

数据类型

数据类型	范围
int	$[-2^{31}, 2^{31}]$
long long	$[-2^{63}, 2^{63} - 1]$

单位

单位	说明
Bit	二进制位
Byte	字节 1 Byte = 8 Bits

操作

1. 数据类型
2. 数据类型转换
 - o 强制转换
 - `(数据类型)a`
 - `(数据类型)a` 强制转换为 `(数据类型)b`
 - `(数据类型)a` 强制转换为 `(数据类型)b`
 - o 隐式转换
 - `(数据类型)r` 强制转换为 `(数据类型)`
 - `(数据类型)` 强制转换为 `(数据类型)`
 - `r` 强制转换为 `(数据类型)`
 - `(数据类型)` 强制转换为 `1`
 - `(数据类型)r`
3. 数据类型

```
##include<iostream>
using namespace std;
const int N = 1e6 + 10;

int n;
int q[N];

void quick_sort(int q[],int l,int r){
    if(l >= r) return;
    int x = q[(int)((l + r) / 2)];
    int i = l - 1 ,j = r + 1;
    while(i < j){
        do i++; while(q[i] < x);
        do j--; while(q[j] > x);
        if(i < j) swap(q[i],q[j]);
    }
    quick_sort(q,l,j);
    quick_sort(q,j + 1,r);
}
```

```

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

    quick_sort(q,0,n - 1);

    for(int i = 0;i < n;i++) printf("%d ",q[i]);

    return 0;
}

```

□□□□

- □□□□□□ $O(n\log n)$
- 1. □□□□□□ $mid = \frac{l + r}{2}$
- 2. □□□□□□□□□□□□
- 3. □□□□□□□□□□□□
 - □□□□□□□□□□□□□□□□□□
 - □□□□□□□□□□□□□□□□
 - □□□□□□□□□□□□□□

```

#include<iostream>
using namespace std;

const int N = 1e6 + 10;
int n;
int q[N],temp[N];

void merge_sort(int q[],int l,int r){
    if(l >= r) return;
    int mid = (l + r) / 2;
    merge_sort(q, l, mid);
    merge_sort(q, mid + 1, r);

    int k = 0,i = l,j = mid + 1;
    while(i <= mid && j <= r){
        if(q[i] <= q[j]) temp[k++] = q[i++];
        if(q[i] > q[j]) temp[k++] = q[j++];
    }
    while(i <= mid) temp[k++] = q[i++];
    while(j <= r) temp[k++] = q[j++];

    for(int i = l, j = 0;i <= r;i++,j++) q[i] = temp[j];
}

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

    for(int i = 0;i < n;i++) scanf("%d",&q[i]);
}

```

```

merge_sort(q,0,n - 1);

for(int i = 0;i < n;i++) printf("%d ",q[i]);

return 0;
}

```

□□□□

```

// □□□□□□□□□□
bool check(int x){...}

// □□□□□□□□□□
int binary_search1(int l,int r){
    while(l < r){
        int mid = (l + r) / 2;
        if(check(mid)) r = mid;
        else l = mid + 1;
    }
    return l;
}

// □□□□□□□□□□
int binary_search2(int l, int r){
    while(l < r){
        int mid = (l + r + 1) / 2; // □□□□□□□□
        if(check(mid)) l = mid;
        else r = mid - 1;
    }
    return l;
}

```

□□□□□

```

bool check(double x){}

// □□□□□□□
double binary_search(double l, double r){
    while(r - l > eps){ // eps□□□□□□□□□□□□□□
        double mid = (l + r) / 2;
        if(check(mid)) r = mid;
        else l = mid;
    }
    return l;
}

```

□□□

- ☐☐☐

- $\square\square\square$
 - $\$A > B \implies \$A - B$
 - $\$A < B \implies -(B - A)$

```
##include<iostream>
##include<vector>

const int N = 1e6 + 10;

int n;

using namespace std;

// 00A0000000B

bool cmp(vector<int> &a,vector<int> &b){
    if(a.size() != b.size()) return a.size() > b.size();
    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;
        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');
    if(cmp(A,B)){
        vector<int> c = sub(A,B);
        for(int i = c.size() - 1;i >= 0 ;i--) printf("%d",c[i]);
    }
    else{
        vector<int> c = sub(B,A);
        printf("-");
        for(int i = c.size() - 1;i >= 0 ;i--) printf("%d",c[i]);
    }
}

