

# Assignment #D: May月考

Updated 1654 GMT+8 May 8, 2024

2024 spring, Compiled by 武昱达 23工院

## 编程环境

Pycharm Windows 11

## 1. 题目

### 02808: 校门外的树

<http://cs101.openjudge.cn/practice/02808/>

思路：

简单模拟

代码

```
1 L,M=map(int,input().split())
2 arr=[1 for _ in range(L+1)]
3 for _ in range(M):
4     start,end=map(int,input().split())
5     for i in range(start,end+1):
6         if arr[i]==1:
7             arr[i]=0
8
9 cnt=0
10 for i in range(L+1):
11     if arr[i]==1:
12         cnt+=1
13 print(cnt)
```

代码运行截图 (至少包含有"Accepted")

#44897413提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
L,M=map(int,input().split())
arr=[1 for _ in range(L+1)]
for _ in range(M):
    start,end=map(int,input().split())
    for i in range(start,end+1):
        if arr[i]==1:
            arr[i]=0

cnt=0
for i in range(L+1):
    if arr[i]==1:
        cnt+=1
print(cnt)
```

基本信息

#: 44897413  
题目: E02808  
提交人: 23n2300011119 (武)  
内存: 3660kB  
时间: 48ms  
语言: Python3  
提交时间: 2024-05-08 15:14:41

## 20449: 是否被5整除

<http://cs101.openjudge.cn/practice/20449/>

思路:

简单问题。

代码

```
1 def judge(x:str):
2     num=int(x,2)
3     return num%5==0
4
5 raw=input()
6 n=len(raw)
7 arr=['0' for _ in range(n)]
8 for i in range(n):
9     if judge(raw[:i+1]):
10         arr[i]='1'
11
12 print(''.join(arr))
```

代码运行截图 (至少包含有"Accepted")

#44897625提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
def judge(x:str):
    num=int(x,2)
    return num%5==0

raw=input()
n=len(raw)
arr=['0' for _ in range(n)]
for i in range(n):
    if judge(raw[:i+1]):
        arr[i]='1'

print(''.join(arr))
```

基本信息

#: 44897625  
题目: E20449  
提交人: 23n2300011119 (武)  
内存: 3604kB  
时间: 20ms  
语言: Python3  
提交时间: 2024-05-08 15:25:57

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

## 01258: Agri-Net

<http://cs101.openjudge.cn/practice/01258/>

思路:

prim

代码

```
1 import sys
2 import heapq
3 class Vertex:
4     def __init__(self, key):
5         self.id = key
6         self.connectedTo = {}
7         self.distance = sys.maxsize
```

```

8         self.pred = None
9
10    def __str__(self):
11        return "*" + str(self.id)
12
13    def addNeighbor(self, nbr, weight=0):
14        self.connectedTo[nbr] = weight
15
16    def getConnections(self):
17        return self.connectedTo.keys()
18
19    def getWeight(self, nbr):
20        return self.connectedTo[nbr]
21
22    def __lt__(self, other):
23        return self.distance < other.distance
24
25    class Graph:
26        def __init__(self):
27            self.vertList = {}
28            self.numVertices = 0
29
30        def addVertex(self, key):
31            newVertex = Vertex(key)
32            self.vertList[key] = newVertex
33            self.numVertices += 1
34            return newVertex
35
36        def getVertex(self, n):
37            return self.vertList.get(n)
38
39        def addEdge(self, f, t, cost=0):
40            if f in self.vertList and t in self.vertList:
41                vert_f = self.vertList[f]
42                vert_t = self.vertList[t]
43                if vert_t in vert_f.connectedTo: return
44
45            if f not in self.vertList:
46                self.addVertex(f)
47            if t not in self.vertList:
48                self.addVertex(t)
49            self.vertList[f].addNeighbor(self.vertList[t], cost)
50            self.vertList[t].addNeighbor(self.vertList[f], cost)
51
52    def prim(graph, start):
53        pq = []
54        start.distance = 0
55        heapq.heappush(pq, (0, start))
56        visited = set()
57
58        while pq:
59            currentDist, currentVert = heapq.heappop(pq)
60            if currentVert in visited:
61                continue
62            visited.add(currentVert)
63

