树相关

——ZZH

树相关

- 1、树的基本性质
- 2、重心和最长链
- 3、DFS序
- 4、最近公共祖先
- 5、树链剖分
- 6、点分治
- 树的生成与构造
- 总结与作业布置

例题声明

本课件中所有例题均满足树的点数等于100000。

1、树的基本性质

定义

树是一张由V个点、V-1条边组成的无环图。

例题一

[Codeforces746G] New Roads

题意：

要求构造一棵N个节点的树，以一号点为根，深度为D，每层的节点数数量分别为d[i]，且叶子数量为K。

如果无解输出-1。

例题一

[Codeforces746G] New Roads

标准算法：

先构造一棵叶子最少的树，然后往上面慢慢加叶子。

例题一

[Codeforces746G] New Roads

标准算法：

先构造一棵叶子最少的树，然后往上面慢慢加叶子。

挺简单的，是不是?

例题一

[Codeforces746G] New Roads

标准算法：

先构造一棵叶子最少的树，然后往上面慢慢加叶子。

挺简单的，是不是?

N=1的时候特判！！

2、重心与最长链

定义与性质

重心：到树上所有点距离之和最小的点。

定义与性质

重心：到树上所有点距离之和最小的点。

充要条件：最大的子树大小不超过全树的一半。

定义与性质

重心：到树上所有点距离之和最小的点。

充要条件：最大的子树大小不超过全树的一半。

通常可以一遍DFS求解

定义与性质

重心：到树上所有点距离之和最小的点。

充要条件：最大的子树大小不超过全树的一半。

通常可以一遍DFS求解

最长链：又称直径，连接树上最远点对的路径。

定义与性质

重心：到树上所有点距离之和最小的点。

充要条件：最大的子树大小不超过全树的一半。

通常可以一遍DFS求解

最长链：又称直径，连接树上最远点对的路径。

通常可以两遍BFS或者一遍DFS求解。

定义与性质

重心：到树上所有点距离之和最小的点。

充要条件：最大的子树大小不超过全树的一半。

通常可以一遍DFS求解

最长链：又称直径，连接树上最远点对的路径。

通常可以两遍BFS或者一遍DFS求解。

注意如果树有负权边，前者的方法会失效。

例题二

[Codeforces709E] Centroids

题意：

给出一棵树，对于树的所有节点，询问如果删掉一条边再加上一条边，能否将其改为重心。

例题二

[Codeforces709E] Centroids

标准算法：

考虑重心的充要条件：不存在任何一棵子树的点数超过总点数的一半。因此，对于每个非重心节点，减掉的那棵子树，必定是在包含重心子树中；进一步分析，必定是以重心为根的一棵子树；更进一步分析，必定是重心的重儿子。（当然要特判掉当前点在重心重儿子子树中的情况）

例题二

[Codeforces709E] Centroids

标准算法：

考虑重心的充要条件：不存在任何一棵子树的点数超过总点数的一半。因此，对于每个非重心节点，减掉的那棵子树，必定是在包含重心子树中；进一步分析，必定是以重心为根的一棵子树；更进一步分析，必定是重心的重儿子。（当然要特判掉当前点在重心重儿子子树中的情况）