```

- □□□

```

vector<int> multi(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 = t / 10;
    }
    while(t > 0){
        c.push_back(t % 10);
        t /= 10;
    }
    // if(t > 0) c.push_back(t);
    while(c.size() > 1 && c.back() == 0) c.pop_back();
    return c;
}

```


11

- ```

00$a_1,a_2,\dots,a_n$$$b_1,b_2,\dots,b_n$$$a_i = b_1 + b_2 + \dots + b_i$$$b$$$a$$$a$$$a$$$
$b$$$ $ $ b_1 = a_1 \setminus b_2 = a_2 - a_1 \setminus b_3 = a_3 - a_2 \setminus \dots \setminus b_n = a_n - a_{\{n-1\}} $ $ $
$[l,r]$$$a_i$$$C$$$$O(N)$$$$a$$$b$$$b_l+C$$$a_l$$$$
$C$$$$b_{\{r+1\}} - C$$$$[l,r]$$$a$$$C$$$$a[i]$$$

```

- \$a\_{ij}\$

```

#include<iostream>
using namespace std;

int n,m,q;

const int N = 1010;
int a[N][N],b[N][N];

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

int main(){
 cin >> n >> m >> q;
 for(int i = 1;i <= n;i++){
 for(int j = 1;j <= m;j++){
 int t;
 cin >> t;
 insert(i,j,i,j,t);
 }
 }
 while(q--){

```

```
int x1,y1,x2,y2,c;
cin >> x1 >> y1 >> x2 >> y2 >> c;
insert(x1,y1,x2,y2,c);
}

for(int i = 1;i <= n;i++){
 for(int j = 1;j <= m;j++){
 b[i][j] = b[i][j] + b[i - 1][j] + b[i][j - 1] - b[i - 1][j - 1]; // 递推
 }
}

for(int i = 1;i <= n;i++){
 for(int j = 1;j <= m;j++){
 cout << b[i][j] << " ";
 }
 cout << endl;
}
}
```

□□□□□

$O(n^2)$   $O(n)$

□□□□□□□□

111

n k

0000

1. `xxxkxxxxxxxxn >> k`
  2. `xxxxxx n & 1`
  3. `n >> k & 1;`
- `$lowbit(x)$xxx×xxxxxxxxxx1xxxxxx`
    - `x = 1010xxx10`
    - `x = 101000xxx1000`
    - `x & -x`

111

□□□□□□□□□□□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

- 
- 
- 

□□□□

- `alls`
- `add_query`



- 二分查找

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

const int N = 3e5 + 10; // 数组大小
int a[N], s[N]; // 数组

vector<int> alls; // 所有元素

typedef pair<int, int> PII;
vector<PII> add, query;
int n, m;
// 输入
int find(int x){
 int l = 0, r = alls.size() - 1;
 while(l < r){
 int mid = l + r >> 1;
 if(alls[mid] >= x) r = mid;
 else l = mid + 1;
 }
 return r + 1;
}

int main(){
 cin >> n >> m;
 // 输入
 for(int i = 0; i < n; i++){
 int x, c;
 cin >> x >> c;
 alls.push_back(x);
 add.push_back({x, c});
 }

 // 输入
 while(m--){
 int l, r;
 cin >> l >> r;
 query.push_back({l, r});
 alls.push_back(l);
 alls.push_back(r);
 }

 // 排序
 sort(alls.begin(), alls.end());
 alls.erase(unique(alls.begin(), alls.end()), alls.end());

 // 输出
```

```

for(auto x: add){
 int t = find(x.first);
 a[t] += x.second;
}

// 查询
for(int i = 1;i <= alls.size();i++){
 s[i] = a[i] + s[i - 1];
}

// 查询
for(auto x: query){
 int l = find(x.first);
 int r = find(x.second);
 cout << s[r] - s[l - 1] << endl;
}
}

```

查询

查询查询查询查询查询

1. 查询查询
2. 查询查询查询查询

```

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

vector<pair<int,int>> t;

int n;

void merge(vector<pair<int,int>> &s){
 vector<pair<int,int>> temp;
 sort(s.begin(),s.end());
 // 初始化
 int l = s[0].first,r = s[0].second;
 for(int i = 1;i < s.size();i++){
 if(s[i].first > r) {
 temp.push_back({l,r});
 l = s[i].first;
 r = s[i].second;
 }
 else r = max(r,s[i].second);// 初始化
 }
 // 初始化
 temp.push_back({l,r});
}

```

