15-16数据结构期末模拟题解析整合

Version 1.0

1. 判断题
2. If a graph is represented by adjacency lists, then the space taken depends only on the number of vertices, not the number of edges. F

邻接表的空间复杂度为O(V+E)

1. In hashing, functions "insert" and "find" have the same time complexity. T

插入和查找的时间复杂度均为O(1)

1. Shell sort is stable. F
2. To sort *N* records by merge sort, the number of merge runs is *O*(*NlogN*). F

the number of merge runs is O(logN)

1. Store a complete binary tree in an array (root at position 1). Then the nodes at positions 23 and 24 are siblings. F

23的父节点为11，24的父节点为12

1. For a sequentially stored linear list of length *N*, the time complexities for deleting the first element and inserting the last element are *O*(1) and *O*(*N*), respectively. F

删除第一个元素和插入最后一个元素的时间复杂度分别是O(N)和O(1)

1. In a directed graph, the sum of the in-degrees must be equal to the sum of the out-degrees of all the vertices. T
2. The Fibonacci number sequence {*FN*​} is defined as: *F*0​=0, *F*1​=1, *FN*​=*FN*−1​+*FN*−2​, *N*=2, 3, .... The time complexity of the function which calculates *FN*​ iteratively is Θ(*FN*​). F

递推的时间复杂度为O(N)

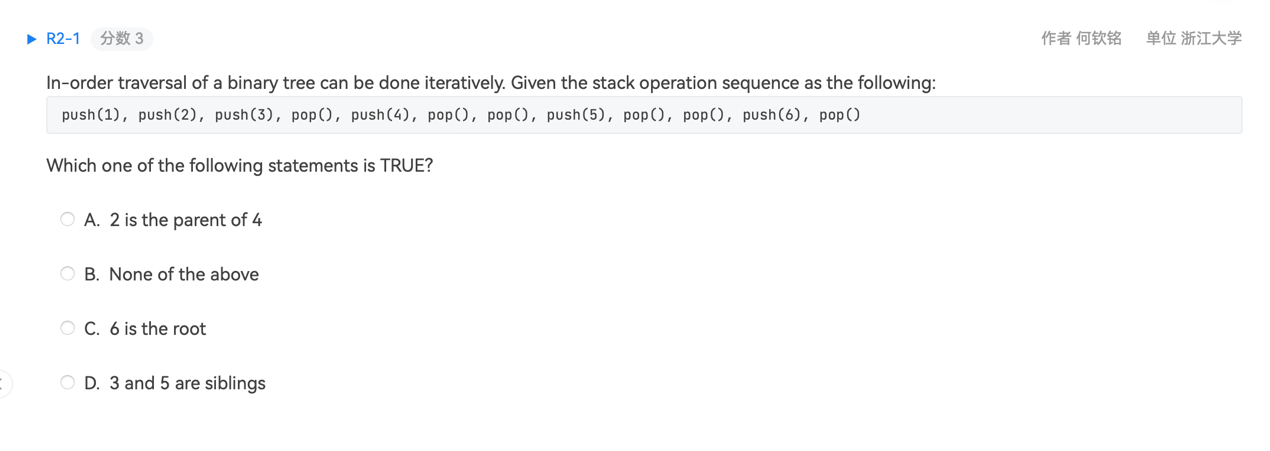
1. There exists a binary tree with 2016 nodes in total, and with 16 nodes having only one child. F

