

Assignment #F: All-Killed 满分

Updated 1844 GMT+8 May 20, 2024

2024 spring, Compiled by 武昱达

编程环境

PyCharm 2023.1.4 (Professional Edition)

1. 题目

22485: 升空的焰火，从侧面看

<http://cs101.openjudge.cn/practice/22485/>

思路：

打层数标记的BFS

代码

```
1 from collections import deque, defaultdict
2 N=int(input())
3 l=[-1 for _ in range(N+1)]
4 r=[-1 for _ in range(N+1)]
5 for i in range(1,N+1):
6     left,right=map(int,input().split())
7     l[i],r[i]=left,right
8
9 def BFS(root):
10     q,step=deque(),0
11     q.append((root,step))
12     buffer,ans=defaultdict(list),[]
13
14     while q:
15         cur,step=q.popleft()
16         buffer[step].append(cur)
17         if l[cur]!=-1:q.append((l[cur],step+1))
18         if r[cur]!=-1:q.append((r[cur],step+1))
19
20     for step in buffer:
21         ans.append(buffer[step][-1])
22     return ans
23
24 ans=BFS(1)
25 print(*ans)
```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
from collections import deque, defaultdict
N=int(input())
l=[-1 for _ in range(N+1)]
r=[-1 for _ in range(N+1)]
for i in range(1,N+1):
    left,right=map(int,input().split())
    l[i],r[i]=left,right

def BFS(root):
    q,step=deque(),0
    q.append((root,step))
    buffer,ans=defaultdict(list),[]

    while q:
        cur,step=q.popleft()
        buffer[step].append(cur)
        if l[cur]!=-1:q.append((l[cur],step+1))
        if r[cur]!=-1:q.append((r[cur],step+1))

    for step in buffer:
        ans.append(buffer[step][-1])
    return ans

ans=BFS(1)
print(*ans)
```

基本信息

#: 45027959
题目: 22485
提交人: 23n2300011119 (武)
内存: 3660kB
时间: 21ms
语言: Python3
提交时间: 2024-05-20 20:09:49

28203:【模板】单调栈

<http://cs101.openjudge.cn/practice/28203/>

思路：
单调递减栈。

代码

```
1 n,row=int(input()),[0]+list(map(int,input().split()))
2 ans,stack=[0 for _ in range(n+1)],[]
3 for i in range(n,-1,-1):
4     while stack and row[stack[-1]]<=row[i]:
5         stack.pop()
6     if stack:ans[i]=stack[-1]
7     stack.append(i)
8 print(*ans[1:])
```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
n,row=int(input()),[0]+list(map(int,input().split()))
ans,stack=[0 for _ in range(n+1)],[]
for i in range(n,-1,-1):
    while stack and row[stack[-1]]<=row[i]:
        stack.pop()
    if stack:ans[i]=stack[-1]
    stack.append(i)
print(*ans[1:])
```

基本信息

#: 45027288
题目: 28203
提交人: 23n2300011119 (武)
内存: 370160kB
时间: 2967ms
语言: Python3
提交时间: 2024-05-20 19:11:32

09202: 舰队、海域出击!

<http://cs101.openjudge.cn/practice/09202/>

思路:

Kahn算法拓扑排序

代码

```
1  from collections import deque
2  for _ in range(T:=int(input())):
3      N,M=map(int,input().split())
4
5      graph={i:[] for i in range(1,N+1)}
6      in_degree=[0 for i in range(N+1)]
7
8      for i in range(M):
9          f,t=map(int,input().split())
10         graph[f].append(t)
11         in_degree[t]+=1
12
13     q,cnt,visited=deque(),0,set()
14     for i in range(1,N+1):
15         if in_degree[i]==0:
16             q.append(i)
17     while q:
18         cur=q.popleft()
19         cnt+=1
20         for vert in graph[cur]:
21             in_degree[vert]-=1
22             if in_degree[vert]==0 and vert not in visited:
23                 visited.add(vert)
24                 q.append(vert)
25
26     print('No' if cnt==N else "Yes")
```