```

```

64         for nextVert in currentVert.getConnections():
65             weight = currentVert.getWeight(nextVert)
66             if nextVert not in visited and weight < nextVert.distance:
67                 nextVert.distance = weight
68                 nextVert.pred = currentVert
69                 heapq.heappush(pq, (weight, nextVert))
70
71 while True:
72     try:
73         N,g=int(input()),Graph()
74         matrix=[[i for i in map(int,input().split())] for _ in range(N)]
75         for i in range(N):
76             for j in range(i,N):
77                 if i==j:continue
78                 g.addEdge(i,j,matrix[i][j])
79         prim(g,g.getVertex(0))
80
81         res=0
82         for vert in g.vertList.values():
83             if vert.pred:
84                 res+=vert.connectedTo[vert.pred]
85         print(res)
86     except:break

```

```

1  # 另外提供一个kruskal算法，可以清晰地看到kruskal简洁很多。
2  class DisJointSet:
3      def __init__(self,num_vertices):
4          self.parent=list(range(num_vertices))
5          self.rank=[0 for _ in range(num_vertices)]
6
7      def find(self,x):
8          if self.parent[x]!=x:
9              self.parent[x] = self.find(self.parent[x])
10         return self.parent[x]
11
12     def union(self,x,y):
13         root_x=self.find(x)
14         root_y=self.find(y)
15         if root_x!=root_y:
16             if self.rank[root_x]<self.rank[root_y]:
17                 self.parent[root_x]=root_y
18             elif self.rank[root_x]>self.rank[root_y]:
19                 self.parent[root_y]=root_x
20             else:
21                 self.parent[root_x]=root_y
22                 self.rank[root_y]+=1
23
24     # graph是邻接表
25     def kruskal(graph:list):
26         res,edges,dsj=[],[],DisJointSet(len(graph))
27         for i in range(len(graph)):
28             for j in range(i+1,len(graph)):
29                 if graph[i][j]!=0:

```

```

30         edges.append((i,j,graph[i][j]))
31
32     for i in sorted(edges,key=lambda x:x[2]):
33         u,v,weight=i
34         if dsj.find(u)!=dsj.find(v):
35             dsj.union(u,v)
36             res.append((u,v,weight))
37     return res
38
39 while True:
40     try:
41         n=int(input())
42         graph=[list(map(int,input().split())) for _ in range(n)]
43         res=kruskal(graph)
44         print(sum(i[2] for i in res))
45     except EOFError:break

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44898401提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

import sys
import heapq
class Vertex:
    def __init__(self, key):
        self.id = key
        self.connectedTo = {}
        self.distance = sys.maxsize
        self.pred = None

    def __str__(self):
        return "*" + str(self.id)

    def addNeighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight

    def getConnections(self):
        return self.connectedTo.keys()

    def getWeight(self, nbr):
        return self.connectedTo[nbr]

    def __lt__(self, other):
        return self.distance < other.distance

class Graph:
    def __init__(self):
        self.vertList = {}
        self.numVertices = 0

    def addVertex(self, key):
        newVertex = Vertex(key)
        self.vertList[key] = newVertex
        self.numVertices += 1
        return newVertex

    def getVertex(self, n):
        return self.vertList.get(n)

```

基本信息

#: 44898401  
题目: M01258  
提交人: 23n2300011119 (武)  
内存: 5368kB  
时间: 47ms  
语言: Python3  
提交时间: 2024-05-08 15:55:49

## 27635: 判断无向图是否连通有无回路(同23163)

<http://cs101.openjudge.cn/practice/27635/>

思路:

is\_connected函数很好理解, 从任意一个点出发 (0), 如果最终BFS结束后的visited长度和顶点数相等, 则连通, 否则不连通。

is\_loop函数的BFS部分稍微难理解一点, 首先local\_visited是字典, 其value为走过的步长。

如果无环，那么对于任意的vert，其next\_vert不应当出现在local\_visited中，除非next\_vert是vert的前驱节点，该情况时steps[next\_vert]==steps[vert]-1。

