卡特兰数

h(n)= h(0)*h(n-1) + h(1)*h(n-2) + ... + h(n-1)h(0) (其中n>=2, h(0)=h(1)=1)

$$h(0)=1, h(n+1)=\frac{2(2n+1)}{n+2}h(n)$$

$$h(n)=\frac{C_{2n}^{n}}{n+1}$$

K. K-skip Permutation

time limit per test: 1.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

For a permutation $P=p_1,p_2,\cdots,p_n$ of n, let $f(P,k)$ be the number of i satisfying $1\le i<n$ and $p_i+k=p_{i+1}$.

Given two integers n and k, your task is to find a permutation P of n such that $f(P,k)$ is maximized.

Recall that in a permutation of n, each integer from 1 to n (both inclusive) appears exactly once.

Input

There is only one test case in each test file.

The first and only line contains two integers n and k ($1\le n,k\le 10^6$).

Output

Output one line containing n integers indicating a permutation P of n that maximizes $f(P,k)$. If there are multiple valid answers you can output any of them.

Please, DO NOT output extra spaces at the end of the line, or your answer may be considered incorrect!

Examples

input	Copy
3 1	
output	Copy
1 2 3	

input	Copy
7 3	
output	Copy
2 5 1 4 7 3 6	

input	Copy
3 7	
output	Copy
1 3 2	

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

int main()
{
    int k, n;
    cin >> n >> k;
    vector<bool>a(n + 1, 1);
    int cnt = 0;
    for (int i = 1;i <= n;i++)
```

```
    {
        if (a[i])
        {
            cout << i;
            a[i] = 0;
            cnt++;
            if (cnt != n)
            {
                cout << ' ';
            }
            bool f = 1;
            int x = i;
            while (f)
            {
                f = 0;
                if (x + k <= n && a[x + k])
                {
                    f = 1;
                    x += k;
                    cout << x;
                    a[x] = 0;
                    cnt++;
                    if (cnt != n)
                    {
                        cout << ' ';
                    }
                }
            }
        }
    }
    return 0;
}
```

题目描述

有一个箱子容量为V（正整数，0 ≤ V ≤ 20000），同时有n个物品（0 < n ≤ 30），每个物品有一个体积（正整数）。
要求n个物品中，任取若干个装入箱内，使箱子的剩余空间为最小。

输入描述:

> 1个整数，表示箱子容量
> 1个整数，表示有n个物品
> 接下来n行，分别表示这n个物品的各自体积

输出描述:

> 1个整数，表示箱子剩余空间。

```
#include<bits/stdc++.h>
using namespace std;
int f[35][20003];//前n个物品放在V中的最大体积
```

```
int main(){
    int V,n;
    cin>>V>>n;
    int a[31];
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    memset(f,0,sizeof(f));
    for(int i=1;i<=n;i++){
        for(int j=1;j<=V;j++){
            if(j-a[i]>=0){
                f[i][j]=max(f[i-1][j-a[i]]+a[i],f[i-1][j]);
            }
            else{
                f[i][j] = f[i - 1][j];
            }

        }
    }
    cout<<V-f[n][V];
    return 0;
}
```

优化为一维 //必须倒序遍历j

```
#include<bits/stdc++.h>
using namespace std;
int f[20001];//f[v]的意思为：容量为v时，能装的最大体积。
int main(){
    int V,n;
    cin>>V>>n;
    int a[31];
    for(int i = 0;i<n;i++){
        cin>>a[i];
    }
    memset(f,0,sizeof(f));
    for(int i = 0;i < n ;i++)
    {
        for(int j = V;j >= a[i];j--)
        {
            f[j] = max(f[j],f[j-a[i]] + a[i]);
        }
    }
    cout<<V-f[n];
    return 0;
}
```

若要恰好装满

你有一个背包，最多能容纳的体积是V。

现在有n个物品，第i个物品的体积为v_i ,价值为w_i。

（1）求这个背包至多能装多大价值的物品?
（2）若背包**恰好装满**，求至多能装多大价值的物品?

输入描述:

第一行两个整数n和V，表示物品个数和背包体积。
接下来n行，每行两个数v_i和w_i，表示第i个物品的体积和价值。

$1 \leq n, V, v_i, w_i \leq 1000$

输出描述:

输出有两行，第一行输出第一问的答案，第二行输出第二问的答案，如果无解请输出0。

示例1

输入 复制