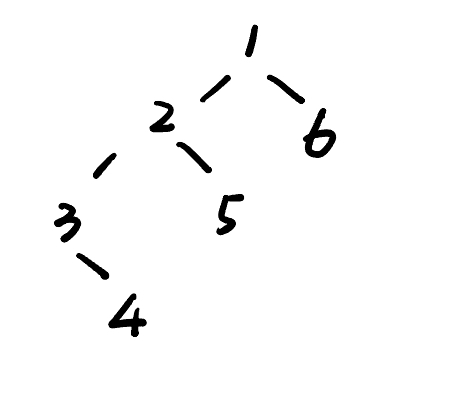
解出n2为非整数，所以二叉树不存在

1. In a graph G, if we have to do BFS twice to visit every one of its vertices, then there must be a cycle in G. F

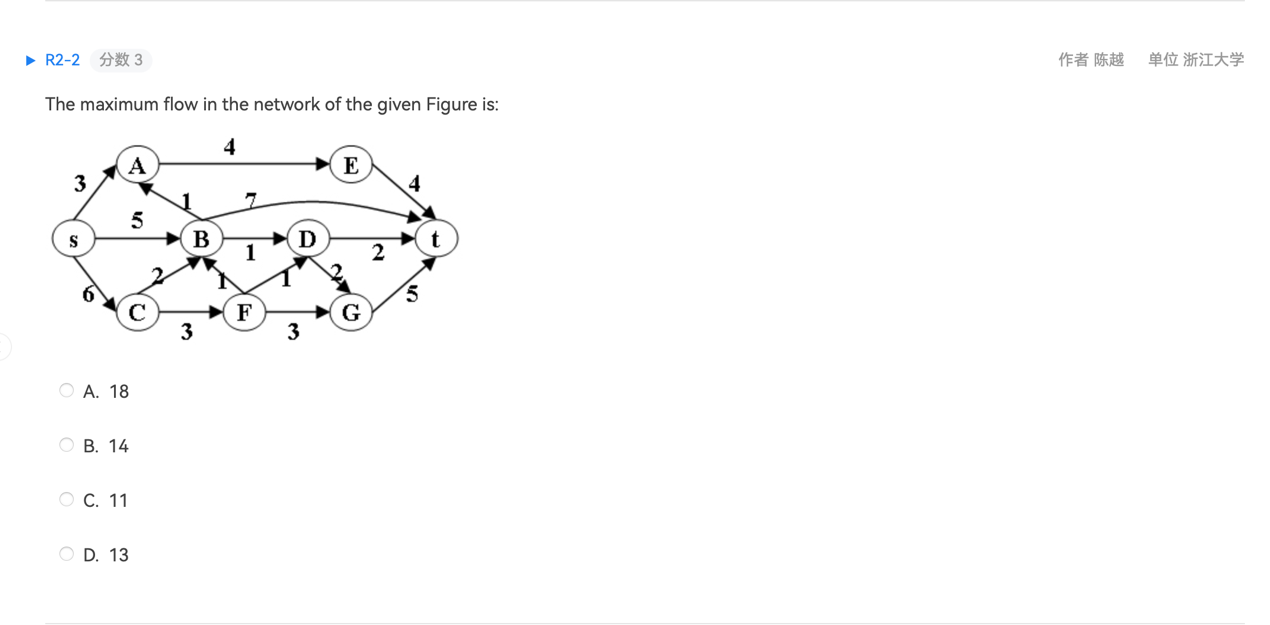
如1->2->3，4->5，这里并没有环，但是需要两次BFS才能遍历全图

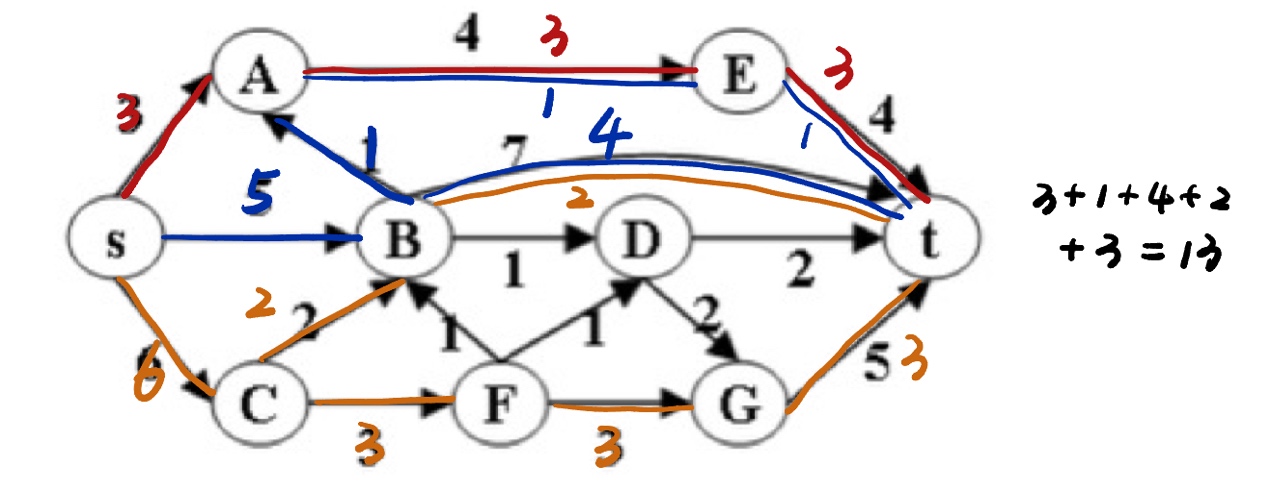
1. 单选题
2. D



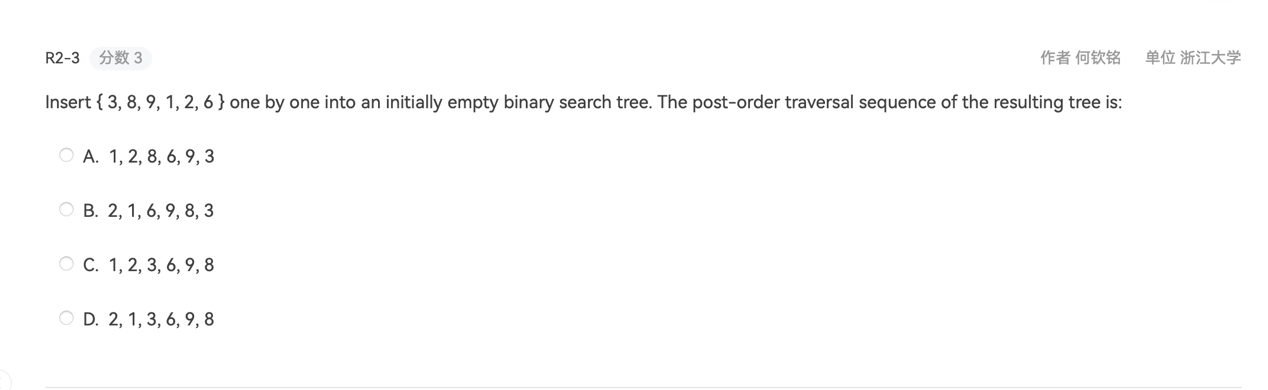


