf("%d %d", &n, &m);
    for(int i=1; i<=n;i++) scanf("%d", &a[i]);
    for(int i=1; i<=n;i++) insert(b,i,i,a[i]);
    while(m--){
        scanf("%d %d %d",&l, &r, &c);
        insert(b,l,r,c);
    }
    for(int i=1;i<=n;i++)
        b[i] += b[i-1];
    for(int i=1;i<=n;i++) printf("%d ", b[i]);
}
```

1.5.4 差分矩阵

```
#include <iostream>
using namespace std;
#define N 1010


int a[N][N], b[N][N];

int n,m,q,x1,x2,y1,y2,c;

int insert(int x1, int x2, int y1 ,int y2, int c){
    b[x1][y1] += c;
    b[x1][y2+1] -= c;
    b[x2+1][y1] -=c;
    b[x2+1][y2+1] +=c;
    return 0;
}

int main(){
    scanf("%d %d %d",&n,&m,&q);
    for(int i = 1;i <=n ;i++)
        for(int j = 1; j<=m; j++){
            scanf("%d", &a[i][j]);
            insert(i,i,j,j,a[i][j]);
        }
    while(q--){
        scanf("%d %d %d %d %d", &x1,&y1,&x2,&y2,&c);
        insert(x1,x2,y1,y2,c);
    }

    for(int i = 1; i<=n ;i++)
        for(int j=1; j<=m; j++)
            b[i][j] = b[i-1][j] + b[i][j-1] - b[i-1][j-1] + b[i][j];
```

```
33
34      for(int i = 1;i<=n;i++){
35          for(int j = 1 ; j<=m; j++) {
36              printf("%d ", b[i][j]);
37          }
38          printf("\n");
39      }
40  }
```

1.6 双指针算法

1.6.1 最长连续不重复子序列

```
1   #include<iostream>
2   #include<algorithm>
3   using namespace std;
4
5
6   #define N 100010
7
8   int n,a[N];
9   int cnt[N];
10
11  int main(){
12      scanf("%d", &n);
13      int res = 0;
14      for(int i=0; i<n;i++) scanf("%d", &a[i]);
15
16      for(int i = 0,j=i ; i< n; i++){
17          ++cnt[a[i]];
18          while(cnt[a[i]] > 1) --cnt[a[j++]];
19          res = max(res, i-j + 1);
20      }
21
22      printf("%d", res);
23
24      return 0;
25  }
```

1.6.2 数组元素的目标和

```
1   #include<iostream>
2   #include<algorithm>
3   using namespace std;
4
5   #define N 100010
6
7   int n,m,x;
8   int a[N], b[N];
9
10  int main(){
11      scanf("%d %d %d", &n, &m, &x);
```

```
    for(int i = 0 ;i < n; i++) scanf("%d", &a[i]);
    for(int i = 0 ;i < m; i++) scanf("%d", &b[i]);
    int i = n-1,j = 0;
    while(i >= 0 || j < m){
        if(i>=0 && a[i] + b[j] > x) i--;
        else if(j < m && a[i] +b[j] < x) j++;
        else if(i>=0 && j< m && a[i]+b[j] == x){
            printf("%d %d",i ,j);
            break;
        };
    }
    return 0;
}
```

1.6.3 判断子序列

```
#include <iostream>
#include <algorithm>
#include <string>
#include <map>
using namespace std;
#define N 100010

int a[N],b[N];
int n,m;
int main(){
    scanf("%d %d", &n, &m);
    for(int i = 0 ;i < n ;i ++) scanf("%d", &a[i]);
    for(int i = 0 ;i < m ;i ++) scanf("%d", &b[i]);
    int i = 0, j = 0;
    for(; j < m; j++){
        if(i < n && a[i] == b[j]) i++;
    }
    if(i == n) printf("Yes");
    else printf("No");

    return 0;
}
```

1.7 位运算

1.7.1 二进制中 1 的个数

```
#include <iostream>
using namespace std;

#define N 100010

int lowbit(int x){
    return x & -x;
}

```

```
int n;

int main(){
    scanf("%d", &n);
    int x;
    for(int i = 0 ; i < n; i++ ){
        scanf("%d", &x);
        int cnt = 0;
        while(x){
            x = x - lowbit(x);
            cnt++;
        }
        printf("%d ",cnt);
    }
    return 0;
}
```

1.8 离散化

1.8.1 区间和

假定有一个无限长的数轴，数轴上每个坐标上的数都是 0。现在，我们首先进行 n 次操作，每次操作将某一位置 x 上的数加 c。接下来，进行 m 次询问，每个询问包含两个整数 l 和 r，你需要求出在区间 [l, r] 之间的所有数的和。

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;

#define N 300000 + 10
typedef pair<int,int> PII;
vector<PII> opers,query;
vector<int> alls;
int a[N];

int n, m, x,c, l,r;

int binary_search(int k){
    int l = 0 , r = alls.size() - 1;
    while(l < r){
        int mid = l + r >> 1;
        if(k <= alls[mid]) r = mid;
        else l = mid + 1;
    }
    return alls[l] == k ? l+1 : -1;
}

int main(){
    scanf("%d %d", &n, &m);
    for(int i = 0 ;i < n; i++){
        scanf("%d %d", &x, &c);
```

```
28          opers.push_back({x,c});
29          alls.push_back(x);
30      }
31
32      for(int i= 0; i < m; i++){
33          scanf("%d %d", &l, &r);
34          query.push_back({l,r});
35          alls.push_back(l);
36          alls.push_back(r);
37      }
38      sort(alls.begin(),alls.end());
39      alls.erase(unique(alls.begin(),alls.end()),alls.end());
40
41      for(auto p : opers){
42          int index = binary_search(p.first);
43          a[index] += p.second;
44      }
45
46      for(int i = 1; i<= alls.size(); i++) a[i] = a[i-1] + a[i];
47
48      for(auto p : query){
49          int l = binary_search(p.first);
50          int r = binary_search(p.second);
51          printf("%d\n", a[r] - a[l-1]);
52      }
53
54  }
```

1.9 区间合并

1.9.1 区间合并

```
1   #include <iostream>
2   #include <vector>
3   #include <algorithm>
4   using namespace std;
5
6   typedef pair<int,int> PII;
7
8   int n;
9   int l, r;
10  vector<PII> segs;
11
12  int merge(vector<PII> &segs){
13      sort(segs.begin(),segs.end());
14      int st = -2e9, ed = -2e9;
15      vector<PII> res;
16      for(auto seg : segs){
17          if(ed < seg.first){
18              if(st!=-2e9) res.push_back({st,ed});
19              st = seg.first;
20              ed = seg.second;
```

```
        }
        else{
            ed = max(ed, seg.second);
        }
    }
    if(st != -2e9) res.push_back({st,ed});
    return res.size();
}

int main(){
    scanf("%d",&n);
    while(n--){
        scanf("%d %d", &l,&r);
        segs.push_back({l,r});
    }
    int res = merge(segs);
    printf("%d", res);
    return 0;
}
```

2 数据结构

2.1 单链表

```

#include <iostream>
#include <string>
using namespace std;
const int N = 100010;
string c;
int head,idx,el[N],ne[N],k,x,m;
void init(){
    head = -1;
    idx = 0;
}
void add_head(int x){
    el[idx] = x, ne[idx] = head, head = idx++;
}
void add(int k, int x){
    el[idx] = x; ne[idx] = ne[k], ne[k] = idx++;
}
void del_head(){
    head = ne[head];
}
void del(int k){
    ne[k] = ne[ne[k]];
}
int main(){
    cin >> m;
    init();
    while(m--){
        cin >> c;
```

```
        if(c == "H"){
            cin >> x;
            add_head(x);
        }
        if(c == "D"){
            cin >> k;
            if(!k) del_head();
            else del(k-1);
        }
        if(c =="I"){
            cin >> k >> x;
            add(k-1,x);
        }
    }
    for(int i = head; i!=-1;i=ne[i]){
       cout << el[i] << ' ';
    }
    return 0;
}
```

2.2 双链表

```
#include <iostream>
#include <string>
using namespace std;

const int N = 100010;

string oper;
int m,k,x,idx,l[N],r[N],el[N];

void init(){
    // 0是左节点，1是右节点；
    r[0] = 1, l[1]=0;
    idx = 2;
}
//在第k个数的右边添加新节点
void add(int k, int x){
    el[idx] = x, l[idx] = k, r[idx] = r[k], l[r[k]] = idx, r[k] = idx ++;
}
//删除第k个节点
void del(int k){
    r[l[k]] = r[k];
    l[r[k]] = l[k];
}
int main(){
    cin >> m;
    init();
    while(m--){
        cin >> oper;
        if(oper == "L"){
            cin >> x;
```

```
            add(0,x);
        }
        if(oper == "R"){
            cin >> x;
            add(l[1],x);
        }
        if(oper == "D"){
            cin >> k;
            del(k+1);
        }
        if(oper == "IL"){
            cin >> k >> x;
            add(l[k+1], x);
        }
        if(oper == "IR"){
            cin >> k >> x;
            add(k+1, x);
        }
    }
    for(int i = r[0]; i!=1; i = r[i]){
        cout << el[i] << ' ';
    }
    return 0;
}
```