```

 t = temp;
 }

 int main(){
 cin >> n;
 while(n--){
 int a,b;
 cin >> a >> b;
 t.push_back({a,b});
 }
 merge(t);
 cout << t.size();
 }

```

□□□□

□□□□□□□

□□

1. □□□□□□□□□□□□□□
  - e[N]□□□□□□□□□□
  - ne[N]□□□□□□□□next□□
2. □□□□□□□□□□□□□□
  - l[N]□□□□□□□□□□
  - r[N]□□□□□□□□□□
3. □□□□n□□□□□

□□□

```

#include<iostream>

using namespace std;

const int N = 1000010;
/*
e[N]□□□□□□□□□□
ne[N]□□□□□□□□□□
head□□□□□□□□
idx□□□□□□□□□□□□
*/
int e[N],ne[N];
int head,idx;

void init(){
 head = -1;
 idx = 0;
}

```

```

void add_to_head(int c){
 e[idx] = c;
 ne[idx] = head;
 head = idx++;
}

void erase(int k){
 // 删除k
 if(k == -1) {
 head = ne[head];
 }
 ne[k] = ne[ne[k]];
}

void add(int x,int c){
 e[idx] = c;
 ne[idx] = ne[x];
 ne[x] = idx++;
}

int main(){
 init();
 int m;
 cin >> m;
 while(m--){
 int k,x;
 char op;
 cin >> op;
 if(op == 'H'){
 cin >> x;
 add_to_head(x);
 }
 else if(op == 'D'){
 cin >> k ;
 erase(k - 1);
 }
 else if(op == 'I'){
 cin >> k >> x;
 add(k - 1,x);
 }
 }
 for(int i = head;i != -1;i = ne[i]){
 cout << e[i] << " ";
 }
}

```

[illegible]

```

 cin >> k >> x;
 add(k + 1,x);
 }
}
for(int i = r[0];i != 1;i = r[i]){
 cout << e[i] << " ";
}
}

```

□

□□□□

```

#include<iostream>
using namespace std;

const int N = 100010;
int s[N];
int top;

void init(){
 top = -1;
}

bool isempty(){
 return top == -1;
}

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

void pop(){
 top--;
}

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

int main(){
 init();
 int m;
 cin >> m;
 while(m--){
 int x;
 string op;
 cin >> op;
 if(op == "push"){
 cin >> x;
 push(x);

```

```

 }
 else if(op == "pop") pop();
 else if(op == "empty"){
 if(isempty()) cout << "YES" << endl;
 else cout << "NO" << endl;
 }
 else if(op == "query") cout << query()<< endl;
}
}

```

□□

□□□□

```

#include<iostream>
using namespace std;

const int N = 100010;
int s[N],top,tail;

void init(){
 top = 0;
 tail = -1;
}

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

bool isempty(){
 return tail < top;
}

void pop(){
 top++;
}

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

int main(){
 int m;
 init();
 cin >> m;
 while(m--){
 int x;
 string op;
 cin >> op;
 }
}

```

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

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



```

int a[N],st[N];
int head = 0,tail = -1;
int main(){

 int n,k;
 cin >> n >> k;
 for(int i = 0;i < n;i++){
 cin >> a[i];
 }
 // 输出
 for(int i = 0;i < n;i++){
 if(i - k + 1 > st[head]) head ++;
 while(tail >= head && a[st[tail]] >= a[i]) tail--;
 st[++tail] = i;
 if(i >= k - 1) cout << a[st[head]] << " ";
 }
 cout << endl;
 // 输入
 head = 0,tail = -1;
 for(int i = 0;i < n;i++){
 if(i - k + 1 > st[head]) head ++;
 while(tail >= head && a[st[tail]] <= a[i]) tail--;
 st[++tail] = i;
 if(i >= k - 1) cout << a[st[head]] << " ";
 }
 cout << endl;
}

```

完整代码

```

#include<iostream>

using namespace std;
const int N = 1e6 + 10;
int n,k;
int tail = -1,head = 0;
int q[N],a[N];

int main(){
 cin >> n >> k;
 for(int i = 1;i <= n;i++){
 cin >> a[i];
 }
 for(int i = 1;i <= n;i++){
 if(head <= tail && i > k && q[head] == a[i - k]) head ++;
 while(tail >= head && q[tail] > a[i]) tail --;
 q[++tail] = a[i];
 if(i > k - 1){
 cout << q[head] << " ";
 }
 }
}

```