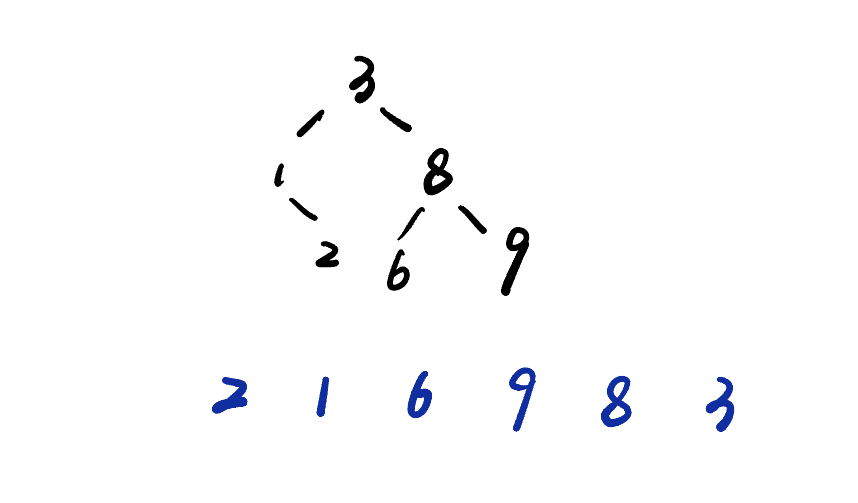
1. D



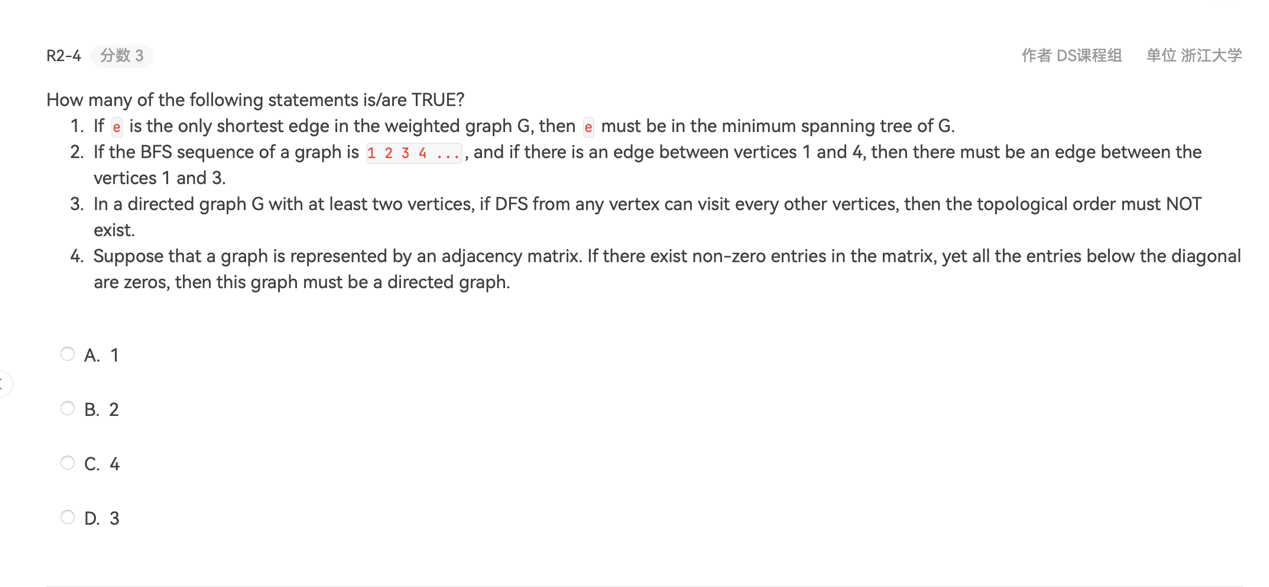


1. B

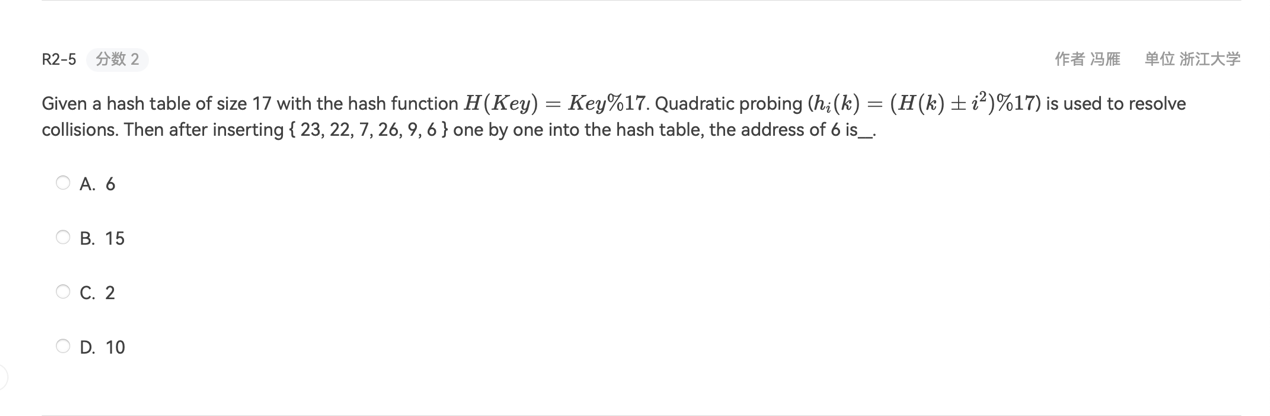




1. C



1. 由Kruskal算法可得证
2. BFS的本质是level-order traversal，所以2 3 4这三个点均位于同一层，都应与1有边
3. 由于以每个点开始的DFS都可以遍历其余所有点，所以可以推断每个点的入度均不为0，因而拓扑排序不存在
4. 这样的邻接矩阵不具备对称性，所以一定是有向图
5. C