2.3 模拟栈

```
#include <iostream>
#include <string>
using namespace std;

const int N = 100010;

int m;
int s[N], t=0, x;
string oper;

void init(){
    t = 0;
}

void push(int x){
    s[++t] = x;
}

void pop(){
    --t;
}

string empty(){
    return t == 0 ? "YES" : "NO";
}
```

```

int query(){
    return s[t];
}


int main(){
    cin >> m;
    init();
    while(m--){
        cin >> oper;
        if(oper == "push"){
            cin >> x;
            push(x);
        }
        if(oper == "pop"){
            pop();
        }
        if(oper == "empty"){
            cout << empty() << endl;
        }
        if(oper == "query"){
            cout << query() << endl;
        }
    }
}
```

2.4 模拟队列

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
const int N = 100010;

int q[N], h,t,m,x;
string o;

void init(){
    h = 0, t = -1;
}

void push(int x){
    q[++t] = x;
}

void pop(){
    ++h;
}

int query(){
    return q[h];
```

```
}

string empty(){
    return h > t ? "YES" : "NO";
}

int main(){
    cin >> m;
    init();
    while(m--){
        cin >> o;
        if(o == "push"){
            cin >> x;
            push(x);
        }
        if(o == "pop"){
            pop();
        }
        if(o == "empty"){
            cout << empty() <<endl;
        }
        if(o == "query"){
            cout << query() << endl;
        }
    }
}
```

2.5 单调栈

```
#include <iostream>
using namespace std;

const int N = 100010;

int s[N], a[N], t=0,n;


int main(){
    scanf("%d", &n);
    for(int i = 0 ;i < n; i++) scanf("%d", &a[i]);
    for(int i = 0;i < n; i++){
        while(t && s[t] >= a[i]) t--;
        if(t){
            printf("%d ",s[t]);
        }
        else{
            printf("%d ",-1);
        }
        s[++t] = a[i];
    }
    return 0;
}
```

2.6 单调队列

```cpp
#include <iostream>
using namespace std;

const int N = 1000010;
int n,k,q[N],a[N];
int h=0,t=-1;


int main(){
    scanf("%d %d",&n,&k);
    for(int i = 0 ; i < n;i++) scanf("%d", &a[i]);

    for(int i = 0; i< n;i++) {
        while(h<=t && q[h] < i - k + 1) h++; // 去除不在滑动窗口内的下标;
        while(h<=t && a[q[t]]>= a[i]) t--; // 单调队列中所有元素小于当前元素;
        q[++t] = i; // 队列储存的是下标;
        if(i+1>=k){ // 滑动窗口未完全进入不输出;
            printf("%d ", a[q[h]]);
        }
    }
    puts("");
    h=0, t=-1;
    for(int i = 0; i< n;i++) {
        while(h<=t && q[h] < i - k + 1) h++;
        while(h<=t && a[q[t]]<= a[i]) t--;
        q[++t] = i;
        if(i+1>=k){
            printf("%d ", a[q[h]]);
        }
    }

    return 0;
}
```

2.7 kmp

```cpp
#include <iostream>
using namespace std;

const int N = 1000010;

int ne[N],n,m;
char p[N],s[N];

int main(){
    scanf("%d %s %d %s",&n, p+1, &m, s+1);
    for(int i = 2 ,j = 0; i <=n;i++){
        while(j && p[i] != p[j+1]) j = ne[j];
        if(p[i] == p[j+1]) j++;
        ne[i] = j;
```

```
    }
    for(int i = 1, j= 0; i<=m;i++){
        while(j && s[i] != p[j+1]) j=ne[j];
        if(s[i] == p[j+1]) j++;
        if(j == n){
            j = ne[j];
            printf("%d ", i-n);
        }
    }
}
```

2.8 Trie 树

```
#include<iostream>
using namespace std;

const int N = 2e4 + 10;

char oper[2],str[N];
int cnt[N],n, s[N][26],idx=0;

void insert(char * str){
    int p = 0;
    for(int i = 0 ; str[i]; i++){
        int c = str[i] - 'a';
        if(!s[p][c]) s[p][c] = ++idx;
        p = s[p][c];
    }
    ++cnt[p];
}

int query(char * str){
    int p = 0;
    for(int i = 0; str[i]; i++){
        int c = str[i] - 'a';
        if(!s[p][c]) return 0;
        p = s[p][c];
    }
    return cnt[p];
}

int main(){
    scanf("%d", &n);
    while(n--){
        scanf("%s %s",oper,str);
        // cout << oper <<endl << str <<endl;
        if(oper[0] == 'I'){
            // printf("%d\n",insert())
            insert(str);
        }
        if(oper[0] == 'Q'){
            printf("%d\n", query(str));
```

```
        }
    }
    return 0;
}
```

2.9 最大异或树

```
#include <iostream>
using namespace std;
const int N = 1e5 * 31 + 10;


int n, son[N][2],idx=0;

void insert(int x){
    int p = 0;
    for(int i = 31; i>=0;i--){ // 贪心，从整数最高位开始构造Tried树;
        int b = x >> i & 1;
        if(!son[p][b]) son[p][b] = ++idx;
        p = son[p][b];
    }
}

int query(int x){
    int p = 0, res=0;
    for(int i = 31;i >=0; i--){
        int b = x >> i & 1;
        if(!son[p][!b]){
            res = 2 * res;
            p = son[p][b];
        }
        else{
            res = 2 * res + 1;
            p = son[p][!b];
        }
    }
    return res;
}


int main(){
    scanf("%d", &n);
    int res = 0,x=0;
    for(int i = 0; i < n;i ++){
        scanf("%d", &x);
        insert(x);
        res = max(res, query(x));
    }
    printf("%d", res);

    return 0;
}
```

2.10 并查集

2.10.1 合并集合

```cpp
#include <iostream>
using namespace std;
const int N = 1e5 * 31 + 10;


int n, son[N][2],idx=0;

void insert(int x){
    int p = 0;
    for(int i = 31; i>=0;i--){ // 贪心，从整数最高位开始构造Tried树;
        int b = x >> i & 1;
        if(!son[p][b]) son[p][b] = ++idx;
        p = son[p][b];
    }
}

int query(int x){
    int p = 0, res=0;
    for(int i = 31;i >=0; i--){
        int b = x >> i & 1;
        if(!son[p][!b]){
            res = 2 * res;
            p = son[p][b];
        }
        else{
            res = 2 * res + 1;
            p = son[p][!b];
        }
    }
    return res;
}


int main(){
    scanf("%d", &n);
    int res = 0,x=0;
    for(int i = 0; i < n;i ++){
        scanf("%d", &x);
        insert(x);
        res = max(res, query(x));
    }
    printf("%d", res);

    return 0;
}
```

2.10.2 连通块中点的数量

```cpp
#include <iostream>
```

```cpp
using namespace std;

const int N = 100010;
char oper[3];
int p[N], s[N], n, m , x, y;

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


int main(){
    scanf("%d %d", &n, &m);
    for(int i = 1; i <=n; i++) p[i] = i,s[i] = 1;
    while(m--){
        scanf("%s",oper);
        if(oper[0] == 'C'){
            scanf("%d%d",&x, &y);
            int px = find(x), py = find(y);
            if(px != py){
                s[py] += s[px];
                p[px] = py;
            }
        }
        if(oper[0] == 'Q'){
            if(oper[1] == '1'){
                scanf("%d%d",&x, &y);
                if(find(x) == find(y)) printf("Yes\n");
                else printf("No\n");
            }
            if(oper[1] == '2'){
                 scanf("%d",&x);
                 printf("%d\n", s[find(x)]);
            }
        }
    }
}
```

2.10.3 食物链

```cpp
#include <iostream>
using namespace std;

const int N = 50010;


int n,k,d,x,y,res=0,dist[N],p[N];

int find(int x){
    if(p[x] != x){
        int t = find(p[x]);
```

```
        dist[x] += dist[p[x]];
        p[x] = t;
    }
    return p[x];
}

int main(){
    scanf("%d%d",&n,&k);
    for(int i = 1;i<=n;i++) p[i] = i, dist[i] = 0;
    while(k--){
        scanf("%d%d%d",&d,&x,&y);
        if( x > n || y > n) res++;
        else{
            if(d==1){
                int px = find(x), py = find(y);
                if(px == py && (dist[x] - dist[y]) % 3) res++;
                else if(px!=py){
                    p[px] = py;
                    dist[px] = dist[y] - dist[x];
                }
            }
            else{
                int px = find(x), py =find(y);
                if(px == py && (dist[x] - dist[y] - 1) %3) res++;
                else if(px!=py){
                    p[px] = py;
                    dist[px] = dist[y] + 1 -dist[x];
                }
            }
        }
    }
    printf("%d",res);
}
```

2.11 堆

2.11.1 堆排序

```
#include <iostream>
#include <algorithm>
using namespace std;

const int N = 100010;

int n,m,h[N];



void down(int u){
    int t = u;
    if(2*u<=n&&h[2*u] < h[t]) t = 2*u;
    if(2*u+1<=n&&h[2*u+1] < h[t]) t = 2*u+1;
```

```
    if(u!=t){
        swap(h[u],h[t]);
        down(t);
    }
}


int main(){
    scanf("%d%d",&n,&m);
    for(int i = 1;i<=n;i++) scanf("%d",&h[i]);
    for(int i = n/2;i;i--){
        down(i);
    }
    while(m--){
        printf("%d ", h[1]);
        swap(h[1],h[n]);
        n--;
        down(1);
    }
    return 0;
}
```

2.11.2 模拟堆