```

cout << endl;
head = 0, tail = -1;
for(int i = 1; i <= n; i++){
 if(head <= tail && i > k && q[head] == a[i - k]) head ++;
 while(tail >= head && q[tail] < a[i]) tail --;
 q[++tail] = a[i];
 if(i > k - 1){
 cout << q[head] << " ";
 }
}
}

```

## KMP

- 字符串匹配问题
- 字符串next数组的求法
- 字符串匹配问题 n + 1 字符串匹配问题 n - next[n] 字符串 n+1 字符串

```

#include<iostream>
using namespace std;

const int N = 1e6 + 10;
char s[N], p[N];
// 字符串匹配问题0字符串匹配问题ne[1]0
int ne[N];
int main(){
 int n, m;
 cin >> n >> p + 1 >> m >> s + 1;
 /*
 j字符串匹配问题
 a b a b a
 1 2 3 4 5
 ne 0 0 1 2 3
 a b a b a
 a b a b a
 */
 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) {
 // 字符串匹配问题0字符串匹配问题i-n+1
 cout << i - n << " ";
 j = ne[j];
 }
 }
}

```

```

 }
}

}

```

## Trie

trie는 문자열을 저장하는 자료구조

trie의 구조

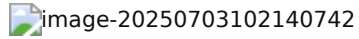


image-20250703102140742

trie는 문자열을 저장하는 자료구조

trie의 구조

```

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

const int N = 1e5 + 10;
int son[N][26],idx,cnt[N];
int n;

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

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

int main(){
 ios::sync_with_stdio(false);
 cout.tie(0),cin.tie(0);
 cin >> n;
 while(n--){
 string s;
 char x[N];
 }
}

```



000

- 6/6/20

- 000000

- 1111

```

#include<iostream>

using namespace std;

const int N = 1e5 + 10;

int n,m;

int h[N];

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

int main(){
 ios::sync_with_stdio(false);
 cin.tie(0),cout.tie(0);
 cin >> n >> m;
 for(int i = 0;i < n;i++) h[i] = i; // □□□□□□□□□□
 int a,b;
 while(m--){
 char c;
 cin >> c;
 if(c == 'M'){

```

115

STL

1. □□□□□
2. □□□□□□
3. □□□□□
4. □□□□□□□□□□

- $x^2 + 2x$
- $x^2 + 2x + 1$

- $n/2$
- **up**
- **down**

□□□□

- ```
// 空白行
#include<iostream>
```

```

#include<cstring>
using namespace std;

const int N = 2e5 + 3; // 00000000000000000000+30000000000000000000
const int NuLL = 0x3f3f3f3f; //0000
int h[N];
int n;

// 00000000x0000000000000000x00000000
int find(int x){
    int k = (x % N + N) % N;
    while(h[k] != NuLL && h[k] != x){
        k++;
        if(k == N) k = 0;
    }
    return k;
}

int main(){
    memset(h,0x3f,sizeof(h)); // 0000
    string s;
    cin >> n;
    while(n--){
        cin >> s;
        int x;
        if(s == "I"){
            cin >> x;
            h[find(x)] = x;
        }
        else if(s == "Q"){
            cin >> x;
            int k = find(x);
            if(h[k] == x) cout << "Yes" << endl;
            else cout << "No" << endl;
        }
    }
}

```

2. 0000

- 00000000000000000000
- 00000000000000000000
- 00000000000000000000

```

// 0000
#include<iostream>
#include<cstring>
using namespace std;

```

```

const int N = 1e5 + 3; // 100000+3
int h[N];
int e[N], ne[N];
int idx = 0;
int n ;

void insert(int x){
    int k = (x % N + N) % N; // 0~N-1
    e[idx] = x;
    ne[idx] = h[k];
    h[k] = idx++;
}

bool find(int x){
    int k = (x % N + N) % N;
    for(int i = h[k]; i != -1; i = ne[i]){
        if(e[i] == x) return true;
    }
    return false;
}