这道题我做错了，因为我没注意到这里的函数是加减i平方，最终的分布情况应为

hash[2]=6

hash[5]=22

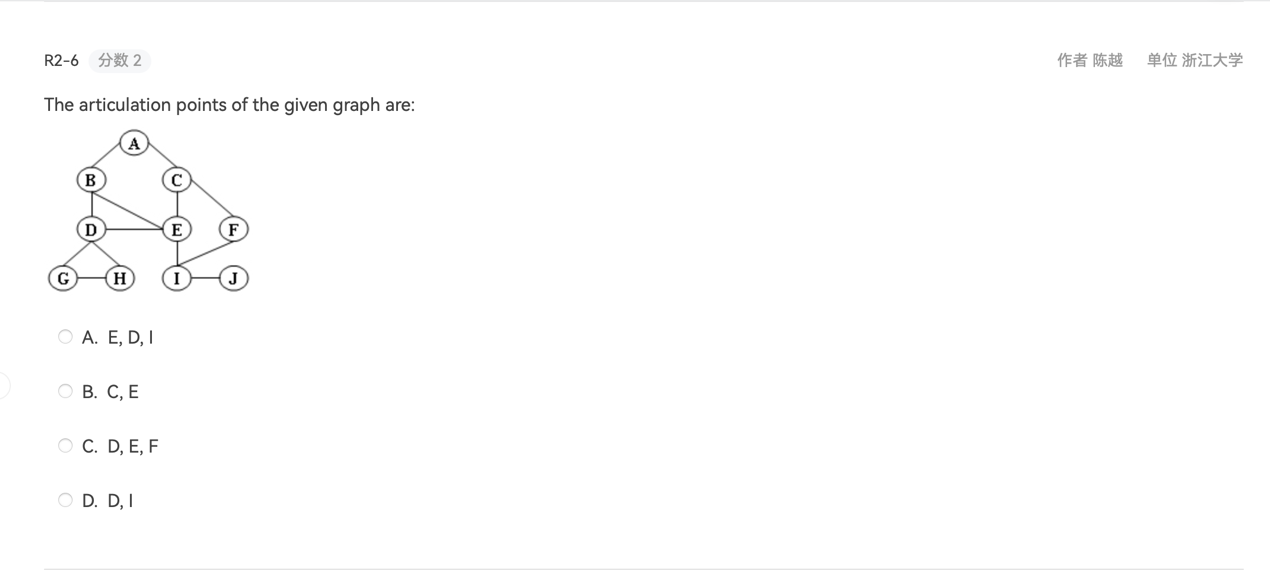
hash[6]=23

hash[7]=7

hash[9]=26

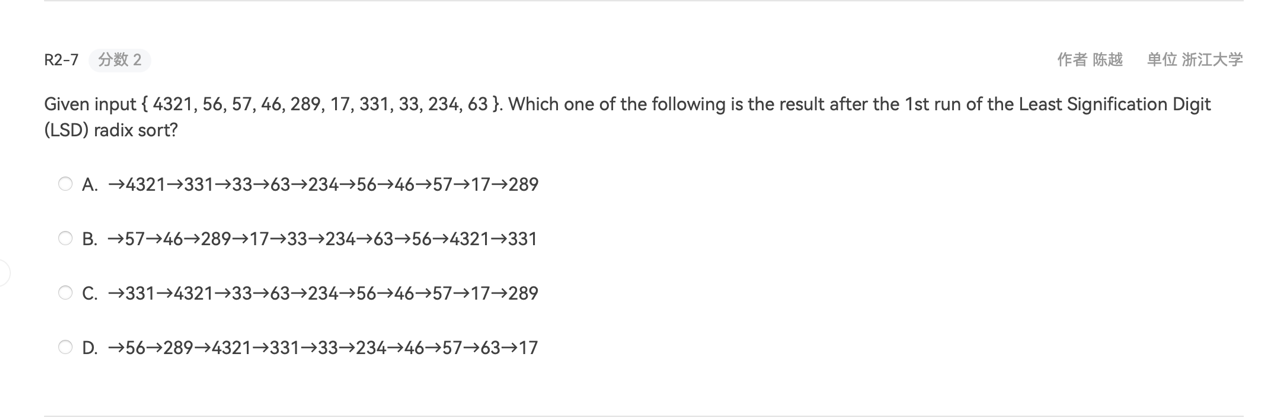
hash[10]=9

1. D



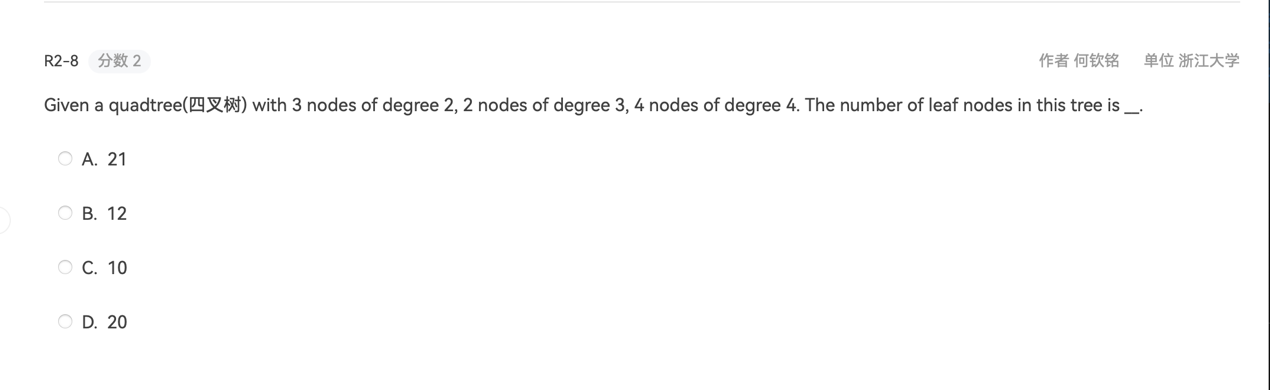
articulation point指的就是去掉之后会使原图分成两个强连通分量的点