```
#include <iostream>
#include <string.h>
using namespace std;
const int N = 100010;

char o[3];
// string o;
int idx=0,n,sz=0,x,k,h[N],hp[N],ph[N];


void heap_swap(int u, int d){
    swap(ph[hp[u]],ph[hp[d]]);
    swap(h[u],h[d]);
    swap(hp[u],hp[d]);
}

int down(int u){
    int t = u;
    if(2*u<=sz && h[2*u] < h[t]) t = 2*u;
    if(2*u+1<=sz &&h[2*u+1] < h[t]) t = 2*u+1;
    if(u!=t){
        heap_swap(u,t);
        down(t);
    }
}


int up(int u){
```

```
    while (u/2 && h[u] < h[u/2]){
        heap_swap(u/2,u);
        u >>=1;
    }
}


int main(){
    scanf("%d", &n);
    while(n--){
        scanf("%s", o);
        if(!strcmp(o,"I")){ // 插入一个数
            scanf("%d", &x);
            h[++sz] = x;
            hp[sz] = ++idx;
            ph[idx] = sz;
            up(sz);
        }
        if(!strcmp(o,"PM")){ // 输出当前集合中的最小值
            printf("%d\n", h[1]);
        }
        if(!strcmp(o,"DM")){ // 删除当前集合的最小值
            heap_swap(1,sz--);
            down(1);
        }
        if(!strcmp(o,"D")){ // 删除第k个插入的数
            scanf("%d", &k);
            int u = ph[k];
            heap_swap(u, sz--);
            up(u);
            down(u);
        }
        if(!strcmp(o,"C")){ // 修第k个插入的数
            scanf("%d%d",&k, &x);
            h[ph[k]] = x;
            up(ph[k]);
            down(ph[k]);
        }
    }
}
```

2.12 哈希表

2.12.1 模拟散列表

```
#include<iostream>
#include <string.h>
using namespace std;

const int N = 100010;
char oper[2];
int idx=0,n,x, el[N],ne[N],h[N], mod =100003;
```

```

void insert(int x){
    int t = (x % mod + mod) % mod;
    el[idx] = x, ne[idx] = h[t], h[t] = idx++;
}
bool query(int x){
    int t = (x % mod + mod) % mod;
    for(int i = h[t]; i!= -1;i = ne[i]){
        if(el[i] == x) return true;
    }
    return false;
}

int main(){
    scanf("%d", &n);
    memset(h,-1,sizeof(h));
    while(n--){
        scanf("%s%d",oper,&x);
        if(oper[0]=='I'){
            insert(x);
        }
        if(oper[0] == 'Q'){
            bool is_exisit = query(x);
            if(is_exisit)
                printf("%s\n","Yes");
            else
                printf("%s\n","No");
        }
    }
}
```

2.12.2 字符串哈希

```
#include<iostream>
using namespace std;
const int N = 100010;

typedef unsigned long long ULL;
ULL p[N],h[N];
int n,m,l1,r1,l2,r2,P=1331;
char str[N];

ULL get(int l,int r){
    return h[r] - h[l-1] * p[r-l+1];
}

int main(){
    scanf("%d%d",&n,&m);
    scanf("%s",str+1);
    p[0] = 1;
    for(int i=1;i <=n;i++){
        p[i] = p[i-1] * P;
```

```
        h[i] = h[i-1] * P + str[i];
    }
    while(m--){
        scanf("%d%d%d%d",&l1,&r1,&l2,&r2);
        if(get(l1,r1) == get(l2,r2)) puts("Yes");
        else puts("No");
    }
}
```

3 搜索与图论

3.1 DFS

3.1.1 排列数字

```
#include<iostream>
using namespace std;

const int N = 8;

int n,p[N],st[N];

void dfs(int u){
    if(u == n){
        for(int i = 0; i< n;i++) printf("%d ",p[i]);
        puts("");
    }

    for(int i = 1; i<=n ; i++){
        if(!st[i]){
            st[i] = true;
            p[u] = i;
            dfs(u+1);
            st[i] = false;
        }
    }
}

int main(){
    scanf("%d",&n);
    dfs(0);
}
```

3.1.2 n 皇后问题

```
#include <iostream>
using namespace std;

const int N = 20;

char q[N][N];
```

```
bool dg[N],udg[N],col[N];
int n;

void dfs(int u ){
    if( u == n){
        for(int i = 0 ; i < n; i++) puts(q[i]);
        puts("");
    }

    for(int i = 0 ; i < n; i ++){
        if(!col[i] && !dg[u+i] && !udg[n-u+i]){
            q[u][i] = 'Q';
            col[i] = dg[u+i] = udg[n-u+i] = true; // 用截距表示对角线与反对角线
            dfs(u+1);
            col[i] = dg[u+i] = udg[n-u+i] = false;
            q[u][i] = '.';
        }
    }
}

int main(){
    scanf("%d",&n);
    for(int i = 0 ;i<n;i++)
        for(int j = 0; j< n; j++)
            q[i][j] = '.';
    dfs(0);
}
```

3.2 BFS

3.2.1 走迷宫

```
#include <iostream>
#include <string.h>
using namespace std;

typedef pair<int,int> PII;
const int N = 110;

int n,m,h=0,t=-1,d[N][N],g[N][N],dx[4]={-1,0,1,0},dy[4]={0,1,0,-1};
PII q[N*N];

int bfs(PII u){
  memset(d,-1,sizeof(d));
  d[u.first][u.second] = 0;
  q[++t] = u;
  while(h<=t){
      auto e = q[h++];
      for(int i = 0 ;i < 4;i++){
          int x = e.first + dx[i], y = e.second + dy[i];
          if(x <= n && x >=1 && y<=m && y>=1 && d[x][y] == -1 && !g[x][y]){
              d[x][y] = d[e.first][e.second] + 1;
```

```
21              q[++t] = {x,y};
22          }
23      }
24   }
25   return d[n][m];
26}
27
28
29int main(){
30    scanf("%d%d",&n,&m);
31    for(int i = 1 ;i <= n ; i++)
32        for(int j = 1; j<= m; j++)
33            scanf("%d",&g[i][j]);
34
35    printf("%d",bfs({1,1}));
36}
```

3.2.2 八重码

```
1 #include <iostream>
2 #include <queue>
3 #include <string>
4 #include <unordered_map>
5 using namespace std;
6
7 int dx[4] = {-1,0,1,0},dy[4] = {0,1,0,-1};
8 queue<string> q;
9 string u;
10 unordered_map<string,int> d;
11
12
13 int bfs(string u){
14     q.push(u);
15     d[u] = 0;
16     string end = "12345678x";
17     while(!q.empty()){
18         auto t = q.front();
19         q.pop();
20         if(t == end) return d[t];
21         int k = t.find('x');
22         int a = k / 3, b = k % 3;
23         int dist = d[t];
24         for(int i = 0 ; i < 4;i++){
25             int x = a + dx[i], y = b + dy[i];
26             if(x>=0 && x < 3 && y>=0 && y < 3){
27                 swap(t[x * 3 + y], t[k]);
28                 if(!d.count(t)){
29                     d[t] = dist + 1;
30                     q.push(t);
31                 }
32                 swap(t[x * 3 + y], t[k]);
33             }
```

```
        }

    }
    return -1;
}


int main(){
    char c;
    for(int i = 0 ; i < 9;i++){
        scanf("%c ",&c);
        u+=c;
    }
    // cout << u << endl;
    printf("%d",bfs(u));
}
```

3.3 树与图的 DFS

3.3.1 树的重心

```
#include <iostream>
#include <string.h>
using namespace std;

const int N = 100010, M = 2 * N; // 无向图边数比有向图边数多两倍;

int h[N], el[M], ne[M], idx = 0, n,x,y;
int ans = N;
bool st[N];


void insert(int x ,int y){
    el[idx] = y, ne[idx] = h[x], h[x] = idx ++; //头插法
}

// 计算以u为根节点的树的节点数（包括u自身）
int dfs(int u){
    st[u] = true;
    int sum = 1, res = 0; // 当前树的节点数，连通子图的节点数最大值;
    for(int i = h[u]; i != -1; i = ne[i]){
        if(!st[el[i]]){
            int s = dfs(el[i]);
            sum += s;
            res = max(res, s);
        }
    }

    res = max(res, n - sum); // 比较剩余父连通图的节点数
    ans = min(ans, res);
    return sum;
}
```

```
32
33
34  int main(){
35      scanf("%d", &n);
36      memset(h,-1,sizeof(h));
37      for(int i = 0 ; i < n-1 ;i++){// n-1条边，题意
38          scanf("%d%d",&x,&y);
39          insert(x,y);
40          insert(y,x);// 无向图插两条边
41      }
42      dfs(1);
43      printf("%d", ans);
44  }
```

3.4 树与图的 BFS

3.4.1 图中点的层次

```
1   #include <iostream>
2   #include <string.h>
3   #include <queue>
4   using namespace std;
5   const int N = 100010, M = 100010;
6   int n,m,h[N],el[N], ne[M], idx=0, x, y ,d[N];
7   queue<int> q;
8
9
10  void insert(int x, int y){
11      el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
12  }
13
14  int bfs(int u){
15      q.push(u);
16      memset(d, -1, sizeof(d));
17      d[u] = 0;
18      while(!q.empty()){
19          int t = q.front();
20          q.pop();
21          for(int i = h[t]; i!=-1 ; i = ne[i]){
22              if(d[el[i]] == -1){
23                  d[el[i]] = d[t] + 1;
24                  q.push(el[i]);
25              }
26          }
27      }
28      return d[n];
29  }
30
31  int main(){
32      scanf("%d%d",&n,&m);
33      memset(h, -1,sizeof(h));
34      for(int i = 0 ; i < m ; i++){
```