如果有环，则必然有next\_vert(且非vert的前驱)出现在local\_visited中，其步长不大于vert的步长（与vert同时或者在vert之前被访问）。

即：

```
1 | if local_visited[next_vert]>=steps:return True
```

代码

```
1 | from collections import defaultdict, deque
2 | # graph是邻接表{1:[2,3,4]}
3 | def is_connected(graph,n):
4 |     dq=deque()
5 |     dq.append(0)
6 |     visited=set()
7 |     visited.add(0)
8 |     while dq:
9 |         cur_vert=dq.popleft()
10 |        for next_vert in graph[cur_vert]:
11 |            if next_vert not in visited:
12 |                dq.append(next_vert)
13 |                visited.add(next_vert)
14 |        return len(visited)==n
15 |
16 | def is_loop(graph):
17 |     global_visited=set()
18 |     for vertex in graph:
19 |         if vertex not in global_visited:
20 |             # 以下是一个BFS函数。
21 |             local_visited={}
22 |             dq=deque()
23 |             dq.append((vertex,0))
24 |             local_visited[vertex]=0
25 |             global_visited.add(vertex)
26 |             while dq:
27 |                 cur_vert,steps=dq.popleft()
28 |                 for next_vert in graph[cur_vert]:
29 |                     if next_vert in local_visited:
30 |                         # 关键步骤
31 |                         if local_visited[next_vert]>=steps:
32 |                             return True
33 |                     else:
34 |                         dq.append((next_vert,steps+1))
35 |                         local_visited[next_vert]=steps+1
36 |                         global_visited.add(next_vert)
37 |             return False
38 |
39 | n,m=map(int,input().split())
40 | graph=defaultdict(list)
41 | for _ in range(m):
```

```

42     a,b=map(int,input().split())
43     graph[a].append(b)
44     graph[b].append(a)
45     print('connected:yes' if is_connected(graph,n) else 'connected:no')
46     print('loop:yes' if is_loop(graph) else 'loop:no')

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```

from collections import defaultdict, deque
# graph是邻接表{1:[2,3,4]}
def is_connected(graph,n):
    dq=deque()
    dq.append(0)
    visited=set()
    visited.add(0)
    while dq:
        cur_vert=dq.popleft()
        for next_vert in graph[cur_vert]:
            if next_vert not in visited:
                dq.append(next_vert)
                visited.add(next_vert)
    return len(visited)==n

def is_loop(graph):
    global_visited=set()
    for vertex in graph:
        if vertex not in global_visited:
            # 以下是一个BFS函数。
            local_visited={}
            dq=deque()
            dq.append((vertex,0))
            local_visited[vertex]=0
            global_visited.add(vertex)
            while dq:
                cur_vert,steps=dq.popleft()
                for next_vert in graph[cur_vert]:
                    if next_vert in local_visited:
                        if local_visited[next_vert]>=steps:
                            return True
                    else:
                        dq.append((next_vert,steps+1))
                        local_visited[next_vert]=steps+1
                        global_visited.add(next_vert)
    return False

n,m=map(int,input().split())
graph=defaultdict(list)
for _ in range(m):
    a,b=map(int,input().split())
    graph[a].append(b)
    graph[b].append(a)
print('connected:yes' if is_connected(graph,n) else 'connected:no')
print('loop:yes' if is_loop(graph) else 'loop:no')

```

基本信息

#: 44958289  
 题目: 27635  
 提交人: 23n2300011119 (武)  
 内存: 3820kB  
 时间: 30ms  
 语言: Python3  
 提交时间: 2024-05-14 10:55:33

## 27947: 动态中位数

<http://cs101.openjudge.cn/practice/27947/>

思路:

维护两个堆, 一个大根堆, 一个小根堆, 保证两个堆的并是当前所有元素, 且大根堆的最大元素不大于小根堆的最小元素

也就是把数据分成两半, 其中一半严格大于另一半且其两个堆size相等或至多相差1, 则易得中位数。

代码

```

1 import heapq

```

```

2 class Medium_finder:
3     def __init__(self):
4         self.big_heap=[]
5         self.small_heap=[]
6         self.big_size=0
7         self.small_size=0
8
9     # 插入元素
10    def insert(self,val):
11        if self.big_size-self.small_size==1:
12            heapq.heappush(self.small_heap,val)
13            self.small_size+=1
14        elif self.big_size==self.small_size:
15            heapq.heappush(self.big_heap,-val)
16            self.big_size+=1
17        if not self.big_heap or not self.small_heap:return
18        while -self.big_heap[0] > self.small_heap[0]:
19            big=-heapq.heappop(self.big_heap)
20            small=heapq.heappop(self.small_heap)
21            heapq.heappush(self.big_heap,-small)
22            heapq.heappush(self.small_heap,big)
23
24    # 查找中位数
25    def findMedium(self):
26        if self.big_size==self.small_size:
27            return (self.small_heap[0]-self.big_heap[0])/2
28        if self.big_size>self.small_size:
29            return -self.big_heap[0]
30
31    for _ in range(n:=int(input())):
32        raw=list(map(int,input().split()))
33        m_finder=Medium_finder()
34        res=[]
35        for idx,val in enumerate(raw):
36            m_finder.insert(val)
37            if (idx+1)%2==1:
38                res.append(m_finder.findMedium())
39    print(len(res))
40    print(*res)

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")



状态: Accepted

源代码

```
import heapq
class Medium_finder:
    def __init__(self):
        self.big_heap=[]
        self.small_heap=[]
        self.big_size=0
        self.small_size=0

    def insert(self,val):
        if self.big_size-self.small_size==1:
            heapq.heappush(self.small_heap,val)
            self.small_size+=1
        elif self.big_size==self.small_size:
            heapq.heappush(self.big_heap,-val)
            self.big_size+=1
        if not self.big_heap or not self.small_heap: return
        while -self.big_heap[0] > self.small_heap[0]:
            big=-heapq.heappop(self.big_heap)
            small=heapq.heappop(self.small_heap)
            heapq.heappush(self.big_heap,-small)
            heapq.heappush(self.small_heap,big)

    def findMedium(self):
        if self.big_size==self.small_size:
            return (self.small_heap[0]-self.big_heap[0])/2
        if self.big_size>self.small_size:
            return -self.big_heap[0]

for _ in range(n:=int(input())):
    raw=list(map(int,input().split()))
    m_finder=Medium_finder()
    res=[]
    for idx,val in enumerate(raw):
        m_finder.insert(val)
        if (idx+1)%2==1:
            res.append(m_finder.findMedium())
    print(len(res))
    print(*res)
```

基本信息

#: 44958829  
题目: 27947  
提交人: 23n2300011119 (武)  
内存: 10816kB  
时间: 498ms  
语言: Python3  
提交时间: 2024-05-14 11:46:24

## 28190: 奶牛排队

<http://cs101.openjudge.cn/practice/28190/>

思路:

再也不想看到这个题了!!!

```
1 #
2 N,res=int(input()),0
3 hi=[int(input()) for _ in range(N)]
4 # left[i]是i左边第一个不小于他的元素的索引, right[i]是i右边第一个不大于他的元素的索引。
5 # 容易知道, 对于指定的i, 如果i作为右端点, left[i]是左端点的一个上界, 反之同理。
6 left,right=[-1 for _ in range(N)],[N for _ in range(N)]
7 stack1,stack2=[],[]
8
9 for i in range(N-1,-1,-1):
10     while stack1 and hi[stack1[-1]]>hi[i]:
11         stack1.pop()
12     if stack1:right[i]=stack1[-1]
13     stack1.append(i)
14
15 for i in range(N):
16     while stack2 and hi[stack2[-1]]<hi[i]:
17         stack2.pop()
18     if stack2:left[i]=stack2[-1]
19     stack2.append(i)
```

```
20
21 for i in range(N):
22     for j in range(right[i]-1,i,-1):
23         if left[j]<i:
24             res=max(j-i+1,res)
25             break
26
27 print(res)
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

## 2. 学习总结和收获

---

如果作业题目简单, 有否额外练习题目, 比如: OJ"2024spring每日选做"、CF、LeetCode、洛谷等网站题目。