因此，对树遍历就可以解决这个问题了。

3、DFS序

定义与性质

有很多种，要结合题目灵活使用。

定义与性质

有很多种，要结合题目灵活使用。

常用的三种：入栈序、入栈出栈序、欧拉序。

定义与性质

有很多种，要结合题目灵活使用。

常用的三种：入栈序、入栈出栈序、欧拉序。

可是这东西有什么用？

定义与性质

有很多种，要结合题目灵活使用。

常用的三种：入栈序、入栈出栈序、欧拉序。

可是这东西有什么用？

把对子树的操作问题转化为对区间操作的问题。

定义与性质

有很多种，要结合题目灵活使用。

常用的三种：入栈序、入栈出栈序、欧拉序。

可是这东西有什么用？

把对子树的操作问题转化为对区间操作的问题。

常结合线段树使用。

定义与性质

有很多种，要结合题目灵活使用。

常用的三种：入栈序、入栈出栈序、欧拉序。

可是这东西有什么用？

把对子树的操作问题转化为对区间操作的问题。

常结合线段树使用。

如果再结合上树链剖分，就可以处理对树链的操作问题了。

声明

考虑到线段树还不作要求，在这里不作展开。

建议大家赶快去自学一下线段树。

4、最近公共祖先

定义与性质

简称LCA。

定义与性质

最近公共祖先

简称LCA。

在一棵有根树中，连接两点的路径上深度最浅的点就是这两点的LCA。

定义与性质

简称LCA。

在一棵有根树中，连接两点的路径上深度最浅的点就是这两点的LCA。

能够把树上路径相关的问题拆分成两段分别维护。

求LCA的方法

倍增算法

Tarjan算法

DFS+ST算法

树链剖分算法

比较求LCA的方法

最近公共祖先

倍增算法（在线，可以维护较多信息，支持加叶子）

Tarjan算法（离线，局限性较大）

DFS+ST算法（在线，可以用±1RMQ优化到O(1)）

树链剖分算法（在线，速度较快）

倍增表

f[i][j]表示从i出发走2^j步能够到达的位置。

倍增表

f[i][j]表示从i出发走2^j步能够到达的位置。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

倍增表

f[i][j]表示从i出发走2^j步能够到达的位置。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

用倍增法求LCA的第一步，是用这种方法将每个点的log个祖先预处理出来。

倍增表

f[i][j]表示从i出发走2^j步能够到达的位置。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

用倍增法求LCA的第一步，是用这种方法将每个点的log个祖先预处理出来。

时间复杂度O(N log N)

倍增LCA

最近公共祖先

f[i][j]表示从i出发走2^j步能够到达的位置。

用倍增法求LCA的第一步，是用这种方法将每个点的log个祖先预处理出来。

之后如果询问x，y的LCA，首先将x和y提到同一深度的位置。

倍增LCA

最近公共祖先

f[i][j]表示从i出发走2^j步能够到达的位置。

用倍增法求LCA的第一步，是用这种方法将每个点的log个祖先预处理出来。

之后如果询问x，y的LCA，首先将x和y提到同一深度的位置。

再把x和y同时往上提，直到他们的父亲相同为止。

倍增LCA

最近公共祖先

f[i][j]表示从i出发走2^j步能够到达的位置。

用倍增法求LCA的第一步，是用这种方法将每个点的log个祖先预处理出来。

之后如果询问x，y的LCA，首先将x和y提到同一深度的位置。

再把x和y同时往上提，直到他们的父亲相同为止。

单次询问时间复杂度O(log N)。

倍增表

反观倍增表维护的过程。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

在此基础上，我们还可以维护很多东西：

倍增表

反观倍增表维护的过程。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

在此基础上，我们还可以维护很多东西：

L[i][j]=L[i][j-1]+L[f[i][j-1]][j-1]

c[i][j]=min(c[i][j-1],c[f[i][j-1]][j-1])

倍增表

反观倍增表维护的过程。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

在此基础上，我们还可以维护很多东西：

L[i][j]=L[i][j-1]+L[f[i][j-1]][j-1]
//计算树链权值和（询问树上两点距离）
c[i][j]=min(c[i][j-1],c[f[i][j-1]][j-1])
//计算树链上的最小权值

倍增表

反观倍增表维护的过程。

递推建表：f[i][j]=f[f[i][j-1]][j-1]

在此基础上，我们还可以维护很多东西：

```
L[i][j]=L[i][j-1]+L[f[i][j-1]][j-1]
//计算树链权值和
c[i][j]=min(c[i][j-1],c[f[i][j-1]][j-1])
//计算树链上的最小权值
q[i][j]=q[i][j-1]*pow10[1<<(j-1)]+q[f[i][j-1]][j-1]
//计算将树链上所有边连起来形成的整数
```