```
35        scanf("%d%d",&x,&y);
36        insert(x,y);
37    }
38    printf("%d", bfs(1));
39 }
```

3.5 拓扑排序

3.5.1 有向图的拓扑排序

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  const int N = 100010, M = 100010;
6  int idx=0,el[M],ne[M],h[N],x,y,n,m,degree[N];
7  int q[M], hh = 0, tt= -1;
8
9  void insert(int x, int y){
10    el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
11 }
12
13 int topsort(){
14    for(int i = 1 ; i <= n ; i++){
15       if(!degree[i]) q[++tt] = i;
16    }
17    while(hh<=tt){
18       int t = q[hh++];
19       for(int i = h[t]; i != -1; i = ne[i]){
20          int j = el[i];
21          --degree[j];
22          if(!degree[j]){
23             q[++tt] = j;
24          }
25       }
26    }
27    return tt == n-1;
28 }
29
30 int main(){
31    memset(h,-1,sizeof(h));
32    memset(degree,0,sizeof(degree));
33    scanf("%d%d",&n,&m);
34    for(int i = 0; i < m; i++){
35       scanf("%d%d",&x,&y);
36       insert(x,y);
37       ++degree[y];
38    }
39    if(topsort()){
40       for(int i = 0; i<=tt;i++){
41          printf("%d ", q[i]);
42       }
```

```
    }
    else{
        printf("%d",-1);
    }
}
```

3.6 Dijkstra

3.6.1 求最短路 1

```
#include<iostream>
#include<string.h>
using namespace std;

const int N = 510, M = 100010;
int d[N][N], dist[M], n, m, x, y, z;
bool st[N];


int dijkstra(int u){
    memset(dist, 0x3f, sizeof dist);
    dist[u] =0;
    for(int i = 0; i<n;i++){
        int t = -1;
        for(int j = 1;j<=n;j++){
           if(!st[j] && ((t == -1) || dist[t] > dist[j]))
               t = j;
        }

        st[t] =true;

        for(int j=1;j<=n;j++){
            dist[j] = min(dist[j],dist[t] + d[t][j]);
        }
    }
    if(dist[n] == 0x3f3f3f3f) return -1;
    else return dist[n];
}

int main(){
    memset(d, 0x3f, sizeof d);
    scanf("%d%d",&n,&m);
    for(int i = 0 ; i < m ;i ++){
        scanf("%d%d%d",&x,&y,&z);
        d[x][y] = min(d[x][y],z);
    }
    int res = dijkstra(1);
    printf("%d",res);
    return 0;
}
```

3.6.2 求最短路 2

```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <string.h>
using namespace std;

const int N = 150010,M = 150010;

typedef pair<int,int> PII;
int dist[N],el[M],ne[M],w[M],h[N],n,m,x,y,z,idx=0;
bool st[N];
priority_queue<PII,vector<PII>,greater<PII>> q;

void add(int x, int y, int z){
    el[idx] = y, w[idx] = z, ne[idx] = h[x], h[x] = idx++;
}

int dijkstra(int u){
    memset(dist, 0x3f, sizeof dist);
    dist[u] = 0;
    q.push({dist[u], u});
    while(q.size()){
        auto t = q.top();
        q.pop();
        int ver = t.second, d = t.first;
        if(st[ver]) continue;
        st[ver] = true;
        for(int i = h[ver]; i != -1; i = ne[i]){
            int y = el[i];
            if(dist[y] > d + w[i]){
                dist[y] = d + w[i];
                q.push({dist[y],y});
            }
        }
    }
    if(dist[n] == 0x3f3f3f3f) return -1;
    return dist[n];

}

int main(){
    scanf("%d%d",&n,&m);
    memset(h, -1, sizeof h);
    for(int i = 0; i < m;i++){
        scanf("%d%d%d",&x,&y,&z);
        add(x,y,z);
    }
    printf("%d",dijkstra(1));
}
```

3.7 Bellman-Ford

3.7.1 有边数限制的最短路

```cpp
#include<iostream>
#include<string.h>
using namespace std;

const int N = 510, M = 10010;
int n,m,k,dist[N],backup[N];

struct Edge{
    int x,y,w;
}edges[M];

int bellman_ford(int u ){
    memset(dist,0x3f,sizeof dist);
    dist[u] = 0;
    for(int i = 0 ; i< k; i++){
        memcpy(backup,dist,sizeof backup);
        for(int j = 0; j < m; j++){
            int x = edges[j].x,y = edges[j].y, w= edges[j].w;
            dist[y] = min(dist[y], backup[x] + w);
        }
    }
    if(dist[n] > 0x3f3f3f3f / 2) return -1;
    return dist[n];
}

int main(){
    scanf("%d%d%d",&n,&m,&k);
    for(int i = 0 ; i < m ; i++){
        scanf("%d%d%d",&edges[i].x,&edges[i].y,&edges[i].w);
    }
    int res = bellman_ford(1);
    if(res == -1) printf("impossible");
    else printf("%d", res);
    return 0;
}
```

3.8 SPFA

3.8.1 SPFA 求最短路

```cpp
#include<iostream>
#include<string.h>
using namespace std;

const int N = 100010, M = 100010;
int idx=0,el[M], w[M], ne[M], h[N], dist[N], n, m, x, y, z;
int hh=0,tt=-1,q[N];
bool st[N];

```

```
10 void add(int x, int y, int z){
11     el[idx] = y, w[idx] = z, ne[idx] = h[x], h[x] = idx++;
12 }
13 
14 int spfa(int u){
15     memset(dist, 0x3f3f3f , sizeof dist);
16     dist[u] = 0;
17     q[++tt] =u;
18     while(hh <= tt){
19         int t = q[hh++];
20         st[t] = false;
21         for(int i = h[t]; i!= -1; i = ne[i]){
22             int j = el[i];
23             if(dist[j] > dist[t] + w[i]){
24                 dist[j] = dist[t] + w[i];
25                 if(!st[j]){
26                     q[++tt] = j;
27                     st[j] = true;
28                 }
29             }
30         }
31     }
32     if(dist[n] == 0x3f3f3f3f) return -1;
33     return dist[n];
34 }
35 
36 int main(){
37     memset(h, -1, sizeof h);
38     scanf("%d%d",&n,&m);
39     for(int i = 0 ; i < m ;i++){
40         scanf("%d%d%d",&x,&y,&z);
41         add(x,y,z);
42     }
43     int res = spfa(1);
44     if(res == -1 ) puts("impossible");
45     else printf("%d",res);
46 }
```

3.8.2 SPFA 判断负环

```
1  #include <iostream>
2  #include<string.h>
3  #include <queue>
4  using namespace std;
5  
6  const int N = 100010, M = 100010;
7  
8  int n,m,x,y,z,idx=0, h[N],el[M],ne[M],w[M],dist[N],cnt[N];
9  bool st[N];
10 queue<int> q;
11 
12 void add(int x, int y, int z){
```

```
    el[idx] = y, w[idx] = z ,ne[idx] = h[x], h[x] = idx ++;
}

bool spfa(){
    for(int i = 1; i<=n; i++){
        q.push(i);
        st[i] = true;
    }
    while(q.size()){
        int t = q.front();
        q.pop();
        st[t] = false;
        for(int i = h[t];i!=-1;i=ne[i]){
            int j = el[i];
            if(dist[j] > dist[t] + w[i]){
                dist[j] = dist[t] + w[i];
                cnt[j] = cnt[t] + 1;
                if(cnt[j] >=n) return true;
                if(!st[j]){
                    q.push(j);
                    st[j] = true;
                }
            }
        }
    }
    return false;
}

int main(){
    memset(h, -1, sizeof h);
    scanf("%d%d",&n,&m);
    for(int i = 0 ; i < m; i++){
        scanf("%d%d%d",&x,&y,&z);
        add(x,y,z);
    }
    if(spfa()) puts("Yes");
    else puts("No");
}
```

3.9 Floyd

3.9.1 Floyd 求最短路

```
#include <iostream>
#include <string.h>
using namespace std;

const int N = 210, M = 20010, INF = 0x3f3f3f3f;
int n, m, k, d[N][N], x, y, z;


void floyd(){
```

```
    for(int k = 1;k <= n; k++)
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j],d[i][k] + d[k][j]);
}


int main(){
    scanf("%d%d%d",&n,&m,&k);

    for(int i = 1; i<=n ;i ++)
        for(int j = 1; j <= n; j++){
            if(i == j) d[i][j] = 0;
            else d[i][j] = INF;
        }

    for(int i = 0; i < m ;i++){
        scanf("%d%d%d",&x, &y, &z);
        d[x][y] = min(d[x][y], z);
    }

    floyd();

    while(k -- ){
        scanf("%d%d",&x, &y);
        if(d[x][y] > INF / 2) puts("impossible");
        else printf("%d\n", d[x][y]);
    }
    return 0;

}
```

3.10 最小生成树

3.10.1 Kruskal

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 1e5 + 10, M = 2e5 + 10;
int n,m,x,y,z, p[N];

struct Edge{
    int x,y,z;
    bool operator < (const Edge &e) const{
        return z < e.z;
    }
}edges[M];