```
3 5
2 10
4 5
1 4
```

输出 复制

```
14
9
```

```cpp
#include<iostream>
using namespace std;
int n,V;
int w[1005];
int v[1005];
int dp[1005][1005];
int f[1005];
#define INF 0x80000000;  //inf为一个极小值（-212874284...）
int main(){
    cin>>n>>V;
    for(int i=1;i<=n;i++){
        cin>>v[i]>>w[i];
    }
    for(int i=1;i<=n;i++){
        for(int j=V;j>=0;j--){
            if(j>=v[i]){
                dp[i][j]=max(dp[i-1][j-v[i]]+w[i],dp[i-1][j]);
            }
            else{
```

```
                dp[i][j]=dp[i-1][j];
            }
        }
    }
    cout<<dp[n][V]<<endl;
    for(int i=0;i<=1005;i++){
        f[i]=INF;
    }
    f[0]=0;  //标明状态
    for(int i=1;i<=n;i++){
        for(int j=V;j>=v[i];j--){
            f[j]=max(f[j],f[j-v[i]]+w[i]);
            if(f[j]<0){
                f[j]=INF;
            }
        }
    }
    if(f[V]>0){
        cout<<f[V];
    }
    else{
        cout<<0;
    }
}
```

LCS最长公共子序列问题

牛可乐得到了两个字符串 s 和 t ,牛可乐想请聪明的你帮他计算出来，两个字符串的最长公共子序列长度是多少。

最长公共子序列的定义是，子序列中的每个字符都能在两个原串中找到，而且每个字符的先后顺序和原串中的先后顺序一致。

输入描述:

> 输入包含多组数据，请读至文件末尾。
> 每行包含两个字符串 s,t，两个字符串用一个空格字符间隔，单个字符串长度不超过 5000。
>
> 数据保证所有数据的字符串 s 长度之和与字符串 t 长度之和均不超过 5000。

输出描述:

> 对于每组数据，输出一个整数，代表最长公共子序列的长度。

示例1

输入 复制

```
abccde bcee
```

输出 复制

```
3
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int dp[5001][5001];
int main(){
    string s1,s2;
    while(cin>>s1>>s2){
       // if(s1[0]=s2[0]){
      //     dp[1][1]=1;
      //  }
        int ans = 0;
        for (int i=1;i<=s1.size();i++) {
            for (int j=1;j<=s2.size();j++) {
                if (s1[i-1]==s2[j-1]) {
                    dp[i][j] = dp[i-1][j-1]+1;
                } else {
                    dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
                }
            }
        }
        cout<<dp[s1.size()][s2.size()]<<endl;
    }
    return 0;
}
```

您将获得一个正整数 n 。找到满足以下条件的最长正整数序列 $a = [a_1, a_2, \ldots, a_k]$ ，并打印该序列：

- $a_i \le n$ for all $1 \le i \le k$.

  $a_i \le n$ 对于所有 $1 \le i \le k$ 。
- a is strictly increasing. That is, $a_i > a_{i-1}$ for all $2 \le i \le k$.

  a 严格递增。也就是说， $a_i > a_{i-1}$ 对于所有 $2 \le i \le k$ 。
- $a_i \mid a_{i-1} = n$ for all $2 \le i \le k$, where $\mid$ denotes the bitwise OR operation.

  $a_i \mid a_{i-1} = n$ 对于所有 $2 \le i \le k$ ，其中 $\mid$ 表示按位 OR手术。

Input 输入

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

每个测试包含多个测试用例。第一行包含测试用例的数量 t ($1 \le t \le 1000$)。测试用例的描述如下。

The only line of each test case contains one integer n ($1 \le n \le 10^{18}$).

每个测试用例的唯一行包含一个整数 n ($1 \le n \le 10^{18}$)。

It's guaranteed that the sum of lengths of the longest valid sequences does not exceed $5 \cdot 10^5$.

保证最长有效序列的长度总和不超过 $5 \cdot 10^5$ 。

Output 输出

For each testcase, print two lines. In the first line, print the length of your constructed sequence, k. In the second line, print k positive integers, denoting the sequence. If there are multiple longest sequences, you can print any of them.

对于每个测试用例，打印两行。在第一行中，打印构建的序列的长度 k 。在第二行中，打印 k 正整数，表示序列。如果有多个最长序列，您可以打印其中任何一个。

Example 例子

input 输入 Copy 复制

```
4
1
3
14
23
```

output 输出 Copy 复制

```
1
1
3
1 2 3
4
4 10 12 14
5
7 18 21 22 23
```

```cpp
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
typedef long long ll;
ll pow_(int x)
{
    ll y=1;
    for (int i = 0;i < x;i++)
        y *= 2;
    return y;
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin >> t;
    ll a[129],b[129];
    while (t--) {
        int n;
```