int main(){
    memset(h, -1, sizeof(h)); // 0~N-1
    cin >> n;
    string s;
    int x;
    while(n--){
        cin >> s;
        if(s == "I"){
            cin >> x;
            insert(x);
        }
        else if(s == "Q"){
            cin >> x;
            if(find(x)) cout << "Yes" << endl;
            else cout << "No" << endl;
        }
    }
}

```

00000

0000000000000000

- 000000000000000000000000p0000000000000000000000q000000000000
 - p0001310013331
 - q002^{64}\$00000000unsigned long long00000000000000000000
- 000000000000000000000000h00000

- `h[0] = 0`
- `h[i] = h[i - 1] * 131 + str[i];`
- 字符串哈希的时间复杂度为 $O(1)$
- $h[L-R] = h[R] - h[L-1] * 131^{R-L+1}$
- 131^{R-L+1} 预处理
- 验证 `h[R] == h[L-1] * 131^{R-L+1} + h[L-1]`

```

#include<iostream>
using namespace std;

const int N = 1e5 + 10;
char s[N];
unsigned long long P[N],h[N];
int n,m;
int p = 131;

int search(int l,int r){
    return h[r] - h[l - 1] * P[r - l + 1];
}

int main(){
    cin >> n >> m;
    cin >> s;
    P[0] = 1;
    for(int i = 0;i < n;i++){
        P[i + 1] = P[i] * p;
        h[i + 1] = h[i] * p + s[i];
    }

    while(m --){
        int l1,r1,l2,r2;
        cin >> l1 >> r1 >> l2 >> r2;
        if(search(l1,r1) == search(l2,r2)) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
}

```

STL

vector

- 字符串哈希
- 定义 `vector<int> a = {10,3}` 初始化10个元素3个
- `size()` 返回元素个数
- `empty()` 是否空
- `clear()` 清空
- `front()/back()` 返回首/尾元素
- `push_back()/pop_back()` 添加/删除元素
- `begin()/end()` 返回迭代器

pair<int,int>

- 容器成员函数
- first()/second() 第1/2个值
- 初始化 `p = (20, "zxb")` 或 `p = make_pair(20, "zxb")`

string

- substr() 子串
 - substr(1,2) 从第1个位置起取2个
- size() 字符串长度
- empty() 是否空
- clear() 清空

queue

- size() 队列长度
- empty() 是否空
- push() 入队
- front() 队首元素
- back() 队尾元素
- pop() 出队

priority_queue 优先队列

容器适配器

- push() 入队
- front() 队首元素
- pop() 出队

deque 双端队列

- size() 队列长度
- empty() 是否空
- clear() 清空
- front() 队首元素
- back() 队尾元素
- push_back() / pop_back() 入队/出队
- push_front() / pop_front() 入队/出队
- begin()/end() 迭代器/尾后迭代器

stack

- size() 栈长度
- empty() 是否空
- push() 入栈
- top() 栈顶元素
- pop() 出栈

set

常函数

- size() 返回元素个数
- empty() 是否空
- clear() 清空
- insert() 插入元素
- find() 查找元素
- count() 统计元素个数
- erase()
 - 返回被删除元素个数
 - 返回被删除元素迭代器
- lower_bound() 返回元素x在集合中的位置end()
- upper_bound() 返回元素x在集合中的位置end()
- begin()/end() 返回集合的起始/结束迭代器

map

- size() 返回元素个数
- empty() 是否空
- clear() 清空
- insert() 插入元素pair
- erase() 删除元素pair
- m['key'] 返回key对应的value
- lower_bound() 返回元素x在集合中的位置end()
- upper_bound() 返回元素x在集合中的位置end()

multiset

常函数

- size() 返回元素个数
- empty() 是否空
- clear() 清空
- insert() 插入元素
- find() 查找元素
- count() 统计元素个数
- erase()
 - 返回被删除元素个数
 - 返回被删除元素迭代器
- lower_bound() 返回元素x在集合中的位置end()
- upper_bound() 返回元素x在集合中的位置end()

multimap

- size() 返回元素个数
- empty() 是否空
- clear() 清空
- insert() 插入元素pair
- erase() 删除元素pair
- m['key'] 返回key对应的value
- lower_bound() 返回元素x在集合中的位置end()
- upper_bound() 返回元素x在集合中的位置end()

unordered_map **unordered_set** **unordered_multiset** **unordered_multimap**

时间复杂度 **$O(1)$**

lower_bound() upper_bound()

模板

DFS

模板

递归函数+参数+返回值

代码