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

```
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i = 0 ;i < m; i++){
        scanf("%d%d%d", &x, &y, &z);
        edges[i] = {x,y,z};
    }

    for(int i = 1; i<= N; i++) p[i] =i;

    sort(edges , edges + m);

    int cnt = 0, res = 0;

    for(int i = 0; i<m; i++){
        int x = edges[i].x, y = edges[i].y, z = edges[i].z;

        int px = find(x), py = find(y);

        if(px != py){
            p[px] = py;
            cnt ++;
            res += z;
        }
    }

    if(cnt != n-1) puts("impossible");
    else printf("%d",res);
}
```

3.10.2 Prim

```
#include<iostream>
#include<string.h>
using namespace std;

const int N = 520, M = 2e105 + 10, INF = 0x3f3f3f3f;
int d[N][N], dist[N], n, m, x, y, z;
bool st[N];

int prime(){
    memset(dist, 0x3f, sizeof dist);
    int res = 0;
    for(int i = 0 ; i<n ;i++){
        int t = -1;
        for(int j = 1; j<=n ;j++){
            if(!st[j] && ((t == -1 ) || dist[t] > dist[j])){
                t = j;
            }
        }
        st[t] = true;
```

```
        if(i && dist[t] == INF) return -1;
        if(i) res+= dist[t];

        for(int j = 1; j<=n; j++){
            dist[j] = min(dist[j],d[t][j]);
        }
    }
    return res;
}

int main(){
    scanf("%d%d",&n,&m);
    memset(d, 0x3f, sizeof d);
    for(int i = 0 ; i < m ;i ++){
        scanf("%d%d%d", &x, &y, &z);
        d[x][y] = d[y][x] = min(d[x][y],z);
    }
    int res = prime();
    if(res == -1) puts("impossible");
    else printf("%d", res);
}
```

3.11 二分图

3.11.1 染色法判定二分图

```
#include<iostream>
#include<string.h>
using namespace std;

const int N = 1e5 + 10, M = 2e5 + 10;

int idx=0,el[M],ne[M],h[N],color[N],n,m,x,y;

void add(int x, int y){
    el[idx] = y, ne[idx] = h[x], h[x] = idx++;
}


bool dfs(int u, int c){
    color[u] = c;
    for(int i = h[u]; i != -1 ; i = ne[i]){
        int j = el[i];
        if(!color[j]){
            if(!dfs(j, 3-c)) return false;
        }
        else if(color[j] == c) return false;
    }
    return true;
}

int main(){
```

```
    memset(h, -1, sizeof h);
    scanf("%d%d",&n,&m);
    while(m --){
        scanf("%d%d",&x,&y);
        add(x,y),add(y,x);
    }

    bool flag = true;
    for(int i = 1 ; i <=n ; i++){
        if(!color[i]){
            if(!dfs(i, 1)){
                flag = false;
                break;
            }
        }
    }
    if(flag) puts("Yes");
    else puts("No");
}
```

3.11.2 匈牙利算法求二分最大匹配

```
#include<iostream>
#include<string.h>
using namespace std;

const int N = 510, M = 1e5 + 10;
int n1,n2,m,x,y,match[N],el[M], ne[M], h[N], idx=0;
bool st[N];
void add(int x, int y){
    el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
}


bool find(int u){
    for(int i = h[u];i!=-1;i=ne[i]){
        int j = el[i];
        if(!st[j]){
            st[j] = true;
            if(match[j] == 0 || find(match[j])){
                match[j] = u;
                return true;
            }
        }
    }
    return false;
}

int main(){
    memset(h, -1, sizeof h);
    scanf("%d%d%d",&n1, &n2, &m);
    for(int i = 0 ; i < m; i++){
```

```
        scanf("%d%d",&x,&y);
        add(x,y);
    }

    int res = 0;

    for(int i = 1 ; i<=n1; i++){
        memset(st, false, sizeof st);
        if(find(i)){
            res ++;
        }
    }
    printf("%d\n",res);
    return 0;

}
```

4 数论

4.1 质数

4.1.1 试除法判定质数

```
#include<iostream>
using namespace std;
const int N = 110;
int m,x;
bool st[N];

bool is_prime(int x){
    if(x < 2) return false;
    for(int i = 2; i <= x/i ; i++){
        if(x % i == 0) return false;
    }
    return true;
}

int main(){
    scanf("%d", &m);
    while(m -- ){
        scanf("%d", &x);
        if(is_prime(x)) puts("Yes");
        else puts("No");
    }
}
```

4.1.2 分解质因数

```
#include<iostream>
using namespace std;
const int N = 110;
```

```
int n, x;


void get_prime(int x){
    for(int i = 2; i <= x/i ; i++){
        if(x % i == 0){
            int s = 0;
            while(x % i == 0){
                x /= i;
                s++;
            }
            cout << i << ' ' << s <<endl;
        }
    }
    if(x > 1) cout << x <<' ' << 1 << endl;
    cout << endl;
}

int main(){
    scanf("%d",&n);
    while(n --){
        scanf("%d", &x);
        get_prime(x);
    }
    return 0;
}
```

4.1.3 筛质数

```
#include<iostream>
using namespace std;

const int N = 1e6 +10;
int primes[N];
bool st[N];
int cnt;

void get_primes(int n){
    for(int i = 2; i <= n ; i++){
        if(!st[i]) primes[cnt++] = i;
        for(int j = 0 ; primes[j] <= n/i ; j++){
            st[primes[j] * i] = true;
            if(i % primes[j] == 0) break;
        }
    }
}

int main(){
    int n;
    scanf("%d", &n);
    get_primes(n);
    printf("%d\n",cnt);
```

```
}
```

4.2 约数

4.2.1 试除法求约数

```cpp
#include<bits/stdc++.h>
using namespace std;

vector<int> get_divisor(int x) {
    vector<int> res;
    for (int i = 1; i <= x / i; ++ i )
        if (x % i == 0) {
            res.push_back(i);
            if (i != x / i) res.push_back(x / i);
        }
    sort(res.begin(), res.end());
    return res;
}

int main() {
    int n;
    cin >> n;

    while (n -- ) {
        int x;
        cin >> x;
        auto res = get_divisor(x);

        for (auto x : res) cout << x << ' ';
        cout << endl;
    }
    return 0;
}
```

4.2.2 约数个数

```cpp
#include<iostream>
#include<unordered_map>
using namespace std;
const int MOD = 1e9+7;
typedef long long LL;
unordered_map<int,int> primes;
int n = 0;

int main(){
    scanf("%d",&n);
    LL res = 1;
    while(n--){
        int x;
        scanf("%d",&x);
        for(int i = 2 ; i <= x / i ; i++){
```

```
            while(x % i == 0){
                x /= i;
                primes[i]++;
            }
        }
        if( x > 1) primes[x] ++;
    }
    for(auto p : primes) res = res * (p.second + 1) % MOD;
    printf("%lld\n", res);
    return 0;

}
```

4.2.3 约数之和

```
#include <iostream>
#include <unordered_map>
using namespace std;

const int MOD = 1e9 + 7;
typedef long long LL;
int n;
unordered_map<int, int> primes;

int main(){
    scanf("%d",&n);
    while(n--){
        int x;
        scanf("%d", &x);
        for(int i = 2 ; i <= x/i ; i++){
            while(x % i == 0){
                x /= i ;
                primes[i] ++;
            }
        }
        if(x > 1) primes[x] ++;
    }
    LL res = 1;
    for(auto x : primes){
        int p = x.first, m = x.second;
        LL t = 1;
        while(m --) t = (t * p + 1) % MOD;
        res = res * t % MOD;
    }
    printf("%lld", res);
    return 0;
}
```

4.2.4 最大公约数

```
#include<iostream>
```

```cpp
#include<algorithm>
using namespace std;

int gcd(int a, int b){
    if(a < b) swap(a,b);
    return b ? gcd(b, a%b) : a;
}

int n;

int main(){
    scanf("%d",&n);
    while(n--){
        int a, b;
        scanf("%d%d",&a, &b);
        printf("%d\n", gcd(a,b));
    }
    return 0;
}
```

4.3 欧拉函数

4.3.1 欧拉函数

```cpp
#include<iostream>
using namespace std;
const int N = 1e4 + 10;
int primes[N], n, phi[N];

int main(){
    scanf("%d", &n);
    while(n -- ){
        int x;
        scanf("%d", &x);
        int res = x;
        for(int i = 2 ; i <= x / i ; i++){
           if(x % i == 0){
               res = res / i * (i-1); // 防止溢出;
               while(x % i == 0) x /= i;
           }
        }
        if( x > 1) res = res / x * (x-1);
        printf("%d\n", res);
    }
}
```

4.3.2 筛法求欧拉函数

```cpp
#include<iostream>
using namespace std;

const int N = 1e6 + 10;
```

```

typedef long long LL;
int phi[N], primes[N],cnt;
bool st[N];

LL get_eulers(int n){
    phi[1] = 1;
    for(int i = 2; i <= n ; i++){
        if(!st[i]){
            primes[cnt++] = i;
            phi[i] = i -1;
        }
        for(int j = 0 ; primes[j] <= n / i ; j ++){
            int t = primes[j] * i;
            st[t] = true;
            if(i % primes[j] == 0){
                phi[t] = phi[i] * primes[j];
                break;
            }
            phi[t] = phi[i] * (primes[j] - 1);
        }
    }
    LL res = 0;
    for(int i = 1; i<=n ;i++){
        res += phi[i];
    }
    return res;
}

int main(){
    int n;
    scanf("%d", &n);
    printf("%lld\n",get_eulers(n));
}
```