1. A



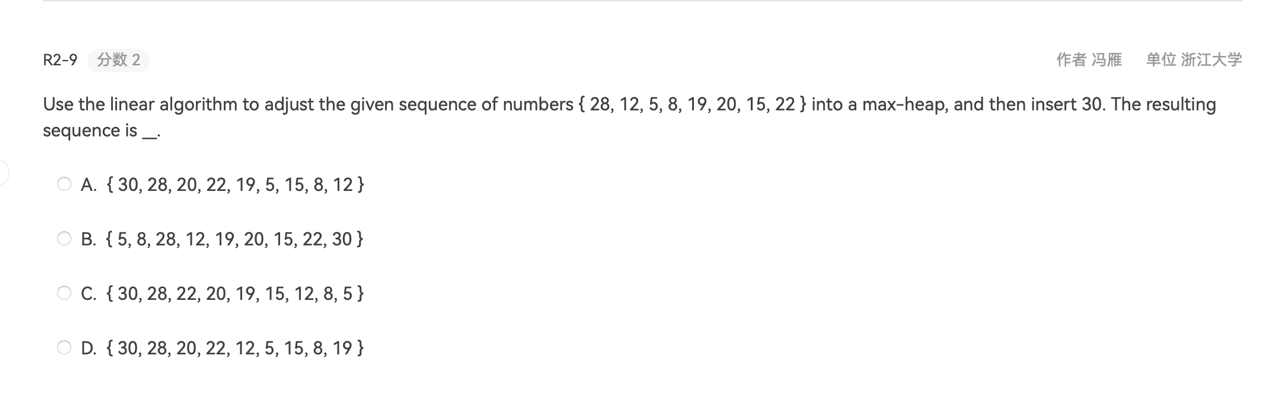
按照最低位排序，注意最低位相等时顺序不变

1. D



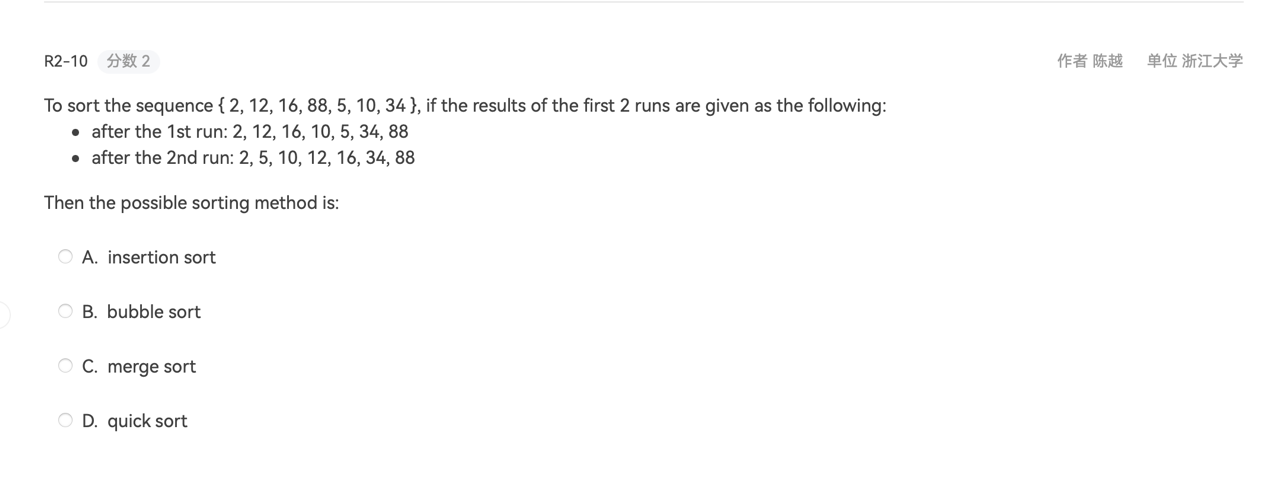
联立二式可求出n0为20

1. A



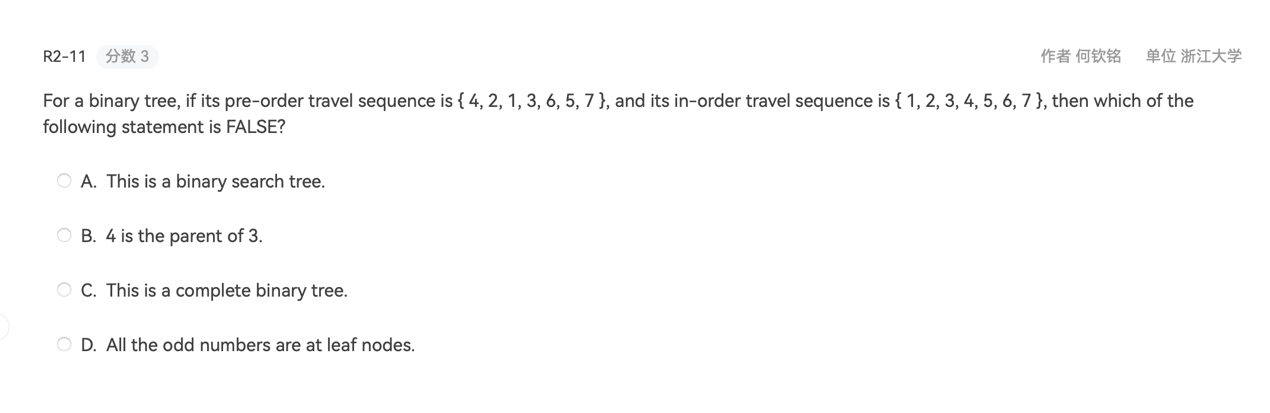
原序列变为最大堆后生成的新序列为{28, 22, 20, 12, 19, 5, 15, 8}，插入30到末尾后采取上滤，得到的序列为{30, 28, 20, 22, 19, 5, 15, 8, 12}

1. D



排序最终结果为2, 5, 10, 12, 16, 34, 88，在第一轮排序后，2和34归位，经过比对后会发现34是quicksort的基准数，恰好满足quicksort的结果，而其他排序算法都是无法得到上述序列的

1. B



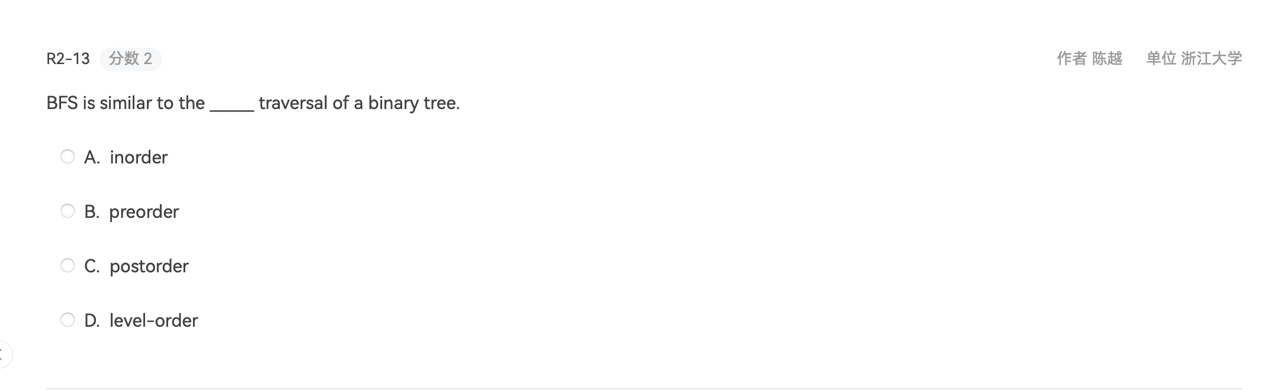
通过pre-order可以知道根节点为4，通过in-order可以知道根节点的左子树由1，2，3构成，右子树由5，6，7构成，对子树在采取相同的操作，就可以得到最终的二叉树为{4, 2, 6, 1, 3, 5, 7}，只有B不满足条件

