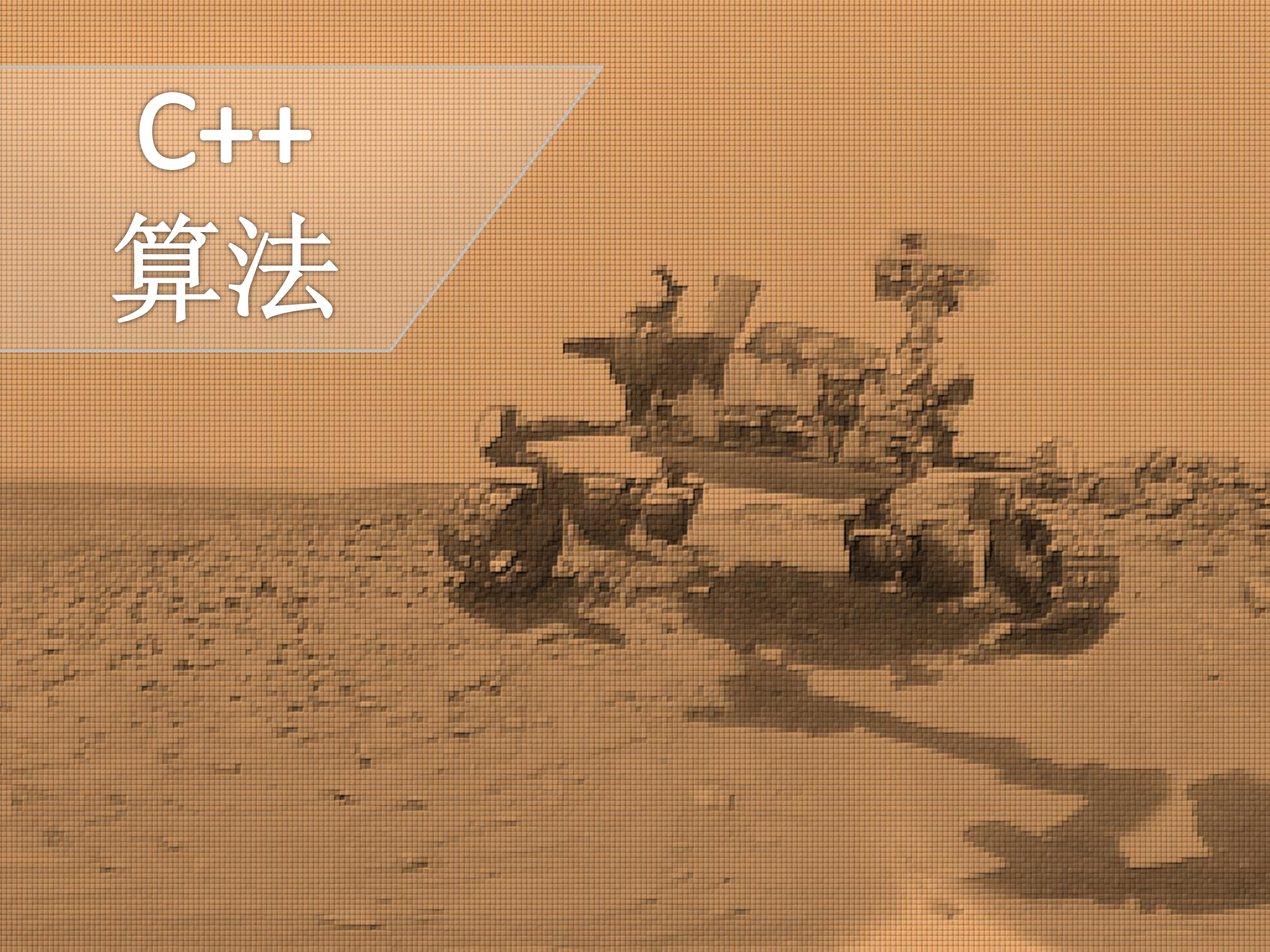
C++ 算法



现场挑战

罗密欧与朱丽叶

你是罗密欧，要去找朱丽叶。共有n个城市，编号1到n，你在1号城市，朱丽叶在n号。城市间共有m条双向道路，路程长度都已知。求你去找朱丽叶的路径中最长一段道路最短是多少？若无法到达输出-1。

第一行为正整数n和m，n<=1000，m<=10000。接着m行为道路信息，每一行为正整数a,b,l代表a号和b号城市间有一条长度为l的路。1<=a,b<=n。l<=1000. 输出一个整数

输入样例：
2 2
1 2 7
2 1 6

输出样例：
6

输入样例：
3 4
1 2 5
2 3 6
3 2 4
1 3 7

输出样例：
5

最长边最短路!