```
        ll cnt = 0;
        ll ans = 0;
        cin >> n;
        ll first = n;
        for (int i = 0;i <=128;i++) {
            a[i]= n & 1;
            n>>=1;
            if (a[i] == 1) {
                b[++cnt] = i;
            }
            cout << a[i] << " ";
        }
        if (cnt== 1)cout << cnt << endl;
        else { cout << cnt + 1 << endl;
        for (int i = cnt;i >= 1;i--) {
            for (int j = 128;j >= 0;j--) {
                if (b[i] == j)continue;
                ans += a[j]*pow_(j);
            }
            cout << ans << " ";
            ans = 0;
        }
        }
        cout << first <<endl;
        for (int i = 0;i < 128;i++) {
            a[i] = 0;
            b[i] = 0;
        }
    }
    return 0;
}
```

lowbit位运算

temp = x&(-x)为第一个1及之后的0，x-temp可得最大公因数，与x异或temp相同a

洛谷p5318　　//图的dfs bfs（存有向边的省内存方法）

```
#include<iostream>  //头文件头文件头文件
#include<vector>
#include<queue>
#include<algorithm>
using namespace std;
struct edge{       //存边结构体
    int u,v;
};
vector <int> e[100001];  //两个vector刚才已经详细讲过了
vector <edge> s;
bool vis1[100001]={0},vis2[100001]={0}; //标记数组
bool cmp(edge x,edge y){  //排序规则
    if(x.v==y.v)
    return x.u<y.u;
```

```
    else return x.v<y.v;
}
void dfs(int x){  //深度优先遍历
    vis1[x]=1;
    cout<<x<<" ";
    for(int i=0;i<e[x].size();i++){
        int point=s[e[x][i]].v;
        if(!vis1[point]){
            dfs(point);
        }
    }
}
void bfs(int x){  //广度优先遍历
    queue <int> q;
    q.push(x);
    cout<<x<<" ";
    vis2[x]=1;
    while(!q.empty()){
        int fro=q.front();
        for(int i=0;i<e[fro].size();i++){
            int point=s[e[fro][i]].v;
            if(!vis2[point]){
                q.push(point);
                cout<<point<<" ";
                vis2[point]=1;
            }
        }
        q.pop();
    }
}
int main(){
    int n,m;  //输入，存边
    cin>>n>>m;
    for(int i=0;i<m;i++){
        int uu,vv;
        cin>>uu>>vv;
        s.push_back((edge){uu,vv});
    }
    sort(s.begin(),s.end(),cmp);  //排序
    for(int i=0;i<m;i++)
        e[s[i].u].push_back(i);
    dfs(1);   //从1号顶点开始深搜
    cout<<endl;
    bfs(1);   //广搜亦同理
}
```

洛谷p3916

反向建边 + dfs

按题目来每次考虑每个点可以到达点编号最大的点，不如考虑较大的点可以反向到达哪些点

循环从N到1，则每个点i能访问到的结点的A值都是i

每个点访问一次，这个A值就是最优的，因为之后如果再访问到这个结点那么答案肯定没当前大了

存边数组则与上一题相同

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

#define MAXL 100010

int N, M, A[MAXL];
vector<int> G[MAXL]; //vector存图

void dfs(int x, int d) {
    if(A[x]) return; //访问过
    A[x] = d;
    for(int i=0; i<G[x].size(); i++)
        dfs(G[x][i], d);
}

int main() {
    int u, v;
    scanf("%d%d", &N, &M);
    for(int i=1; i<=M; i++) {
        scanf("%d%d", &u, &v);
        G[v].push_back(u); //反向建边
    }
    for(int i=N; i; i--) dfs(i, i);
    for(int i=1; i<=N; i++) printf("%d ", A[i]);
    printf("\n");
    return 0;
}
```

lower_bound属于C++内置STL的一种，可以实现在有序数组下进行二分查找，它可以用来找到第一个大于等于待查元素的值的位置。

没有大于等于待查元素的值函数会输出结束查询的位置

类似的还有upper_bound，可以找到第一个小于等于待查元素的值。

```
int c=lower_bound(开始查询的位置【闭区间】，结束查询的位置【开区间】， 查询的数)-数组指针;
e=lower_bound(a,b,c);
```

位运算

位运算符	解释
`lowbit(i)` 即 `i & -i`	返回 `i` 的最后一位1
`n>>k` & 1	求n的第k位数字
x | (1 << k)	将x第k位 置为1
x ^ (1 << k)	将x第k位取反
x & (x - 1)	将x最右边的1置为0(去掉最右边的1)
x | (x + 1)	将x最右边的0置为1
x & 1	判断奇偶性 真为奇，假为偶

lowbit两种实现方式

```
1 int lowbit(x)
2 {
3     return x - (x & (x - 1));
4 }
int lowbit(x)
{
    return x & -x;
}
```

You are given an array b of $n - 1$ integers.