1. A

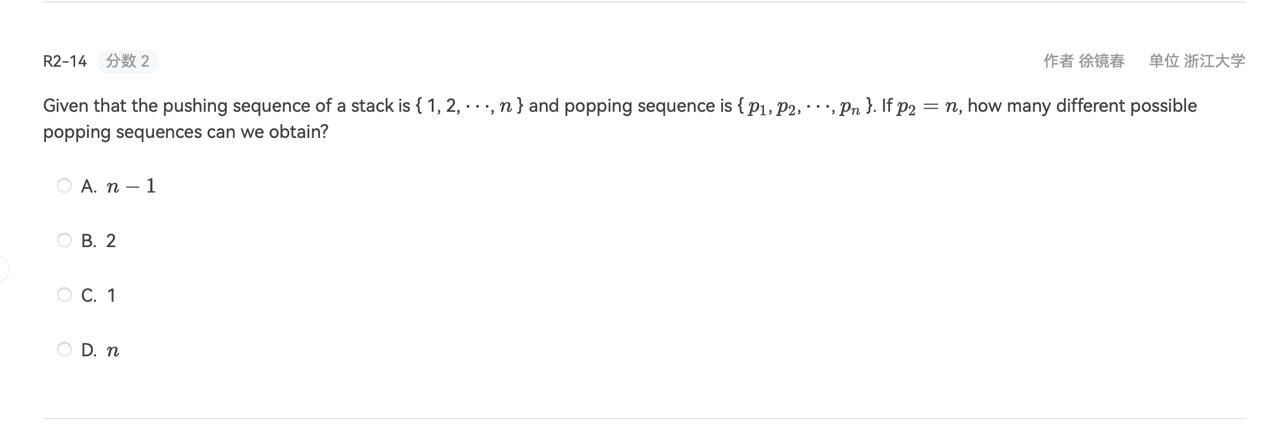


由题可以推断V1到V4之间一定是没有边的，否则会与题目的DFS的边集矛盾；而A项与V1相连的点还没遍历就到了V4，所以不是DFS序列

1. D

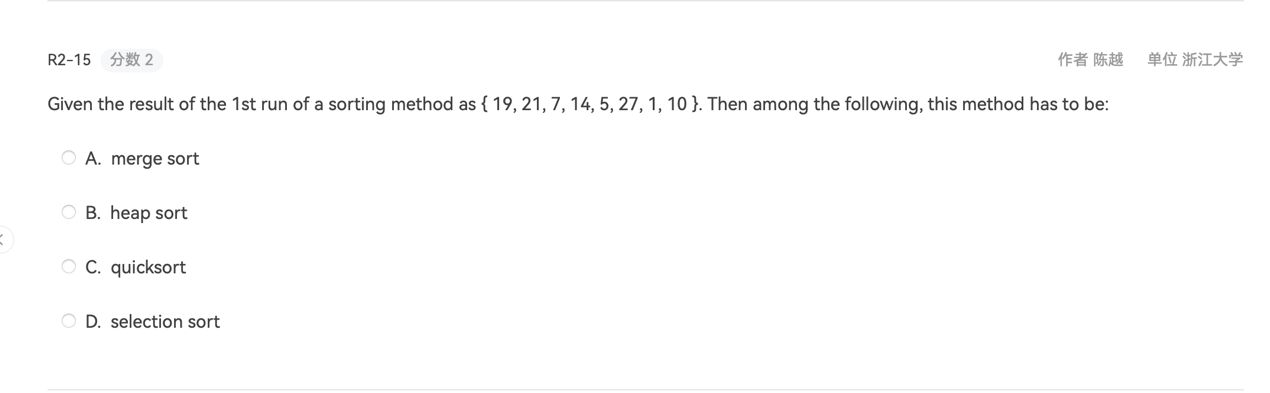


1. A



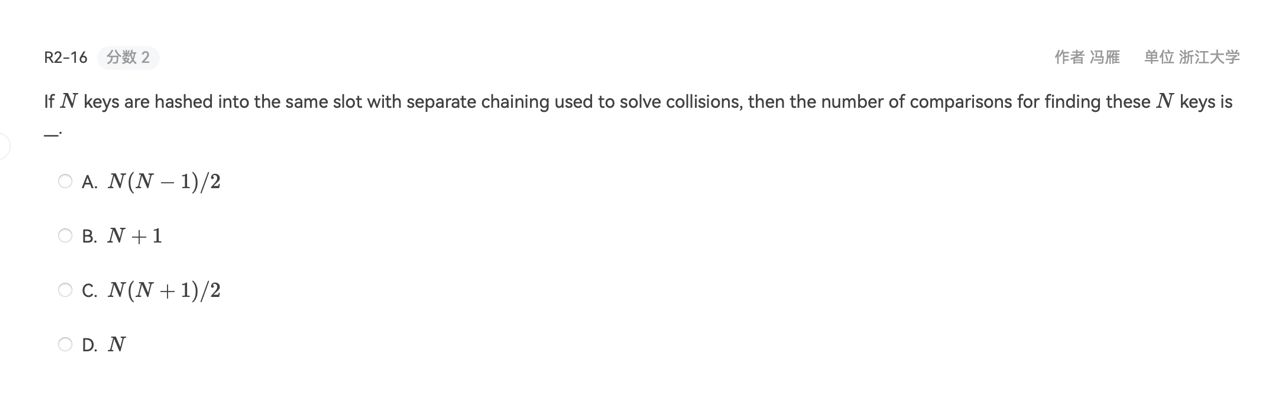
p1可以是1到n-1任意一个数，在n出栈后，其余数均在栈中，所以顺序是确定的，综上所述总共有n-1种情况