用纸和笔写出：
1.算法步骤
2.复杂度

限时10分钟

破题路径

如何想到解法?

从题面探索出答案
推荐3种常见路径

算法分类	依次尝试几种基本算法思想 贪心**T**,枚举**M**,动归**D**,数学S
建模元素	识别问题要素,套用常规手段
经典算法	联系新问题与经典老题 改写老题的解法

算法1：二分答案+判连通性

算法	枚举答案x：二分x	请写出OK(x)含义
	判可行性：能否连通	DFS/BFS/UFDS

如何想到这个算法?

破题路径：有2条路可以想到这个算法

算法分类	依次尝试 TMDS	尝试M: 枚举答案 再判断可行性
建模元素	识别要素 "最大值最小化"	常规手段: 二分答案

算法1：二分答案+判连通性

现场挑战
用电脑实现
二分+DFS
解决"罗密欧与朱丽叶"

限时15分钟

算法1：二分答案+判连通性

```
bool OK(){
	fill(vst,vst+n+1,0);
	dfs(1);
	return 
}
```

```
	ll l=0,r=1000,ans=1000;
	while(l<=r){
		mid=l+(r-l)/2;
		if(OK())
		else 
	}
	cout<<ans<<endl;
```

算法1：二分答案+判连通性

算法

枚举答案x：二分x	请写出OK(x)含义
判可行性：能否连通	DFS/BFS/UFDS

连通性判断3件套

3种方法有何优劣?

DFS	BFS	UFDS并查集

可以在不断加边时动态查询连通性

观察发现

二分+并查集判连通性时每次判断都要重新加边无法利用并查集优势

算法2：Kruskal变种

观察发现	二分答案后每次判可行性 都要清空并查集，形成浪费	
优化枚举顺序	顺序枚举答案 并查集不清空 累积可以使用的边	边长从小到大 不断加边 直到源汇连通
	Kruskal框架	原用于MST
	MST上路径就是 最长边最短路径	
	不用求完MST，可以提前结束	

最小生成树MST

最大边最小生成树
最小生成树
两者有啥区别？为什么？

最大生成树MST

最小边最大生成树
最大生成树
有啥区别？为什么？

算法2：Kruskal变种

如何想到这个算法?

破题路径：有3条路可以想到这个算法

算法分类	尝试M: 枚举答案但不二分答案 顺序枚举答案再判断可行性	发现相邻两答案有关系
算法分类	尝试T贪心: 边长从小到大 依次添加，直到连通为止	
建模元素	识别要素 "最长边最短路径"	MST上路径就是 最长边最短路径

算法2：Kruskal变种

现场挑战
用电脑实现
Kruskal变种
解决"罗密欧与朱丽叶"

限时15分钟

算法2：Kruskal变种

```
ll kruskal(){
	sort(e,e+m,cmp);
	for(ll i=1;i<=n;i++)id[i]=i;
	ll ans=0;
	for(ll k=0;k<m;k++){
		ll fa=find(e[k].a),fb=find(e[k].b);
		if(fa==fb)continue;
		id[fa]=fb;
		if(                )return          
	}
	return      
}
```

算法3：Dijkstra变种

仿照Dijkstra步骤框架，但重新定义d[i]为从1号点到i号间，最长边最短是几

d[]数组初始INF，d[1]为0

每次找d[i]最小元素确定为最终值

再更新i的邻居们

算法3：Dijkstra变种

```
void dijkstra(){
    fill(d,d+n+9,INF);
    fill(ok,ok+n+9,0);
    d[1]=0;
    for(ll k=1;k<=n;k++){
        ll u=n+1;
        for(ll v=1;v<=n;v++)
            if(!ok[v]&&d[v]<d[u])u=v;
        ok[u]=1;
        if(u==n||u==n+1)break;
        for(ll i=0;i<to[u].size();i++)
            d[to[u][i]]=min(d[to[u][i]],max(d[u],w[u][i]));
    }
}
```

算法3：Dijkstra变种

仿照Dijkstra步骤框架，但重新定义d[i]为从1号点到i号间，最长边最短是几

如何想到这个算法?

算法分类	尝试D动归: 自然定义状态d[i] 寻找依赖关系

算法分类	尝试T贪心: 从最容易确定答案的节点开始逐个确定，d[1]最容易

经典算法	联系"最长边最短路径"和"最短路径"问题相似	改编Dijkstra

总结思维路径

用纸和笔写出
"最大边最短路"问题
各种算法可能如何想到

作业

快快920
快快686
快快592