给你一个数组 b 的 $n - 1$ 整数。

An array a of n integers is called *good* if $b_i = a_i \,\&\, a_{i+1}$ for $1 \le i \le n - 1$, where $\&$ denotes the bitwise AND operator.

数组 a 的 n 整数被称为*好的*，如果 $b_i = a_i \,\&\, a_{i+1}$ 为了 $1 \le i \le n - 1$， 在哪里 $\&$ 表示按位与运算符。

Construct a good array, or report that no good arrays exist.

构造一个好的数组，或者报告不存在好的数组。

Input 输入

Each test contains multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of test cases follows.

每个测试包含多个测试用例。第一行包含一个整数 t （$1 \le t \le 10^4$）— 测试用例的数量。测试用例的描述如下。

The first line of each test case contains a single integer n ($2 \le n \le 10^5$) — the length of the array a.

每个测试用例的第一行包含一个整数 n （$2 \le n \le 10^5$）— 数组的长度 a 。

The second line of each test case contains $n - 1$ integers $b_1, b_2, \ldots, b_{n-1}$ ($0 \le b_i < 2^{30}$) — the elements of the array b.

每个测试用例的第二行包含 $n - 1$ 整数 $b_1, b_2, \ldots, b_{n-1}$ （$0 \le b_i < 2^{30}$）— 数组的元素 b 。

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

保证总和为 n 所有测试用例不超过 10^5 。

Output 输出

For each test case, output a single integer -1 if no good arrays exist.

对于每个测试用例，输出一个整数 -1 如果不存在好的数组。

Otherwise, output n space-separated integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$) — the elements of a good array a.

否则，输出 n 空格分隔的整数 $a_1, a_2, \ldots, a_n$ （$0 \le a_i < 2^{30}$）— 一个好的数组的元素 a 。

If there are multiple solutions, you may output any of them.

如果有多个解决方案，您可以输出其中任何一个。

Example 例子

input 输入 Copy 复制

```
4
2
1
3
2 0
4
1 2 3
5
3 5 4 2
```

output 输出 Copy 复制

```
5 3
3 2 1
-1
3 7 5 6 3
```

Let's construct an array a using the formula $a_i = b_{i-1}|b_i$ (assuming b_0 and b_n are 0). We then verify the condition $b_i = a_i \& a_{i+1}$ for $1 \le i < n$. If any condition fails, then such an array a does not exist.

我们来构建一个数组 a 使用公式 $a_i = b_{i-1}|b_i$ （假设 b_0 和 b_n 是 0 ）。然后我们验证条件 $b_i = a_i \& a_{i+1}$ 为了 $1 \le i < n$。如果任何条件失败，则这样的数组 a 不存在。

Explanation 解释

Using the above construction method, we get: $a_i \& a_{i+1} = (b_{i-1}|b_i) \& (b_i|b_{i+1}) = b_i|(b_{i-1} \& b_{i+1})$

使用上面的构造方法，我们得到： $a_i \& a_{i+1} = (b_{i-1}|b_i) \& (b_i|b_{i+1}) = b_i|(b_{i-1} \& b_{i+1})$

If for any i, $b_i|(b_{i-1} \& b_{i+1}) \neq b_i$, then the constructed array a does not satisfy the required condition. This indicates that there exists a bit in b where $b_{i-1} = 1$, $b_i = 0$, and $b_{i+1} = 1$, which can prove there is no solution. **Proof:** If $a_{i-2} \& a_{i-1} = b_{i-1} = 1$, then $a_{i-1} = 1$. Similarly, since $a_i \& a_{i+1} = b_i = 1$, then $a_i = 1$. However, $a_{i-1} \& a_i = 1 \& 1 = 1$, which contradicts $b_i = 0$. Therefore, this configuration is unsolvable.

如果对于任何 i，$b_i|(b_{i-1} \& b_{i+1}) \neq b_i$，然后构建的数组 a 不满足所需条件。这表明存在一个位 b 在哪里 $b_{i-1} = 1$，$b_i = 0$， 和 $b_{i+1} = 1$，可以证明无解。**证明：** 如果 $a_{i-2} \& a_{i-1} = b_{i-1} = 1$， 然后 $a_{i-1} = 1$。同样，由于 $a_i \& a_{i+1} = b_i = 1$， 然后 $a_i = 1$。然而， $a_{i-1} \& a_i = 1 \& 1 = 1$，这与 $b_i = 0$。因此，这种配置是无解的。

This shows that our construction method can construct a valid array a, except in cases where the above bit configuration in b makes it unsolvable.