树上倍增算法

最近公共祖先

利用倍增表，主要可以处理一类静态树上链查询问题。

例题三

[Codeforces609E]
Minimum spanning tree for each edge

题意：

给出一张无向图，依次强制每一条边在生成树上，要求生成树的边权和最小。

例题三

[Codeforces609E]
Minimum spanning tree for each edge

标准算法：

如果不是最小生成树上的边，那么删去后对答案并不产生影响；

例题三

[Codeforces609E]
Minimum spanning tree for each edge

标准算法：

如果不是最小生成树上的边，那么删去后对答案并不产生影响；

如果是最小生成树上的边，删去后，必须有一条边来替代。

例题三

[Codeforces609E]
Minimum spanning tree for each edge

标准算法：

如果不是最小生成树上的边，那么删去后对答案并不产生影响；

如果是最小生成树上的边，删去后，必须有一条边来替代。

每一条不是最小生成树上的边，加到最小生成树上后，必然产生一个环，用贪心的策略，环上面权值最大的边应该被取代。

例题三

[Codeforces609E]
Minimum spanning tree for each edge

标准算法：

如果不是最小生成树上的边，那么删去后对答案并不产生影响；

如果是最小生成树上的边，删去后，必须有一条边来替代。

每一条不是最小生成树上的边，加到最小生成树上后，必然产生一个环，用贪心的策略，环上面权值最大的边应该被取代。

可以用倍增法维护树上路径上的最大边权。

DFS+ST算法

前面讲到了DFS序。
相信何梓滢应该也已经给你们讲过ST表了。

DFS+ST算法

前面讲到了DFS序。
相信何梓滢应该也已经给你们讲过ST表了。

如果把它们结合起来.....

DFS+ST算法

前面讲到了DFS序。
相信何梓滢应该也已经给你们讲过ST表了。

如果把它们结合起来.....

维护欧拉序及各点的深度d[i]，对深度建ST表，并记录每个点第一次出现的位置R[i]。

DFS+ST算法

前面讲到了DFS序。
相信何梓滢应该也已经给你们讲过ST表了。

如果把它们结合起来.....

维护欧拉序及各点的深度d[i]，对深度建ST表，并记录每个点第一次出现的位置R[i]。

询问x和y的LCA，等价于求d[R[x]],d[R[x]+1],d[R[x]+2].....d[R[y]-1],d[R[y]]的最小值。

DFS+ST算法

前面讲到了DFS序。
相信何梓滢应该也已经给你们讲过ST表了。

如果把它们结合起来.....