1. A

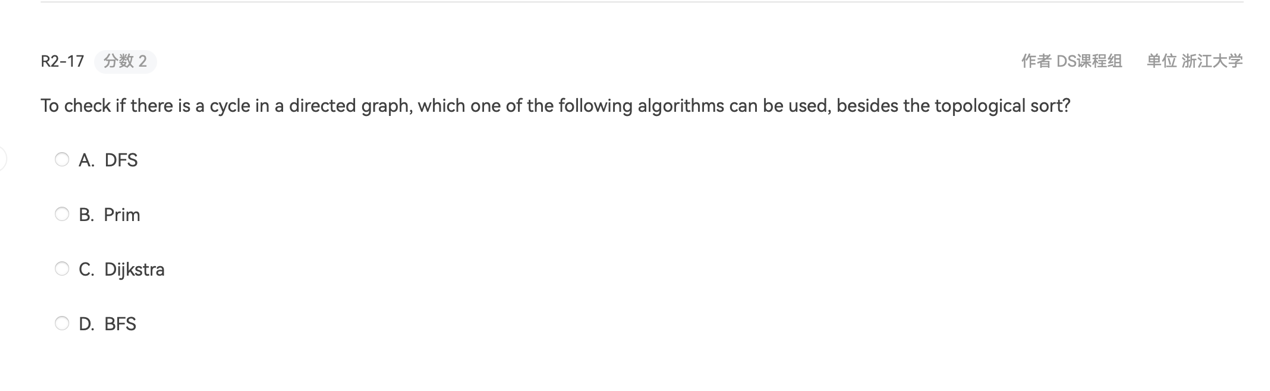


(19, 21) (7, 14) (5, 27) (1, 10)，很明显的归并特征

1. C

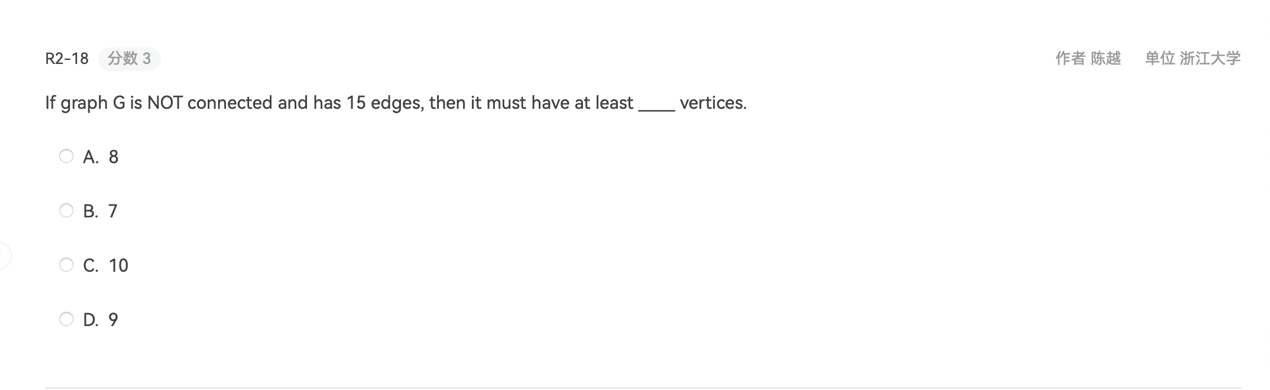


第k个关键词要比较k次，所以总共需要比较1+2+3+…+N=N(N+1)/2次



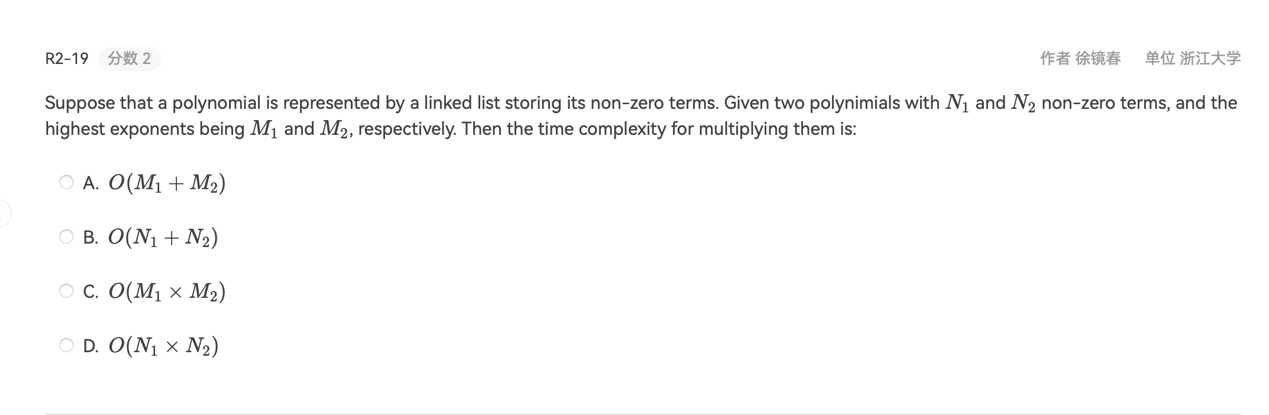
DFS如果拜访到已被拜访过的节点，说明存在环

1. B



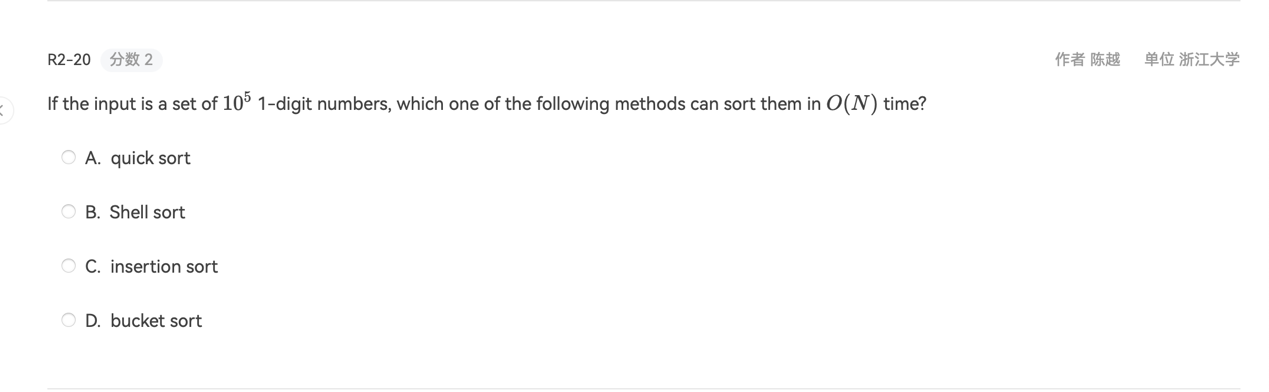
对于有n个点的无向图而言，如果边的数量不少于C(n-1, 2)+1，则一定连通，所以要让G不连通，则一定有15<C(n-1, 2)+1, 即n>=7

1. D



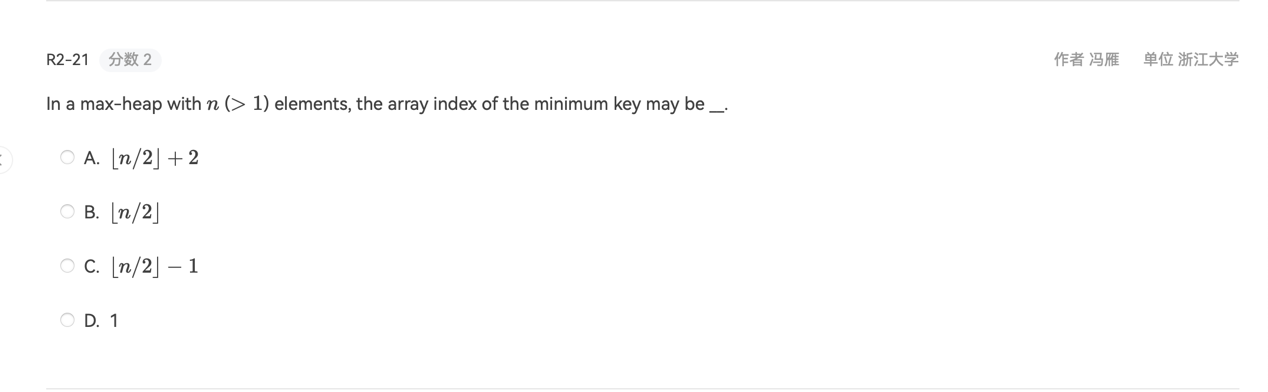
两个多项式相乘后共有N1\*N2项，所以一共要处理N1\*N2次乘法，时间复杂度即为O(N1\*N2)

1. D



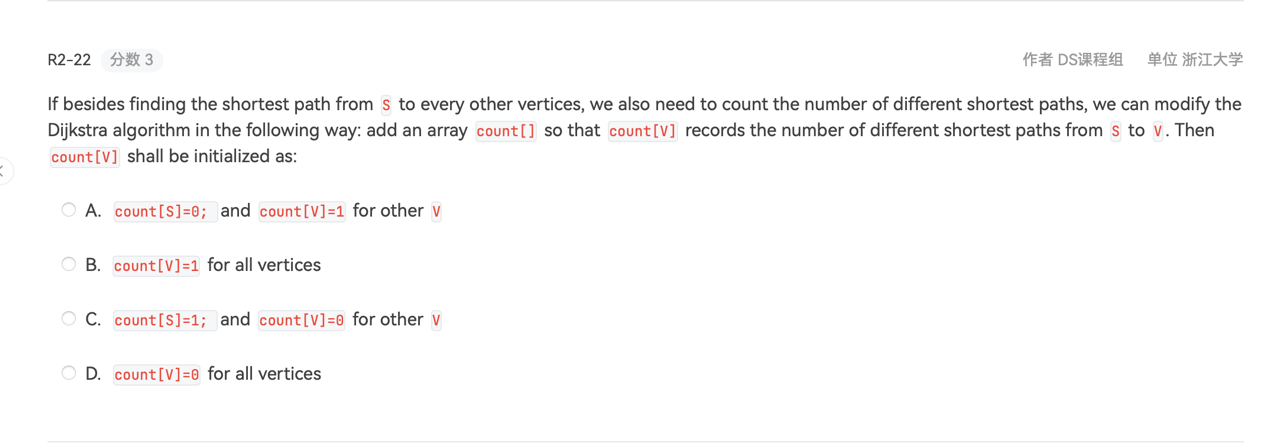
桶排序的时间复杂度是O(N)，这是一种用“空间换时间”的算法，这里的数字均只有一位数字，所以用桶排序最合适