这说明我们的构造方法可以构造出一个有效的数组 a， 除非上述位配置在 b 使其无解。

结论：ai = bi|bi-1 当且仅当a[i] & a[i + 1]) == b[i] (定义a0 、 an = 0)

KMP

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include<cstring>
#define inf 0x3f3f3f3f3f
#define MAXN 1000010
using namespace std;
int kmp[MAXN];
int lena, lenb, j;
char a[MAXN], b[MAXN];
//KMP from index 1
int main()
{
    cin >> a + 1;
    cin >> b + 1;
    lena = strlen(a + 1);
    lenb = strlen(b + 1);
    for (int i = 2;i <= lenb;i++)
    {
        while (j && b[i] != b[j + 1])
            j = kmp[j];
        if (b[j + 1] == b[i])j++;
        kmp[i] = j;
    }
    j = 0;
    for (int i = 1;i <= lena;i++)
    {
        while (j > 0 && b[j + 1] != a[i])
            j = kmp[j];
```

```
        if (b[j + 1] == a[i])
            j++;
        if (j == lenb) {
            cout << i - lenb + 1 << endl;
            j = kmp[j];
        }
    }

    for (int i = 1;i <= lenb;i++) {
        cout << kmp[i] << " ";
    }
    return 0;
}
```

字典树模板

```
#include <cstdio>
#include <algorithm>
#include <cmath>
#include <cstring>
#include <iostream>
using namespace std;
char s[10010];
int nex[500010][26],n,cnt=0;
bool b[500010][1010];

inline void insert(int x)
{
    scanf("%s",s+1);
    int l=strlen(s+1);
    int now=0;
    for(int i=1;i<=l;i++)
    {
        int p=s[i]-'a';
        if(!nex[now][p])          //如果$Trie$树中没有这个单词的前缀就进行编号
            nex[now][p]=++cnt;    //上文中说到的编号
        now=nex[now][p];          //接着深入一层，更新现在的位置
    }
    b[now][x]=1;                  //这个单词在第x行出现了
}
inline void check()
{
    scanf("%s",s+1);
    int l=strlen(s+1);
    int now=0,flag=1;
    for(int i=1;i<=l;i++)
    {
        int p=s[i]-'a';
        if(!nex[now][p])          //如果在Trie树中没有当前的字符，就可以直接break掉了
        {
            flag=0;
            break;
        }
```

```
        now=nex[now][p];          //否则就更新位置
    }
    if(flag)
        for(int i=1;i<=n;i++)     //题面上说按字典序输出
            if(b[now][i])
                printf("%d ",i); //输出在哪些句子中出现过
    puts("");                     //相当于printf("\n");其实这个条件很容易看不到，一定要注
意啊！！
}
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>x;
        for(int j=1;j<=x;j++)     //一个单词一个单词的插入Trie树里
            insert(i);
    }
    cin>>m;
    for(int i=1;i<=m;i++)
        check();
    return 0;
}
```

ST表

解决区间内最值查询

洛谷p2251（也可数组构造单调队列）

```
#include<iostream>
#include<vector>
#include<cstring>
#include<algorithm>
#include<math.h>
#include<queue>
#define MAXN 1000005
#define inf 0x3f3f3f3f3f
using namespace std;
typedef long long ll;

int n, m;
int q1[100001]; //存a组元素的下标
int a[100001];
//数组构造单调队列
void min_deque() {
    int h = 1, t = 0;
    for (int i = 1; i <= n; i++) {
        while (h <= t && q1[h] + m <= i) h++; //如果队首元素已经不在区间内，弹出
        while (h <= t && a[i] < a[q1[t]]) t--; //如果队尾元素大于新元素，弹出
        q1[++t] = i;//新元素入队
        if (i >= m) printf("%d\n", a[q1[h]]);//输出当前区间的最小值
    }
```

```
}

int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    min_deque();
    return 0;
}
```

线段树模板

洛谷p3374 p3368

```
//p3374
    #include <iostream>
    #include <algorithm>
    #include <cstdio>
    #include <cstring>
    #include <cmath>
    #include <queue>
    using namespace std;
    int n,m;
    int ans;
    int he=0;
    int input[500010];
    struct node
    {
        int left,right;
        int num;
    }tree[2000010];
    void build(int left,int right,int index)
    {
        he++;
        tree[index].left=left;
        tree[index].right=right;
           if(left==right)
            return ;
        int mid=(right+left)/2;
        build(left,mid,index*2);
        build(mid+1,right,index*2+1);
    }
    int add(int index)
    {
        if(tree[index].left==tree[index].right)
        {
            //cout<<index<<" "<<input[tree[index].right]<<endl;
            tree[index].num=input[tree[index].right];
            return tree[index].num;
        }
        tree[index].num=add(index*2)+add(index*2+1);
        return tree[index].num;
```