维护欧拉序及各点的深度d[i]，对深度建ST表，并记录每个点第一次出现的位置R[i]。

询问x和y的LCA，等价于求d[R[x]],d[R[x]+1],d[R[x]+2].....d[R[y]-1],d[R[y]]的最小值。

而这个可以用ST表维护。

例题四

[USACO 2010 Holiday] Cow Politics

题意：

给出一棵树，第i个点的颜色为c[i]。

对于所有的颜色，求树上该种颜色的最远点对距离。

保证每种颜色的点至少有两个。

例题四

[USACO 2010 Holiday] Cow Politics

标准算法：

任意定根，考虑距离最远的两个点，必然有一个点是该属性在树上位置最深的点。

例题四

[USACO 2010 Holiday] Cow Politics

标准算法：

任意定根，考虑距离最远的两个点，必然有一个点是该属性在树上位置最深的点。

因此，找到每个属性位置最深的点后，暴力求LCA即可。

声明

对Tarjan算法和±1RMQ感兴趣的同学可以自主学习。

5、树链剖分

定义与性质

令一个节点儿子中子树最大的为重儿子。

定义与性质

令一个节点儿子中子树最大的为重儿子。

令每个节点到其重儿子的边为重边，重边连成重链。

定义与性质

令一个节点儿子中子树最大的为重儿子。

令每个节点到其重儿子的边为重边，重边连成重链。

可以证明，任意节点到根的路径上，重链不超过log条。

树链剖分求LCA

记一个点的相对深度为根到它经过的轻边数量。

树链剖分求LCA

记一个点的相对深度为根到它经过的轻边数量。

求LCA时，不断地提深度较深的点，直到两点出现在同一条重链上。

树链剖分求LCA

记一个点的相对深度为根到它经过的轻边数量。

求LCA时，不断地提深度较深的点，直到两点出现在同一条重链上。

记一个点的绝对深度为根到它经过的边数。这时，绝对深度较小的点就是这两点的LCA。

树链剖分求LCA

记一个点的相对深度为根到它经过的轻边数量。

求LCA时，不断地提深度较深的点，直到两点出现在同一条重链上。

记一个点的绝对深度为根到它经过的边数。这时，绝对深度较小的点就是这两点的LCA。

因为树链剖分求LCA的复杂度并不满，所以一般跑得更快一些；但是理论复杂度和倍增是一样的。

树链剖分与DFS序

如果DFS时特殊处理，保持重链DFS序的连续性？

树链剖分与DFS序

如果DFS时特殊处理，保持重链DFS序的连续性？

在线段树的维护下，就可以支持动态链上操作！

树链剖分与DFS序

如果DFS时特殊处理，保持重链DFS序的连续性？

在线段树的维护下，就可以支持动态链上操作！

通常有单次修改O(log^2)的复杂度保证。

例题五

[Codeforces600E] Lomsat gelral

题意：

给出一棵树，求其所有子树内出现次数最多的颜色编号和。

例题五

[Codeforces600E] Lomsat gelral

标准算法：

对整棵树进行树链剖分。

例题五

[Codeforces600E] Lomsat gelral

标准算法：

对整棵树进行树链剖分。

然后DFS统计答案。对于每个节点，按照依次DFS轻儿子、清空轻儿子标记——遍历重儿子——重新计算轻儿子标记并与重儿子标记合并——计算当前点答案来处理。

例题五

[Codeforces600E] Lomsat gelral

标准算法：

对整棵树进行树链剖分。

然后DFS统计答案。对于每个节点，按照依次DFS轻儿子、清空轻儿子标记——遍历重儿子——重新计算轻儿子标记并与重儿子标记合并——计算当前点答案来处理。

看起来复杂度是O(N^2)？

例题五

[Codeforces600E] Lomsat gelral

标准算法：

对整棵树进行树链剖分。

然后DFS统计答案。对于每个节点，按照依次DFS轻儿子、清空轻儿子标记——遍历重儿子——重新计算轻儿子标记并与重儿子标记合并——计算当前点答案来处理。

看起来复杂度是O(N^2)？

注意了，我们的做法是基于树链剖分的。因为每个点到根路径上的轻边不会超过log条，所以每个点作为轻儿子最多只会被重复计算log次。时间复杂度O(N log N)。

6、点分治

定义与性质

点分治算法的主要过程是找到树的重心，完成重心的统计后，递归各个子树进行下一层统计。

定义与性质

点分治算法的主要过程是找到树的重心，完成重心的统计后，递归各个子树进行下一层统计。

因为每次经过一个重心，子树的大小就会至少减小一半，所以整个递归的深度是log级别的。

定义与性质

点分治算法的主要过程是找到树的重心，完成重心的统计后，递归各个子树进行下一层统计。

因为每次经过一个重心，子树的大小就会至少减小一半，所以整个递归的深度是log级别的。

这种算法通常用于解决一类静态树上路径统计问题。

例题六

[IOI2011] Race

题意：

给出一棵带权树，求树上有多少条长度为K的路径。

K≤1000000

例题六

[IOI2011] Race

标准算法：

点分治。

例题六

[IOI2011] Race