1. A

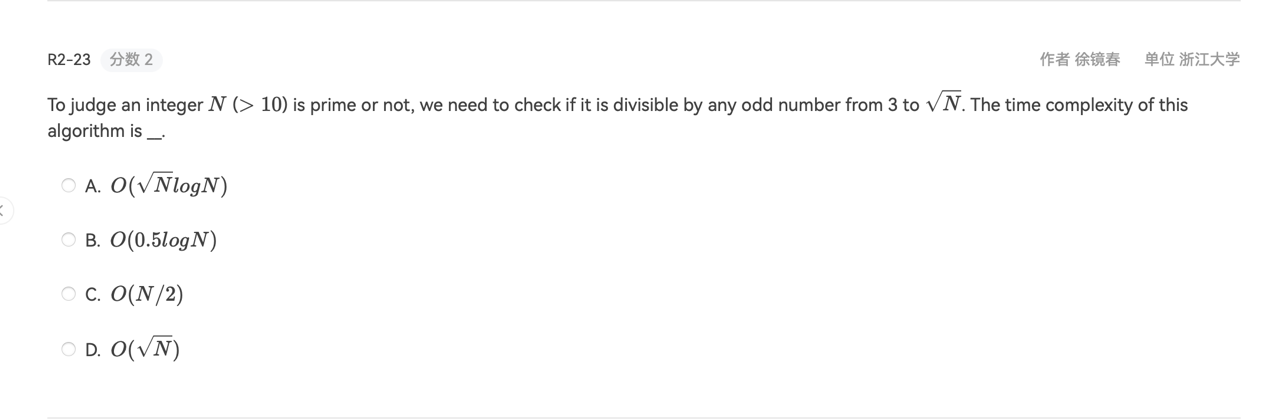


floor(n/2)是最后一个节点的父节点，在所有选项中只有A项才有可能是叶节点

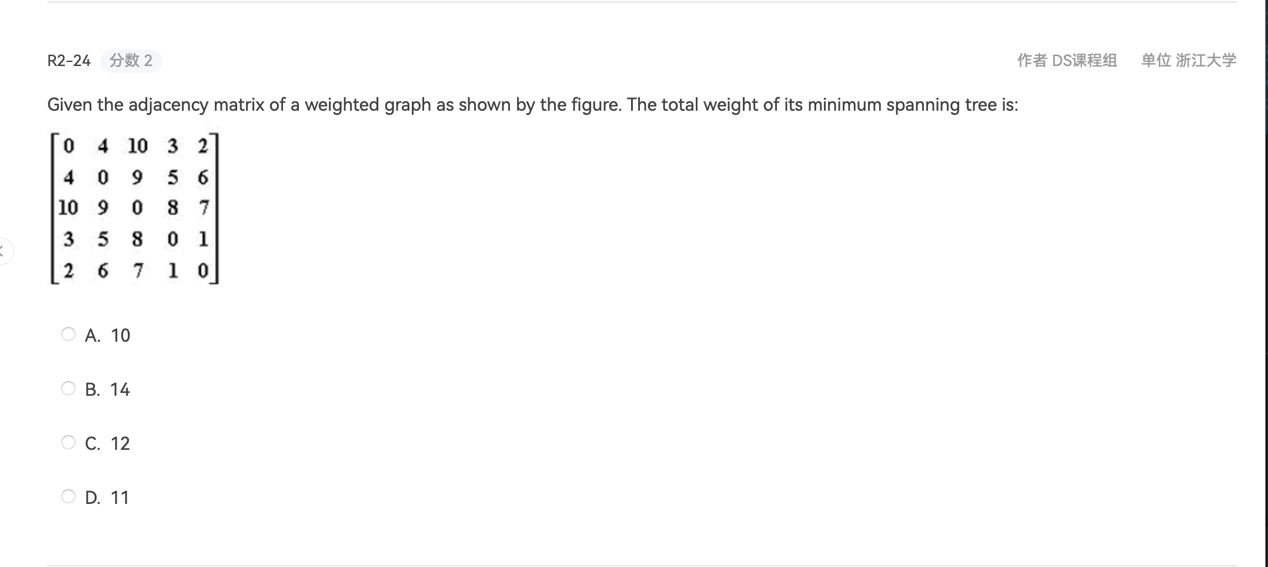
1. C

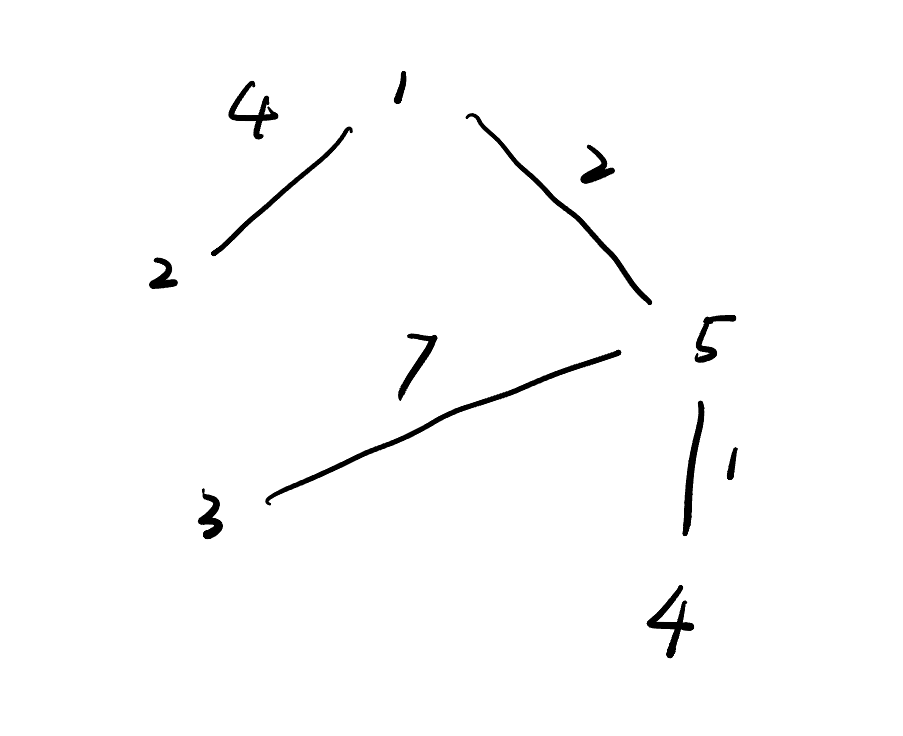


1. D



1. B





1. 程序填空题
2. 4 4 -6 4
3. diff >= -1 && diff <=1

RHeight : LHeight

1. j <= n – i + 1

r[j]->key < r[mini]->key

mini == i