1. 0-Indegee Toposort: Ambiguity 11A1

本节证明了，有向无环图必然可以拓扑排序，但可能存在多个排序。

a) 拓扑排序的不唯一性，具体由什么因素决定?

b) 什么样的有向无环图，拓扑排序是唯一的?

2. 0-Outdegree Toposort: Edge Classification 11A2

基于DFS的拓扑排序算法，同样会试图将图中的边划分为四类。对拓扑排序而言，这四类边各有什么含义?

3. 0-Outdegree Toposort: Recursion vs. Iteration 11A2

本节基于DFS实现的拓扑排序算法是递归式的，试将其改写为等效的迭代版。

4. BCC: Criteria 11B1

本节针对DFS树的叶节点、根节点和内部节点，分别给出了关节点的判定准则，试逐条确认这些准则。

5. BCC: Algorithm 11B2

本节所列算法，使用一个全局有效的栈来顺次记录访问的顶点。

a) 试证明：每当检出一个BCC时，除了其对应的起点v外，其余顶点都集中存放在栈顶;

b) 试在讲义所列的实例中指认，即便新检出BCC中的其余顶点均已弹出，v确实仍可能不是栈顶;

c) 上述现象在什么条件下会发生?

6. PFS: BFS & DFS 11C

从理论上讲，本节引入的PFS覆盖了所有的图搜索算法，包括此前的BFS和DFS。也就是说，只要适当地定义顶点的优先级，并实现对应的优先级更新器，便可通过统一的PFS框架来模拟BFS和DFS。

a) 在BFS和DFS的过程中，图中各顶点的优先级是按什么规则不断更新，直至确定的?

b) 试将你所归纳出来的更新规则，描述并实现为对应的优先级更新器BfsPU()和DfsPU();

c) 阅读示例代码中这两个优先级更新器的实现，与你的实现做一对照。

7. Dijkstra 11D

如果带权图中包含负权边，而且存在总权重为负的环路，则不难理解，最短路径将无法定义，更遑论计算。

a) 试说明，尽管含有负权边，但只要没有负权环路，Dijkstra在经过适当调整后，依然可以构造出SPT;

b) 这种调整会否导致时间复杂度增加？最坏情况下复杂度会高达多少?

c) 针对这类情况，你有什么进一步改进的办法?

8. Prim 11E

讲义中指出，鉴于可能存在等权的跨边，常规的证明方法并不能成立（尽管结论正确)。

a) 试阅读《习题解析》中对应的习题，了解严格的证明方法。

b) 试阅读《习题解析》中对应的习题，了解合称数、扰动等消除上述歧义的技巧。

9. Dijkstra vs. Prim 11[D+E]

这两节将Dijkstra算法与Prim算法纳入了PFS的框架，并基于该框架简明地加以实现。然而，毕竟这是两个不同的问题。比如正如讲义中提到的，Prim算法对各边的权重范围没有限制，即便含有负权边也依然可以明确地定义最优解，算法也不必做任何调整；反之，Dijkstra算法则需对边权有所限制。

试从Priority Updater的角度对二者做一对比，并解释上述差异。

10. Kruskal **11F1**

试证明，被Kruskal算法保留的边，都是应该被选用的；反之，被扔弃的边，都是不应选用的。

11. Kruskal **11F1**

在Kruskal算法中，一旦已经选出了$n-1$条边，便可随即结束算法。而此前被考查过的边实际上可能远远少于e条，于是算法初始阶段花费$\mathcal{O}(e \log n)$时间对所有边所做的排序，就显得很不明智。

试针对这一问题，提出你的优化方案。

12. Union-Find **11F2**

查阅相关资料，详细了解并查集的以下优化方法：

a) 在union(x,y)操作时，依照x与y的size确定合并的方向；

b) 在union(x,y)操作时，依照x与y的height确定合并的方向；

c) 每次完成find(x)操作的同时，将x在沿途经过的祖先都“折叠”起来，直接作为根的孩子。