```
##include<iostream>
using namespace std;
const int N = 10;
int n,t[N];
bool state[N];

void dfs(int p){
    if(p == n){
        for(int i = 0;i < n;i++) cout << t[i] << " ";
        cout << endl;
    }
    for(int i = 1;i <= n;i++){
        if(!state[i]){
            t[p] = i;
            state[i] = true;
            dfs(p + 1);
            state[i] = false;
        }
    }
}

int main(){
    cin >> n;
    dfs(0);
}
```

n

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

const int N = 12,M = 2 * N;
char d[N][N];
int n;
```



```

int n,m;
int bfs(){
    st[0][0] = true;
    p[0][0] = 0;
    q[++tail] = {0,0};
    while(head <= tail){
        auto x = q[head++];
        for(int i = 0;i < 4;i++){
            int a = x.first + dx[i],b = x.second + dy[i]; // a□□□□b□□□
            if(a >= 0 && a < n && b >= 0 && b < m && !st[a][b] && d[a][b] == 0){
                st[a][b] = true;
                p[a][b] = p[x.first][x.second] + 1;
                q[++tail] = {a,b};
            }
        }
    }
    return p[n - 1][m - 1];
}
int main(){
    cin >> n >> m;
    for(int i = 0;i < n;i ++){
        for(int j = 0;j < m;j ++){
            cin >> d[i][j];
        }
    }
    cout << bfs();
}

```

□□□□□□

```

#include<iostream>
#include<queue>
#include<cstring>
using namespace std;
const int N = 1e5 + 10;
int dist[N],e[N],ne[N],h[N],idx;
bool st[N];
int n,m;
queue<int> q;

void add(int a,int b){
    e[idx] = b,ne[idx] = h[a],h[a] = idx ++;
}

int main(){
    cin >> n >> m;
    memset(h,-1,sizeof h);
    memset(dist,0x3f,sizeof(dist));
    while(m--){
        int a,b;
        cin >> a >> b;
    }
}

```

```

        add(a,b);
    }
    q.push(1);
    dist[1] = 0;
    st[1] = true;
    while(q.size()){
        int t = q.front();
        q.pop();
        for(int i = h[t]; i != -1; i = ne[i]){
            int j = e[i];
            if(!st[j]){
                dist[j] = dist[t] + 1;
                q.push(j);
                st[j] = true;
            }
        }
    }
    if(dist[n] == 0x3f3f3f3f) cout << -1;
    else cout << dist[n];
}

```

□□□□

$(u \rightarrow v)$ $\iff u$ \implies v

□□□□□□□

```

#include<iostream>
#include<cstring>

using namespace std;

const int N = 1e5 + 10;

int n,m;
int h[N],e[N],ne[N],idx;
int q[N],d[N];

void add(int a,int b){
    e[idx] = b,ne[idx] = h[a],h[a] = idx++;
}

bool topsort(){
    int head = 0,tail = -1;
    for(int i = 1;i <= n;i++){
        if(!d[i]) q[++tail] = i;
    }
    while(head <= tail){
        int x = q[head ++];
        for(int i = h[x];i != -1;i = ne[i]){
            int j = e[i];
            d[j]--;
            if(!d[j]) q[++tail] = j;
        }
    }
}

```


```

        return tail == n - 1;
    }

    int main(){
        memset(h,-1,sizeof h);
        cin >> n >> m;
        int a,b;
        for(int i = 1;i <= m;i++){
            cin >> a >> b;
            add(a,b);
            d[b] ++;
        }
        if(topsort()){
            for(int i = 0;i < n;i++) cout << q[i] << " ";
        }
        else {
            cout << -1;
        }
    }
}

```

□□□

 image-20250714111902028

- □□Dijkstra□□□□□□□□
- □□□□Dijkstra□□□□□□□□□□□□□□
- □□□□□□□□
- □□□□□□□□□□
- □□□□□□□□SPFA

□□Dijkstra□□

- □□□□□□□□□□□□□□
- □□□□□□□□□□□□

```

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

const int N = 510,M = 1e5 + 10;
int h[N],e[M],w[M],ne[M],idx;
int n,m;
int dist[N];
bool st[N];

void add(int a,int b,int c){
    e[idx] = b,ne[idx] = h[a],w[idx] = c,h[a] = idx ++;
}