4.4 快速幂

4.4.1 快速幂

```
#include<iostream>
using namespace std;

int n, a, b ,p;
typedef long long LL;

LL qmi(int a,int b, int p){
    LL res = 1;
    while(b){
        if(b & 1){
            res = (LL)res * a % p;
        }
        a = (LL) a * a % p;
```

```
        b >>= 1;
    }
    return res;
}

int main(){
    scanf("%d", &n);
    while(n--){
        scanf("%d%d%d", &a, &b, &p);
        printf("%lld\n", qmi(a,b,p));
    }
}
```

4.4.2 快速幂求逆元

```
#include<iostream>
using namespace std;

typedef long long LL;

LL pmi(int a, int b, int p){
    LL res = 1;
    while(b){
        if(b&1){
            res = (LL)res * a % p;
        }
        a = (LL) a * a % p;
        b >>=1;
    }
    return res;
}

int main(){
    int n ;
    scanf("%d",&n);
    while(n -- ){
        int a, p;
        scanf("%d%d",&a,&p);
        LL res = pmi(a, p -2 , p);
        if(a % p) printf("%lld\n", res);
        else puts("impossible");
    }

}
```

4.5 扩展欧几里得算法

4.5.1 扩展欧几里得算法

```
#include<iostream>
using namespace std;

```

```
/*
(a,b) = (b, a%b) ==>
ax + by = d
(a - a//b * b)x + by = d
ax + b(y - a //b *x) = d
y' = y - a//b *x
*/
int exgcd(int a, int b, int &x, int &y){
    if(!b){
        x = 1, y = 0;
        return a;
    }
    int d = exgcd(b,a%b,y,x);
    y -= (a/b)*x;
    return d;
}

int main(){
    int n;
    scanf("%d",&n);
    while(n--){
        int a, b, x ,y;
        scanf("%d%d",&a,&b);
        exgcd(a,b,x,y);
        printf("%d %d\n",x,y);
    }
    return 0;
}
```

4.5.2 线性同余方程

```
/*
        a * x === b % m
    ==> ax = my + b
    ==> ax - my = b
    ==> ax + my'= b
        gcd(a, m) | b 则有解
     x = x0 * b / d % m 相当于倍增
*/
#include<bits/stdc++.h>
using namespace std;

using LL = long long;

int exgcd(int a, int b, int & x, int & y) {
    if (!b) {
        x = 1, y = 0;
        return a;
    }
    int d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
```

```
}

int main() {
    int n;
    cin >> n;
    while (n -- ) {
        int a, b, m;
        cin >> a >> b >> m;
        int x, y;
        int d = exgcd(a, m, x, y);
        if (b % d) cout << "impossible" << endl;
        else cout << (LL)b / d * x % m << endl;
    }
}
```

4.6 中国剩余定理

4.6.1 表达整数的奇怪方式

```
#include<iostream>
using namespace std;

typedef long long LL;

LL exgcd(LL a, LL b, LL &x, LL &y){
    if(!b){
        x = 1, y = 0;
        return a;
    }
    LL d = exgcd(b, a%b, y, x);
    y -= (a/b) * x;
    return d;
}

int main(){
    int n;
    LL a1,m1;
    scanf("%d",&n);
    scanf("%lld%lld", &a1, &m1);
    bool has_ans = true;
    for(int i = 0 ; i < n - 1 ;i++){
        LL a2,m2,k1,k2;
        scanf("%lld%lld", &a2, &m2);
        LL d = exgcd(a1,a2,k1,k2);
        if((m2-m1) % d){
            has_ans = false;
            break;
        }
        int t = a2 / d;
        k1 *= (m2-m1) / d;
        k1 = (k1 % t + t) % t;

```

```
        m1 = a1 * k1 + m1;
        a1 = abs(a1 / d * a2);
    }
    if(has_ans) printf("%lld\n", (m1 % a1 + a1) % a1);
    else printf("%d\n",-1);
    return 0;
}
```

4.7 高斯消元

4.7.1 高斯消元解线性方程组

4.7.2 高斯消元解异或线性方程组

```
#include<iostream>
using namespace std;
const int N = 110;

int n,a[N][N];


int solve(){
    int r, c;
    for(r=0,c=0; c < n; c ++){
        int t = r;
        for(int i = r ; i < n ; i++)
            if(a[i][c]){
                t = i;
                break;
            }

        if(!a[t][c]) continue;

        for(int i = c; i < n + 1; i++) swap(a[t][i], a[r][i]);

        for(int i = r + 1; i < n; i++)
            if(a[i][c]){
                for(int j = n ; j >= c; j --)
                    a[i][j] ^= a[r][j];
            }
        r ++;
    }
    if(r < n){
        for(int i = r ; i < n; i++)
            if(a[i][n]) return 2;
        return 1;
    }
    for(int i = n -1 ; i >=0 ; i--)
        for(int j = i + 1; j < n ; j++)
            a[i][n] ^= a[i][j] & a[j][n];
    return 0;
```

```
}

int main(){
    cin >> n;
    for(int i = 0 ; i < n ; i++)
        for(int j = 0 ; j < n + 1; j++)
            cin >> a[i][j];

    int t = solve();
    if(t ==0){
        for(int i = 0; i < n ;i ++)
            cout << a[i][n] << endl;
    }
    else if(t == 1) puts("Multiple sets of solutions");
    else puts("No solution");
    return 0;
}
```

4.8 求组合数

4.8.1 上三角法

```
#include<iostream>
using namespace std;
const int N = 2010, mod = 1e9 + 7;

int C[N][N], n;

void solve(){
    for(int a = 0 ; a < N; a ++)
        for(int b = 0 ; b <= a ; b++){
            if(!b) C[a][b] = 1;
            else{
                C[a][b] = (C[a-1][b] + C[a-1][b-1]) % mod;
            }
        }
}

int main(){
    cin >> n;
    int a, b;
    solve();
    while(n--){
        cin >> a >> b;
        cout << C[a][b] <<endl;
    }
    return 0;
}
```

4.8.2 小费马定理与逆元法

```cpp
#include<iostream>
#include<algorithm>
#include<string.h>
using namespace std;

typedef long long LL;

const int N = 1e5 + 10, mod = 1e9 + 7;

LL qmi(LL a, LL k, LL p){
    LL res =1;
    while(k){
        if(k & 1) res = (LL) res * a % p;
        a = (LL) a * a % p;
        k >>= 1;
    }
    return res;
}

int main(){
    int n;
    cin >> n;
    LL fact[N], infact[N];
    fact[0] = infact[0] = 1;
    for(int i =1 ; i < N ;i ++){
        fact[i] = fact[i-1] * i % mod;
        infact[i] = infact[i-1] * qmi(i, mod - 2, mod) % mod;
    }
    LL a, b;
    while(n--){
       cin >> a >> b;
       cout << fact[a] * infact[b] % mod * infact[a-b] % mod << endl;
    }
    return 0;
}
```

4.8.3 卢卡斯定理

```cpp
#include<iostream>
using namespace std;

typedef long long LL;

LL qmi(LL a ,LL k, LL p){
    LL res = 1;
    while(k){
       if(k & 1) res = (LL) res * a % p;
       a = (LL) a * a % p;
       k >>= 1;
    }
    return res;
}
```

```

LL C(LL a, LL b, LL p){
    LL res = 1;
    for(int i = 1, j = a ; i <= b; i++, j--){
        res = (LL)res * j % p;
        res = (LL) res * qmi(i, p-2 ,p) % p;
    }
    return res;
}

LL lucica(LL a, LL b, LL p){
   if(a < p && b < p) return C(a,b,p);
   return C(a % p, b % p, p) * lucica(a / p, b / p, p) % p;
}

int main(){
    int n;
    LL a,b,p;
    scanf("%d", &n);
    while(n--){
        scanf("%lld%lld%lld", &a,&b,&p);
        printf("%lld\n", lucica(a,b,p));
    }
}
```

4.8.4 高精度

```
#include <iostream>
#include<vector>
using namespace std;

const int N = 5010;
typedef long long LL;

int primes[N],sum[N], cnt=0;
bool st[N];

int get(int n, int p){
    int res = 0;
    while(n){
        res += n / p;
        n /= p;
    }
    return res;
}

void get_primes(int n){
    for(int i = 2 ; i <= n; i++){
        if(!st[i]) primes[cnt++] = i;
        for(int j = 0 ; primes[j] <= n / i ; j ++){
            st[primes[j] * i] = true;
            if(i % primes[j] == 0) break;
```

```
        }
    }
}

vector<int> mul(vector<int> &a , int b){
    vector<int> c ;
    int t = 0;
    for(int i = 0; i < a.size(); i++){
        t += a[i] * b;
        c.push_back(t%10);
        t /= 10;
    }
    while(t) {
        c.push_back(t % 10);
        t /= 10;
    }
    return c;
}

int main(){
    int a, b;
    scanf("%d%d",&a,&b);
    get_primes(a);
    for(int i = 0 ; i < cnt; i++){
        int p = primes[i];
        sum[i] = get(a,p) - get(a-b,p) - get(b,p);
    }
    vector<int> res;
    res.push_back(1);
    for(int i = 0 ; i < cnt; i++)
        for(int j = 0; j < sum[i]; j ++){
            int p = primes[i];
            res = mul(res, p);
        }

    for(int i = res.size() -1; i>=0 ; i--)
        printf("%d",res[i]);
    return 0;
}
```