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 deque
for _ in range(T:=int(input())):
    N,M=map(int,input().split())

    graph={i:[] for i in range(1,N+1)}
    in_degree=[0 for i in range(N+1)]

    for i in range(M):
        f,t=map(int,input().split())
        graph[f].append(t)
        in_degree[t]+=1

    q,cnt,visited=deque(),0,set()
    for i in range(1,N+1):
        if in_degree[i]==0:
            q.append(i)
    while q:
        cur=q.popleft()
        cnt+=1
        for vert in graph[cur]:
            in_degree[vert]-=1
            if in_degree[vert]==0 and vert not in visited:
                visited.add(vert)
                q.append(vert)

    print('No' if cnt==N else "Yes")
```

基本信息

#: 45028509
题目: 09202
提交人: 23n2300011119 (武)
内存: 62348kB
时间: 3996ms
语言: Python3
提交时间: 2024-05-20 20:44:49

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

04135: 月度开销

<http://cs101.openjudge.cn/practice/04135/>

思路:

经典二分法

代码

```
1 n,m=map(int,input().split())
2 expend=[int(input()) for i in range(n)]
3 def check(x):
4     # 判断x作为最大月度开销是否可以实现, 如果可以实现, 则说明不够小或刚好符合题意。
5     # 看m个方案是否可行。
6     nums,s=1,0
7     for i in range(n):
8         if expend[i]+s>x:
9             s=expend[i]
10            nums+=1    # 求和大于设定的最大月度开销, 则应该插入挡板, 份数+1
11        else:s+=expend[i]
12    return nums>m    # if nums>m return True,else return False
13
14 lo,hi,res=max(expend),sum(expend)+1,1
15 while lo<hi:
16     mid=(lo+hi)//2
17     if check(mid):lo=mid+1
18     else:res,hi=mid,mid
19
20 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

```
n,m=map(int,input().split())
expend=[int(input()) for i in range(n)]
def check(x):
    # 判断x作为最大月度开销是否可以实现,如果可以实现,则说明不够小或刚好符合题意。
    # 看m个方案是否可行。
    nums,s=1,0
    for i in range(n):
        if expend[i]+s>x:
            s=expend[i]
            nums+=1 # 求和大于设定的最大月度开销,则应该插入挡板,份数+1
        else:s+=expend[i]
    return nums>m # if nums>m return True,else return False

lo,hi,res=max(expend),sum(expend)+1,1
while lo<hi:
    mid=(lo+hi)//2
    if check(mid):lo=mid+1
    else:res,hi=mid,mid

print(res)
```

基本信息

#: 45028717
题目: 04135
提交人: 23n2300011119 (武)
内存: 7960kB
时间: 510ms
语言: Python3
提交时间: 2024-05-20 20:58:45

©2002-2022 POJ 京ICP备20010980号-1

[English](#) [帮助](#) [关于](#)

07735: 道路

<http://cs101.openjudge.cn/practice/07735/>

思路:

- 1 # 重要剪枝: N个节点,不回头情况下至多N-1步。
- 2 if cost+nc<=K and step+1<N

代码

```
1 from heapq import heappop,heappush
2 from collections import defaultdict
3 K,N,R=int(input()),int(input()),int(input())
4 graph=defaultdict(list)
5 for i in range(R):
6     S,D,L,T=map(int,input().split())
7     graph[S].append((D,L,T))
8 def Dijkstra(graph):
9     global K,N,R
10    q,ans=[],[]
11    heappush(q,(0,0,1,0))
12    while q:
13        l,cost,cur,step=heappop(q)
14        if cur==N:return l
15        for next,nl,nc in graph[cur]:
16            if cost+nc<=K and step+1<N:
17                heappush(q,(l+nl,cost+nc,next,step+1))
18    return -1
19 print(Dijkstra(graph))
```

代码运行截图 (AC代码截图,至少包含有"Accepted")

状态: Accepted

源代码

```
from heapq import heappop, heappush
from collections import defaultdict
K, N, R = int(input()), int(input()), int(input())
graph = defaultdict(list)
for i in range(R):
    S, D, L, T = map(int, input().split())
    graph[S].append((D, L, T))
def Dijkstra(graph):
    global K, N, R
    q, ans = [], []
    heappush(q, (0, 0, 1, 0))
    while q:
        l, cost, cur, step = heappop(q)
        if cur == N: return l
        for next, nl, nc in graph[cur]:
            if cost + nc <= K and step + 1 < N:
                heappush(q, (l + nl, cost + nc, next, step + 1))
    return -1
print(Dijkstra(graph))
```