```



```

void add(int a,int b,int c){
    e[idx] = b,ne[idx] = h[a],w[idx] = c,h[a] = idx ++;
}

int dijkstra(){
    memset(dist,0x3f,sizeof(dist));
    dist[1] = 0;
    priority_queue <PII,vector<PII>,greater<PII>> heap;
    heap.push({0,1});

    while(heap.size()){
        auto x = heap.top();
        heap.pop();
        if(st[x.second]) continue;
        st[x.second] = true;
        for(int i = h[x.second];i != -1;i = ne[i]){
            int j = e[i];
            if(dist[j] > x.first + w[i]){
                dist[j] = x.first + w[i];
                heap.push({dist[j],j});
            }
        }
    }
    return dist[n];
}

int main(){
    cin >> n >> m;
    memset(h,-1,sizeof h);
    int a,b,c;
    while(m--){
        cin >> a >> b >> c;
        add(a,b,c);
    }
    int res = dijkstra();
    if(res == 0x3f3f3f3f) cout << -1;
    else cout << res;
}

```

Bellman-ford

- 时间复杂度 $O(n^3)$
 - 空间复杂度 $O(n^2)$
- 可以处理负权边

SPFA

- 时间复杂度
- 空间复杂度
- 可以处理负权边，st 数组记录是否入队

```

#include<iostream>
#include<queue>
#include<cstring>
using namespace std;
const int N = 1e5 + 10;
typedef pair<int,int> PII;
int n,m;

int e[N],ne[N],idx,w[N],h[N],dist[N];
bool st[N];
void add(int x,int y,int z){
    e[idx] = y,ne[idx] = h[x],w[idx] = z,h[x] = idx ++;
}

int spfa(){
    memset(dist,0x3f,sizeof dist);
    queue<PII> q;
    dist[1] = 0;
    q.push({0,1}); // 0 1
    st[1] = true;
    while(q.size()){
        auto t = q.front();
        q.pop();
        int dis = t.first,vec = t.second;
        st[vec] = false;
        for(int i = h[vec];i != -1;i = ne[i]){
            int j = e[i];
            if(dist[j] > dist[vec] + w[i]){ // 松弛操作
                dist[j] = dist[vec] + w[i];
                if(!st[j]){
                    q.push({dist[j],j});
                    st[j] = true;
                }
            }
        }
    }
    return dist[n] >= 0x3f3f3f3f / 2; // 是否可达
}

int main(){
    memset(h,-1,sizeof h);
    cin >> n >> m;
    while(m--){
        int x,y,z;
        cin >> x >> y >> z;
        add(x,y,z);
    }
    if(spfa()) cout << "impossible" << endl;
    else cout << dist[n] << endl;
}

```

- 初始化
 - 初始化cnt数组为0
 - 初始化st数组为false
 - dist数组初始化为-1

```

#include<iostream>
#include<queue>
#include<cstring>
using namespace std;
const int N = 2010, M = 10010;

int n,m;

int e[M],ne[M],idx,w[M],h[N],dist[N],cnt[N];
bool st[N];
void add(int x,int y,int z){
    e[idx] = y,ne[idx] = h[x],w[idx] = z,h[x] = idx ++;
}

bool spfa(){
    queue<int> q;
    for(int i = 1;i <= n;i++){
        q.push(i);
        st[i] = true;
    } // 初始化000001
    while(q.size()){
        auto t = q.front();
        q.pop();
        st[t] = false;
        for(int i = h[t];i != -1;i = ne[i]){
            int j = e[i];
            if(dist[j] > dist[t] + w[i]){ // 松弛操作
                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);

```

```

    cin >> n >> m;
    while(m--){
        int x,y,z;
        cin >> x >> y >> z;
        add(x,y,z);
    }
    if(spfa()) cout << "Yes" << endl;
    else cout << "No" << endl;
}

```

Floyd

- 模板
- 模板题

```

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

const int N = 210;
int n,m,k;
int d[N][N];

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(){
    memset(d,0x3f,sizeof d);
    cin >> n >> m >> k;
    for(int i = 1;i <= n;i++) d[i][i] = 0;
    while(m--){
        int x,y,z;
        cin >> x >> y >> z;
        d[x][y] = min(d[x][y],z);
    }
    floyd();
    while(k--){
        int x,y;
        cin >> x >> y;
        if(d[x][y] >= 0x3f3f3f3f / 2) cout << "impossible" << endl;
        else cout << d[x][y] << endl;
    }
}

```

```
}  
}
```