4.8.5 卡特兰数

```
#include <iostream>
#include <algorithm>
using namespace std;

typedef long long LL;
const LL mod = 1e9 + 7;


typedef long long LL;

```

```

LL qmi(LL a, LL k, LL p){
    LL res = 1;
    while(k){
        if(k & 1) res = (LL) res * a % p ;
        a = (LL) a * a % p;
        k >>= 1;
    }
    return res;
}

int main(){
    int n;
    scanf("%d", &n);
    int a = 2 *n, b = n;
    LL res =1 ;
    for(int i = a; i > a- b ; i--) res = (LL) res * i % mod;
    for(int i = 1; i <= b ; i++) res = (LL) res * qmi(i, mod - 2, mod) % mod;
    res = (LL) res * qmi(n+1, mod - 2, mod) % mod;
    printf("%lld\n",res);
    scanf("%d", &n);

}
```

4.9 容斥原理

4.9.1 能被整除的数

```
#include<iostream>
using namespace std;

const int N = 20;
typedef long long LL;
int p[N];
int main(){
    int n, m;
    scanf("%d%d",&n,&m);
    int res = 0;
    for(int i = 0 ; i < m; i++) scanf("%d", &p[i]);
    for(int i = 1 ; i < 1 << m; i++){
        int t = 1, cnt = 0;
        for(int j = 0 ; j < m; j++){
            if(i >> j & 1){
                if((LL)t * p[j] > n){
                    t = -1;
                    break;
                }
                cnt++;
                t *= p[j];
            }
        }
        if(t != -1){
```

```
25          if(cnt % 2) res += n / t;
26          else res -= n / t;
27      }
28   }
29   printf("%d\n", res);
30   return 0;
31}
```

4.10 博弈论

4.10.1 Nim 游戏

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6    int n;
7    scanf("%d",&n);
8    int x;
9    scanf("%d", &x);
10   for(int i = 0; i < n -1; i++){
11      int y;
12      scanf("%d",&y);
13      x ^=y;
14   }
15   if(x) puts("Yes");
16   else puts("No");
17   return 0;
18}
```

4.10.2 台阶-Nim 游戏

```
1 #include<iostream>
2 using namespace std;
3
4
5 int main(){
6    int n;
7    scanf("%d",&n);
8    int res = 0;
9    for(int i = 1; i <= n ; i ++){
10      int x;
11      scanf("%d",&x);
12      if(i%2) res ^= x;
13   }
14   if(res) puts("Yes");
15   else puts("No");
16   return 0;
17}
```

4.10.3 集合-Nim 游戏

```cpp
#include<iostream>
#include<algorithm>
#include<unordered_set>
#include<string.h>

using namespace std;

const int N = 110, M = 10010;
int s[N], f[M], n, k;


int sg(int x){
    if(f[x] != -1) return f[x];

    unordered_set<int> S;
    for(int i = 0; i < k; i++){
        if(x >= s[i]) S.insert(sg(x - s[i]));
    }
    for(int i = 0; i < n; i++){
        if(!S.count(i)) {
            return f[x] = i;
        }
    }
}

int main(){
    memset(f, -1, sizeof f);
    scanf("%d", &k);
    for(int i = 0 ; i < k; i ++) scanf("%d", &s[i]);
    int res = 0;
    scanf("%d", &n);
    for(int i = 0 ; i < n; i ++) {
       int x;
       scanf("%d", &x);
       res ^= sg(x);
    }
    if(res) puts("Yes");
    else puts("No");
    return 0;
}
```

4.10.4 拆分-Nim 游戏

```cpp
#include<iostream>
#include<unordered_set>
#include<algorithm>
#include<string.h>
using namespace std;
const int N = 110;
int s[N],f[N], n;
```

```cpp
int sg(int x){
    if(f[x]!=-1) return f[x];

    unordered_set<int> S;

    for(int i = 0 ; i < x; i++)
        for(int j = 0 ; j <= i; j++)
            S.insert(sg(i) ^ sg(j));

    for(int i = 0; ; i++)
        if(!S.count(i)) return f[x] = i;
}

int main(){
    scanf("%d", &n);
    int res = 0;
    memset(f, -1, sizeof f);
    for(int i = 0 ; i < n; i++){
        int x;
        scanf("%d", &x);
        res ^= sg(x);
    }
    if(res) puts("Yes");
    else puts("No");
}
```

5 动态规划

5.1 背包问题

5.1.1 01 背包问题

```cpp
#include<iostream>
using namespace std;

const int N = 1010, M = 1010;
int n,m,f[M], v[N], w[N];

int main(){
    cin >> n >> m;
    for(int i = 1; i <=n; i++){
        cin >> v[i] >> w[i];
    }

    for(int i = 1; i <=n ;i++)
        for(int j = m; j >= v[i]; j--)
            f[j] = max(f[j], f[j - v[i]] + w[i]);
    cout << f[m] << endl;
    return 0;
}
```

5.1.2 完全背包问题

```cpp
#include<iostream>
using namespace std;


const int N = 1010, M = 1010;
// int n,m, f[N][M], v[N], w[N];
int n,m, f[M], v[N], w[N];

int main(){
    cin >> n >> m;
    for(int i = 1 ; i<= n ; i++) cin >> v[i] >> w[i];

    for(int i = 1; i<=n ; i++)
        for(int j = v[i]; j <=m; j++)
            f[j] = max(f[j], f[j - v[i]] + w[i]);

    // for(int i = 1; i <= n; i++)
    // for(int j = 0; j <= m; j++)
    // for(int k = 0; k * v[i] <= j; k++){
    // f[i][j] = max(f[i][j], f[i-1][j - k*v[i]] + k * w[i]);
    // }

    cout << f[m] <<endl;
}
```

5.1.3 多重背包问题

```cpp
#include<iostream>
using namespace std;
const int N = 110, M = 110;
int n, m , f[N][M], v[N], w[N], s[N];

int main(){
    cin >> n >> m;
    for(int i = 1 ; i <=n ; i++) cin >> v[i] >> w[i] >> s[i];

    for(int i = 1; i<=n; i++)
        for(int j = 0; j <=m ; j++)
            for(int k = 0; k <=s[i] && k * v[i] <= j; k++)
                f[i][j]= max(f[i][j], f[i-1][j - k * v[i]] + k * w[i]);
    cout << f[n][m] << endl;
}
```

5.1.4 多重背包问题 2

```cpp
#include<iostream>
using namespace std;

const int N = 10010, M = 2010;
```

```

int n,m,cnt=0,f[M], v[N], w[N];

int main(){
    cin >> n >> m;
    for(int i = 1 ; i <=n; i++){
        int a,b,s;
        cin >> a >> b >> s;
        int k = 1;
        while(k <= s){
            v[++cnt] = k * a;
            w[cnt] = k * b;
            s -= k;
            k *=2;
        }
        if(s > 0){
            v[++cnt] = s * a;
            w[cnt] = s * b;
        }
    }

    for(int i = 1; i <= cnt; i ++)
        for(int j = m; j >= v[i]; j--)
            f[j] = max(f[j], f[j - v[i]] + w[i]);

    cout << f[m] <<endl;
    return 0;
}
```

5.1.5 分组背包问题

```
#include<iostream>
using namespace std;
const int N = 110, M = 110;
int n,m,f[M],w[N],v[N],s[N];

int main(){
    cin >> n >> m;
    for(int i = 1; i<=n; i++){
        cin >> s[i];
        for(int j = 1; j <=s[i]; j++) cin >> v[j] >> w[j];
        for(int j = m; j >=0; j--)
            for(int k = 1; k <= s[i]; k++)
                if(j >= v[k])
                    f[j] = max(f[j], f[j - v[k]] + w[k]);
    }

    cout << f[m] << endl;
    return 0;
}
```

5.2 线性 DP

5.2.1 数字三角形

```cpp
#include<iostream>
using namespace std;

const int N = 510;

int n,a[N][N], f[N][N], INF = 1e9;


int main(){
    cin >> n;
    for(int i = 1 ; i <=n; i++)
        for(int j = 1; j <=i; j++)
            cin >> a[i][j];
    for(int i = 1; i <=n; i++)
        for(int j = 0; j <= i+1; j++)
            f[i][j] = -INF;
    f[1][1] = a[1][1];
    for(int i = 1; i <=n ; i++)
        for(int j = 1; j <= i; j++)
            f[i][j] = max(f[i-1][j-1] + a[i][j], f[i-1][j] + a[i][j]);

    int res = -INF;
    for(int i = 1; i<=n; i++)
        res = max(res, f[n][i]);

    cout << res <<endl;
    return 0;
}
```

5.2.2 最长上升子序列

```cpp
#include<iostream>
using namespace std;
const int N = 1010;
int a[N],f[N];

int main(){
    int n;
    cin >> n;
    int res = 1;
    for(int i = 1 ; i<=n; i++) cin >> a[i];
    for(int i = 1; i <=n ; i++){
        f[i] =1;
        for(int j = 1; j < i; j ++){
            if(a[i] > a[j]) f[i] = max(f[i], f[j] + 1);
        }
        res = max(res, f[i]);
    }
    cout << res << endl;
```