```
    }
    void my_plus(int index,int dis,int k)
    {
        tree[index].num+=k;
        if(tree[index].left==tree[index].right)
            return ;
        if(dis<=tree[index*2].right)
            my_plus(index*2,dis,k);
        if(dis>=tree[index*2+1].left)
            my_plus(index*2+1,dis,k);
    }
    void search(int index,int l,int r)
    {
        //cout<<index<<" ";
        if(tree[index].left>=l && tree[index].right<=r)
        {
            ans+=tree[index].num;
            return ;
        }
        if(tree[index*2].right>=l)
            search(index*2,l,r);
        if(tree[index*2+1].left<=r)
            search(index*2+1,l,r);
    }
    int main()
    {
        cin>>n>>m;
        for(int i=1;i<=n;i++)
            scanf("%d",&input[i]);
        build(1,n,1);
        add(1);
        for(int i=1;i<=m;i++)
        {
            int a,b,c;
            scanf("%d%d%d",&a,&b,&c);
            if(a==1)
            {
                my_plus(1,b,c);
            }
            if(a==2)
            {
                ans=0;
                search(1,b,c);
                printf("%d\n",ans);
            }
        }
    }
```

```
//p3368
    #include <iostream>
    #include <algorithm>
    #include <cstdio>
    #include <cstring>
```

```cpp
#include <cmath>
#include <queue>
using namespace std;
int n,m;
int ans;
int input[500010];
struct node
{
    int left,right;
    int num;
}tree[2000010];
void build(int left,int right,int index)
{
    tree[index].num=0;
    tree[index].left=left;
    tree[index].right=right;
       if(left==right)
        return ;
    int mid=(right+left)/2;
    build(left,mid,index*2);
    build(mid+1,right,index*2+1);
}
/*int add(int index)
{
    if(tree[index].left==tree[index].right)
    {
        tree[index].num=input[tree[index].right];
        return tree[index].num;
    }
    tree[index].num=add(index*2)+add(index*2+1);
    return tree[index].num;
}
*/
void pls(int index,int l,int r,int k)
{
    if(tree[index].left>=l && tree[index].right<=r)
    {
        tree[index].num+=k;
        return ;
    }
    if(tree[index*2].right>=l)
       pls(index*2,l,r,k);
    if(tree[index*2+1].left<=r)
       pls(index*2+1,l,r,k);
}
void search(int index,int dis)
{
    ans+=tree[index].num;
    if(tree[index].left==tree[index].right)
        return ;
    if(dis<=tree[index*2].right)
        search(index*2,dis);
    if(dis>=tree[index*2+1].left)
        search(index*2+1,dis);
}
int main()
```

```
    {
        int n,m;
        cin>>n>>m;
        build(1,n,1);
        for(int i=1;i<=n;i++)
            scanf("%d",&input[i]);
        for(int i=1;i<=m;i++)
        {
            int a;
            scanf("%d",&a);
            if(a==1)
            {
                int x,y,z;
                scanf("%d%d%d",&x,&y,&z);
                pls(1,x,y,z);
            }
            if(a==2)
            {
                ans=0;
                int x;
                scanf("%d",&x);
                search(1,x);
                printf("%d\n",ans+input[x]);
            }
        }
    }
```

拓扑排序模板

洛谷p4017

```
#include<iostream>
#include<vector>
#include<queue>
#include<string>
using namespace std;
#define mod 80112002
typedef long long ll;
const int N = 5e3 + 2;
queue<ll>q;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    ll n, m;
    cin >> n >> m;
    vector<ll>in(n + 1, 0), out(n + 1, 0),  num(n + 1, 0);
    vector<vector<ll>>a(N);
    //num为记录到这个点的类食物连的数量
 // 拓扑排序模板
    ll ans = 0;
    for (int i = 1;i <= m;i++) {
```