标准算法：

点分治。

每层重心处统计经过当前重心符合条件的树链数量。可以用桶来统计答案。

树的生成与构造

随机树的常用生成方式

方法一：

对于点i，随机向编号为1——i-1的点连边。

随机树的常用生成方式

方法一：

对于点i，随机向编号为1——i-1的点连边。

生成树的性质：树高稳定在log级别。编号越小的点树度期望越大。

随机树的常用生成方式

方法一：

对于点i，随机向编号为1——i-1的点连边。

生成树的性质：树高稳定在log级别。编号越小的点树度期望越大。

优点：容易实现，能够较好地用于一般的调试与对拍。

随机树的常用生成方式

方法一：

对于点i，随机向编号为1——i-1的点连边。

生成树的性质：树高稳定在log级别。编号越小的点树度期望越大。

优点：容易实现，能够较好地用于一般的调试与对拍。

缺点：树高并不大以至于一些暴力算法都能过去，造出的数据缺乏杀伤力。

随机树的常用生成方式

方法二：

用并查集维护树的生成过程：随机两个点，如果他们已经在同一集合里则不管，否则把它们连上边。

随机树的常用生成方式

方法二：

用并查集维护树的生成过程：随机两个点，如果他们已经在同一集合里则不管，否则把它们连上边。

生成树的性质：（由随机多次测试得到）

点数	10^1	10^2	10^3	10^4	10^5	10^6
期望树高	5	25	65	170	400	700

随机树的常用生成方式

方法二：

用并查集维护树的生成过程：随机两个点，如果他们已经在同一集合里则不管，否则把它们连上边。

优点：随机性强。

随机树的常用生成方式

方法二：

用并查集维护树的生成过程：随机两个点，如果他们已经在同一集合里则不管，否则把它们连上边。

优点：随机性强。

缺点：对于较大的数据生成时间较长。

数据的加强

随机数生成的树一般不能卡掉暴力算法，这时候就需要加强数据。

数据的加强

一种常用的思路是削弱随机的效果，这种思路常通过修改随机参数来实现。

数据的加强

一种常用的思路是削弱随机的效果，这种思路常通过修改随机参数来实现。

另一种常用的思路是增加构造的成分，半随机或者不随机，使数据的杀伤力更强。但是，这种构造方法通常会带有比较大的偶然性。

数据的加强

一种常用的思路是削弱随机的效果，这种思路常通过修改随机参数来实现。

另一种常用的思路是增加构造的成分，半随机或者不随机，使数据的杀伤力更强。但是，这种构造方法通常会带有比较大的偶然性。

树链、菊花树、满二叉树等都是构造树的常见形式。

数据的加强

一种常用的思路是削弱随机的效果，这种思路常通过修改随机参数来实现。

另一种常用的思路是增加构造的成分，半随机或者不随机，使数据的杀伤力更强。但是，这种构造方法通常会带有比较大的偶然性。

树链、菊花树、满二叉树等都是构造树的常见形式。

造数据不是随机得越多越好，也不是构造得越多越好。要把握好尺度，追求两者的平衡。

总结与作业布置

总结

本节课讲到的内容中，前四块必须掌握，后两块不作要求。

总结

本节课讲到的内容中，前四块必须掌握，后两块不作要求。

前四块均为树的基本知识，但是，将来你可能会发现，树的基础知识点远远不止这些，更多的题目需要你去变通。

总结

本节课讲到的内容中，前四块必须掌握，后两块不作要求。

前四块均为树的基本知识，但是，将来你可能会发现，树的基础知识点远远不止这些，更多的题目需要你去变通。

希望第七块的内容能够对大家调试程序有所帮助。

总结

还有你或许会发现，没有线段树，本节课一半内容的真正威力无法发挥出来！线段树将来会有其他同学细讲，但建议大家提早学习，越早越好！

总结

还有你或许会发现，没有线段树，本节课一半内容的真正威力无法发挥出来！线段树将来会有其他同学细讲，但建议大家提早学习，越早越好！

如果你学了线段树，请你再回过头来看一遍第三块、第五块和第六块。它们还是它们原来的样子吗？！

作业布置

作业中的题目要求必做，扩展要求中的题目稍难，可以根据自己的情况选做。

作业

掌握所有的例题，并解决下列题目：

[洛谷1268] 树的重量
[AHOI2008,BZOJ1787] 紧急集合
[Codeforces519E] A and B and Lecture Rooms
[Codeforces686D] Kay and Snowflake
[Codeforces701E] Connecting Universities
[Codeforces702E]
Analysis of Pathes in Functional Graph

扩展要求（可根据自己的水平选做）

[POJ2763] Housewife Wind
[POJ3237] Tree
[ZJOI2008,BZOJ1036] 树的统计
[BZOJ2152] 聪聪可可
[Codeforces593D] Happy Tree Party
[Codeforces716E] Digit Tree
[Codeforces741D]
Arpa’s letter-marked tree and
Mehrdad’s Dokhtar-kosh paths

谢谢观看