```
    return 0;
}
```

5.2.3 最长上升子序列 2

```
#include<iostream>
using namespace std;

const int N = 100010;
int a[N], q[N], n;

int main(){
    scanf("%d", &n);
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    q[0] = -1e9;
    int len = 0;
    for(int i = 0; i < n; i++){
        int l = 0, r = len;
        while(l < r){
            int mid = l + r + 1 >>1;
            if(q[mid] < a[i]) l = mid;
            else r = mid -1;
        }
        len = max(len, r+1);
        q[r+1] = a[i];
    }
    cout << len << endl;
}
```

5.2.4 最长公共子序列

```
#include<iostream>
using namespace std;


const int N = 1010, M = 1010;

char a[N], b[M];
int n,m,f[N][M];

int main(){
    scanf("%d%d", &n, &m);
    scanf("%s%s", a + 1, b + 1);

    for(int i = 1; i <= n ;i ++)
        for(int j =1 ; j <=m ;j ++){
            f[i][j] = max(f[i-1][j], f[i][j-1]);
            if(a[i] == b[j])
                f[i][j] = max(f[i][j], f[i-1][j-1] + 1);
        }
    cout << f[n][m] <<endl;
```

```
}
```

5.2.5 最短编辑距离

```cpp
#include<iostream>
using namespace std;

const int N = 1010, M = 1010;
int n, m, f[N][M];
char a[N], b[M];

int main(){
    scanf("%d%s", &n, a + 1);
    scanf("%d%s", &m ,b + 1);

    for(int i =1 ; i<=n; i++) f[i][0] = i;
    for(int i = 1; i <=m; i++) f[0][i] = i;

    for(int i =1 ; i <=n ; i++)
        for(int j = 1 ; j <=m ; j++){
            f[i][j] = min(f[i-1][j] + 1, f[i][j-1] + 1);
            if(a[i] == b[j])
                f[i][j] = min(f[i][j], f[i-1][j-1]);
            else
                f[i][j] = min(f[i][j], f[i-1][j-1] + 1);
        }
    printf("%d\n", f[n][m]);
    return 0;
}
```

5.3 区间 DP

5.3.1 石子合并

```cpp
#include<iostream>
using namespace std;

const int N = 310;
int n, m, f[N][N], s[N];

int main(){
    cin >> n;
    for(int i = 1 ; i<=n; i++) cin >> s[i];
    for(int i = 1; i <=n ; i++) s[i] += s[i-1];
    for(int len = 2; len <= n; len ++){
      for(int l = 1; l + len - 1 <= n; l++){
          int r = l + len -1;
          f[l][r] = 1e9;
          for(int k = l; k < r; k++)
              f[l][r] = min(f[l][r], f[l][k] + f[k+1][r] + s[r] - s[l-1]);
      }
    }
```

```
    cout << f[1][n] << endl;
    return 0;
}
```

5.4 计数类 DP

5.4.1 整数划分

```
/*
f[i][j] = f[i-1][j] + f[i-1][j-i] + f[i-1][j-2*i] + ... f[i-1][j-s * i];
f[i][j-i] = f[i-1][j-i] + f[i-1][j-2*i] + ... f[i-1][j - s * i];
f[i][j] = f[i-1][j] + f[i][j-i];
*/
#include<iostream>
using namespace std;
const int N = 1010, mod = 1e9 + 7;
int n, f[N];
int main(){
    scanf("%d", &n);
    f[0] = 1;
    for(int i = 1; i <=n ; i++)
        for(int j = i; j <=n; j++)
            f[j] = (f[j] + f[j - i]) % mod;
    printf("%d\n", f[n]);
    return 0;
}
```

5.5 数位统计 DP

5.5.1 计数问题

```
#include<iostream>
#include<cstring>
using namespace std;
const int N = 100;
int dp[N][N][2][2], digit[N];

/*
    x: 统计的数字
    l: 第几位
    lead: 前导0
    sum: 当前整数包含x的个数
    limit: 是否达到上限
*/
int f(int x, int l, int sum, bool lead, bool limit){
    // 当前数字统计完了
    if(!l) return sum;
    // 记忆化搜索
    if(dp[l][sum][lead][limit] != -1) return dp[l][sum][lead][limit];
    int res =0;
    int k = limit ? digit[l] : 9;
    for(int i = 0; i <= k; i ++){
```

```
22        bool ne_limit = limit && (i == k);
23        if(lead && !i) // 含前导0, 跳过继续搜
24            res += f(x, l-1, sum, true, ne_limit);
25        else // 不含前导零, 判断当前位是否为x
26            res += f(x, l-1, sum + (i == x), false, ne_limit);
27    }
28    // 返回第1~l位中的统计结果
29    return dp[l][sum][lead][limit] = res;
30 }
31
32 int solve(int n, int x){
33    int len = 0;
34    memset(dp, -1 ,sizeof dp);
35    while(n){
36        digit[++len] = n % 10;
37        n /= 10;
38    }
39    return f(x, len, 0, true, true);
40 }
41
42 int main(){
43    int a, b;
44    while(scanf("%d%d",&a, &b) && a && b){
45        if(a > b) swap(a, b);
46        for(int i = 0 ; i <= 9; i++)
47            printf("%d ", solve(b,i) - solve(a-1,i));
48        puts("");
49    }
50    return 0;
51 }
```

5.6 状态压缩 DP

5.6.1 最短 Hamilton 路径

```
1  #include<iostream>
2  #include<cstring>
3  using namespace std;
4  const int N = 21, M = 1 << N;
5  int dp[M][N], a[N][N], n;
6
7  int main(){
8     scanf("%d", &n);
9     for(int i = 0; i < n; i++)
10        for(int j = 0; j < n; j++)
11            scanf("%d", &a[i][j]);
12    memset(dp, 0x3f, sizeof dp);
13    dp[1][0] = 0;
14    for(int i= 0; i < 1 << n; i++)
15        for(int j = 0; j < n; j++)
16            if(i >> j & 1){
17                for(int k = 0; k < n; k ++)
```

```
                if(i ^ 1 << j >> k & 1){
                    dp[i][j] = min(dp[i][j], dp[i ^ 1 << j][k] + a[k][j]);
                }
            }
    printf("%d\n", dp[(1 << n) - 1][n-1]);
    return 0;
}
```

5.6.2 蒙德里安的梦想

```
#include<iostream>
#include<cstring>
using namespace std;
const int N = 12, M = 1 << N;
int n,m;
long long f[N][M];
bool st[M];

int main(){
    while(scanf("%d%d", &n, &m) && (n || m)){
        memset(f, 0, sizeof f);
        memset(st, 0, sizeof st);
        for(int i = 0; i < 1 << n; i++){
            int cnt = 0;
            st[i] = true;
            for(int j = 0; j < n;j ++){
                if(i >> j & 1){
                    if(cnt & 1) st[i] = false;
                    cnt = 0;
                }
                else cnt ++;
            }
            if(cnt & 1) st[i] = false;
        }
        f[0][0] = 1;
        for(int i =1; i<=m; i++)
            for(int j = 0; j < 1 << n; j++)
                for(int k = 0; k < 1 << n; k++)
                    if((j & k) == 0 && st[j | k])
                        f[i][j] += f[i-1][k];
        printf("%lld\n", f[m][0]);
    }
    return 0;
}
```

5.7 树形 DP

5.7.1 没有上司的舞会

```
#include<iostream>
#include<cstring>
using namespace std;
```

```cpp
const int N = 6010;
int dp[N][2], happy[N], el[N], ne[N], h[N], idx = 0, n;
bool has_father[N];

void insert(int x, int y){
    el[idx] = y, ne[idx] = h[x], h[x] = idx ++;
}

int dfs(int u){
    dp[u][1] = happy[u]; // 初始化，只选u的快乐值
    for(int i = h[u]; i!=-1; i = ne[i]){
        int j = el[i];
        dfs(j);
        dp[u][0] += max(dp[j][0], dp[j][1]);
        dp[u][1] += dp[j][0];
    }
    return max(dp[u][0], dp[u][1]);
}

int main(){
    memset(h, -1, sizeof h);
    scanf("%d", &n);
    for(int i = 1; i <=n ;i ++) scanf("%d", &happy[i]);
    for(int i = 0; i < n-1; i++){
        int l,k;
        scanf("%d%d", &l, &k);
        insert(k, l); // k是l的祖先，符合头插法
        has_father[l] = true;
    }
    int root =1 ;
    while(has_father[root]) root++;
    printf("%d\n", dfs(root));
    return 0;
}
```

5.8 记忆化搜索

5.8.1 滑雪

```cpp
#include<iostream>
#include<cstring>
using namespace std;
const int N = 310, M=310;
int h[N][M], f[N][M], n, m;
int dx[4] = {-1,0,1,0}, dy[4] = {0,1,0,-1};

int dfs(int x, int y){
    int &v = f[x][y];
    if(v != -1) return v;
    v = 1;
    for(int i=0; i < 4; i++){
        int a = x + dx[i], b = y + dy[i];
```

```
14          if(a<=n && a >= 1 && b <=m && b >=1 && h[x][y] > h[a][b])
15              v = max(v, dfs(a,b) + 1);
16      }
17      return v;
18  }
19
20  int main(){
21      memset(f, -1, sizeof f);
22      scanf("%d%d", &n, &m);
23      for(int i = 1; i <= n; i++)
24          for(int j = 1; j <=m; j++)
25              scanf("%d", &h[i][j]);
26
27      int res = 0;
28      for(int i = 1; i <=n; i++)
29          for(int j = 1; j<=m; j++)
30              res = max(res, dfs(i,j));
31      printf("%d\n", res);
32  }
```