```
        int x, y;
        cin >> x >> y;
        in[y]++;
        out[x]++;
        a[x].push_back(y); //单向边
    }
    for (int i = 1; i <= n; ++i) {
        if (!in[i]) {
            num[i] = 1;
            q.push(i);
        }
    }
    while (!q.empty()) {
        ll top = q.front();
        q.pop();
        ll len = a[top].size();
        for (int i = 0;i < len;i++) {
            ll next = a[top][i];
            in[next]--;
            num[next] = (num[next] + num[top]) % mod;//更新到下一个点的路径数量
            if (in[next] == 0)q.push(a[top][i]);//如果这个点的入度为0了,那么压入队列
        }
    }
    for (int i = 1; i <= n; ++i) {
        if (!out[i])
            ans = (ans + num[i]) % mod;
        }
    cout << ans << endl;
    return 0;
}
```

邻接矩阵/邻接表/链式前向星 存储图

[图的两种存储形式（邻接矩阵、邻接表） 邻接矩阵的特点-CSDN博客]

[链式前向星--最通俗易懂的讲解-CSDN博客]

邻接矩阵好写效率低（浪费一半空间） 邻接表难写效率高 链式前向星综合起来都较为一般

SPFA最短路问题

可处理有负权边的情况 [SPFA算法（最短路径算法）-CSDN博客]

洛谷p3371

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<vector>
#include<algorithm>
#include<math.h>
```

```cpp
#include<queue>
#define inf 0x3f3f3f3f3f
using namespace std;
typedef long long ll;
const int maxn = 10001;
const long long INF = 2147483647;
int dis[maxn];//记录最小路径的数组
int vis[maxn];//标记
int n, m, s;
struct node {
    int s1;//记录结点
    int side;//边权
};
void init() {
    for (int i = 1; i <= n; i++) {
        dis[i] = INF;
        vis[i] = 0;
    }
}
vector<node>mp[maxn];//用vector建立邻接表
void Spfa(int s) {
    queue<int>v;
    vis[s] = 1; v.push(s); dis[s] = 0;
    while (!v.empty()) {
        int q = v.front();
        v.pop(); vis[q] = 0;
        for (int i = 0; i < mp[q].size(); i++) {
            if (dis[mp[q][i].s1] > dis[q] + mp[q][i].side) {
                dis[mp[q][i].s1] = dis[q] + mp[q][i].side;//更新最短路径。
                if (vis[mp[q][i].s1]) {
                    continue;//如果已经标记，则继续下一次循环
                }
                v.push(mp[q][i].s1);
            }

        }
    }
}
int main()
{
    int x, y, r;

    cin >> n >> m >> s;
    init();
    while (m--) {
        node h;
        cin >> x >> y >> r;
        h.s1 = y;//因为该图为有向图，记录指向的结点
        h.side = r;//记录路径
        mp[x].push_back(h);
    }
    Spfa(s);
    for (int i = 1; i <= n; i++) {
            cout << dis[i] << " ";
    }
    return 0;
```

```
}
```

dijkastra最短路

不能处理有负权边的情况

重要：洛谷p4779 p3371 //可以用堆优化时间复杂度

```
#include<iostream>
#include<cstring>
#include<vector>
#include<algorithm>
#include<math.h>
#include<queue>
#define inf 0x3f3f3f3f3f
using namespace std;
typedef long long ll;

int head[100000], cnt;
long long ans[1000000];
bool vis[1000000];
int m, n, s;
struct edge
{
    int to;
    int nextt;
    int weight;
}edge[1000000];
void addedge(int x, int y, int z)
{
    edge[++cnt].to = y;
    edge[cnt].weight = z;
    edge[cnt].nextt = head[x];
    head[x] = cnt;
}
int main()
{
    cin >> m >> n >> s;
    for (int i = 1;i <= n;i++)
    {
        ans[i] = 2147483647;
    }
    ans[s] = 0;
    for (int i = 1;i <= n;i++)
    {
        int a, b, c;
        cin >> a >> b >> c;
        addedge(a, b, c);
    }
    int now_pos = s;
    while (vis[now_pos] == 0)
    {
```

```
        ll minn = 2147483647;
        vis[now_pos] = 1;
        for (int i = head[now_pos];i != 0;i = edge[i].nextt)
        {
            if (!vis[edge[i].to] && ans[edge[i].to] > ans[now_pos] +
edge[i].weight)
            {
                ans[edge[i].to] = ans[now_pos] + edge[i].weight;  //更换到小值
            }
        }
        for (int i = 1;i <= m;i++)
        {
            if (ans[i] < minn && vis[i] == 0)  //记录走过地方的小值及换点搜索
            {
                minn = ans[i];
                now_pos = i;
            }
        }
    }
    for (int i = 1;i <= m;i++)
    {
        cout << ans[i] << ' ';
    }
    return 0;
}
```

优先队列优化:

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<vector>
#include<algorithm>
#include<math.h>
#include<queue>
#define inf 0x3f3f3f3f3f
using namespace std;
typedef long long ll;

int head[100000], cnt;
long long ans[1000000];
bool vis[1000000];
int m, n, s;
struct edge
{
    int to;
    int nextt;
    int wei;
}edge[1000000];
struct priority
{
    int ans;
    int id;
    bool operator <(const priority& x)const
    {
```

```
        return x.ans < ans;
    }
};
void addedge(int x, int y, int z)
{
    edge[++cnt].to = y;
    edge[cnt].wei = z;
    edge[cnt].nextt = head[x];
    head[x] = cnt;
}

priority_queue<priority> q;

int main()
{
    cin >> m >> n >> s;
    for (int i = 1;i <= n;i++)
    {
        ans[i] = 2147483647;
    }
    ans[s] = 0;
    for (int i = 1;i <= n;i++)
    {
        int a, b, c;
        cin >> a >> b >> c;
        addedge(a, b, c);
    }
    int u;
    q.push((priority) { 0, s });
    while (!q.empty())     //利用优先队列优化时间复杂度 不需要每次都遍历全部了
    {
        priority temp = q.top();
        q.pop();
        u = temp.id;
        if (!vis[u])
        {
            vis[u] = 1;
            for (int i = head[u];i;i = edge[i].nextt)
            {
                int v = edge[i].to;
                if (ans[v] > ans[u] + edge[i].wei)
                {
                    ans[v] = ans[u] + edge[i].wei;
                    if (!vis[v])
                    {
                        q.push((priority) { ans[v], v });
                    }
                }
            }
        }
    }
    for (int i = 1;i <= m;i++)
    {
        cout << ans[i] << ' ';
    }
    return 0;
```

```
}
```

图dfs小模板（会了记得删）

wangbozhouniuke

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
vector<int>a[50];
int sum,vis[50],ans;
void dfs(int x)
{
    for(int i=0;i<a[x].size();i++)
    {
        if(!vis[a[x][i]]&&a[x][i]!=x)
        {
            sum++;
            ans=max(ans,sum);
            int temp[50]={0};
            for(int i=0;i<50;i++)
            {
                temp[i]=vis[i];
            }
            for(int j=0;j<a[x].size();j++)
            {
                vis[a[x][j]]=1;
            }
            dfs(a[x][i]);
            sum--;
            for(int i=0;i<50;i++)
            {
                vis[i]=temp[i];
            }
        }
    }
}
int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=m;i++)
    {
        int u,v;
        cin>>u>>v;
        a[u].push_back(v);
        a[v].push_back(u);
    }
```

```
    for(int i=1;i<=n;i++)
    {
        memset(vis,0,sizeof(vis));
        sum=0;
        vis[i]=1;
        dfs(i);
    }
    cout<<ans+1<<endl;
    return 0;
}
```

洛谷企鹅与猫猫

```
#include<iostream>
#include<cstring>
#include<vector>
#include<algorithm>
#include<math.h>
#include<queue>
#define inf 0x3f3f3f3f3f
using namespace std;
typedef long long ll;

inline 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;
}
ll n, d;
vector<vector<ll>>a(n + 1, vector<ll>(n + 1, 0));
vector<bool>vis(n + 1);
ll ans = 0;

void dfs(ll now, ll dis) {
    vis[now] = 1;
    if (dis == d)return;
    for (int i = 0;i < a[now].size();i++) {
        if (!vis[a[now][i]]) {
            dfs(a[now][i], dis + 1);
            ans++;
        }
    }
}

int main()
{
```

```
    cin >> n >> d;
    a.resize(n + 1);
    vis.resize(n + 1);
    for (int i = 1;i <= n - 1;i++) {
        ll u, v;
        cin >> u >> v;
        a[u].push_back(v);
        a[v].push_back(u);
    }
    dfs(1,0);
    cout << ans << endl;
    return 0;
}
```

快速幂模板

```
long long int quik_power(int base, int power)
{
    long long int result = 1;    //用于存储项累乘与返回最终结果，由于要存储累乘所以要初始化为1
    while (power > 0)            //指数大于0说明指数的二进制位并没有被左移舍弃完毕
    {
        if (power & 1)           //指数的当前计算二进制位也就是最末尾的位是非零位也就是1的时候
                                 //例如1001的当前计算位就是1， 100*1* 星号中的1就是当前计算使用的位
            result *= base;      //累乘当前项并存储
        base *= base;            //计算下一个项，例如当前是n^2的话计算下一项n^2的值
                                 //n^4 = n^2 * n^2;
        power >>= 1;             //指数位右移，为下一次运算做准备
                                 //一次的右移将舍弃一个位例如1011(2)一次左移后变成101(2)
    }
    return result;               //返回最终结果
}
```