基本信息

#: 45032288
题目: 07735
提交人: 23n2300011119 (武)
内存: 6988kB
时间: 42ms
语言: Python3
提交时间: 2024-05-21 12:44:30

01182: 食物链

<http://cs101.openjudge.cn/practice/01182/>

思路:

一个并查集可以理解为带有两个虚部，非常巧妙。

代码

```
1 class DisjointSet:
2     def __init__(self, n):
3         # 设 [1, n] 区间表示同类, [n+1, 2*n] 表示 x 吃的动物, [2*n+1, 3*n] 表示吃 x 的动物。
4         self.parent = [i for i in range(3 * n + 1)] # 每个动物有三种可能的类型,
5         # 用 3 * n 来表示每种类型的并查集
6         self.rank = [0] * (3 * n + 1)
7
8     def find(self, u):
9         if self.parent[u] != u:
10             self.parent[u] = self.find(self.parent[u])
11         return self.parent[u]
12
13     def union(self, u, v):
14         pu, pv = self.find(u), self.find(v)
15         if pu == pv:
16             return False
17         if self.rank[pu] > self.rank[pv]:
18             self.parent[pv] = pu
19         elif self.rank[pu] < self.rank[pv]:
20             self.parent[pu] = pv
21         else:
22             self.parent[pv] = pu
23             self.rank[pu] += 1
24         return True
25
26 def is_valid(n, statements):
27     dsu = DisjointSet(n)
28
```

```

29     false_count = 0
30     for d, x, y in statements:
31         if x > n or y > n:
32             false_count += 1
33             continue
34
35         if d == 1: # 同类
36             # 如果x和y中有任意一个未出现过，则二者之根（未出现过的那个的自身是其根）必然
不相等，不存在冲突问题；
37             # 如果x和y都出现过，则其必然已经分属三棵树，且y+n,y+2n,x+n,x+2n也已经分属
三棵树。
38             if dsu.find(x) == dsu.find(y+n) or dsu.find(x) == dsu.find(y+2*n):
39                 # 不是同类，与条件矛盾
40                 false_count += 1
41             else:
42                 dsu.union(x,y) # x,y是同种动物
43                 dsu.union(x+n,y+n) # x,y吃同一种动物
44                 dsu.union(x+2*n,y+2*n) # x,y被同一种动物吃
45             else: # x吃y
46                 if dsu.find(x) == dsu.find(y) or dsu.find(x + 2*n) ==
dsu.find(y): # 如果x,y是同类
47                     false_count += 1
48                 else: # [1,n] 区间表示同类，[n+1,2*n]表示x吃的动物，[2*n+1,3*n]表示吃x的
动物
49                     dsu.union(x+n,y) # x吃的动物和y是同类
50                     dsu.union(x,y+2*n) # x和吃y的动物是同类
51                     dsu.union(x+2*n,y+n) # 吃x的动物和y吃的动物是同类
52
53     return false_count
54
55 if __name__ == "__main__":
56     N, K = map(int, input().split())
57     statements = []
58     for _ in range(K):
59         D, X, Y = map(int, input().split())
60         statements.append((D, X, Y))
61     result = is_valid(N,statements)
62     print(result)

```

代码运行截图 (AC代码截图，至少包含有"Accepted")

状态: Accepted

源代码

```
class DisjointSet:
    def __init__(self, n):
        # 设[1,n] 区间表示同类, [n+1,2*n]表示x吃的动物, [2*n+1,3*n]表示吃x的动物
        self.parent = [i for i in range(3 * n + 1)] # 每个动物有三种可能的关系
        self.rank = [0] * (3 * n + 1)

    def find(self, u):
        if self.parent[u] != u:
            self.parent[u] = self.find(self.parent[u])
        return self.parent[u]

    def union(self, u, v):
        pu, pv = self.find(u), self.find(v)
        if pu == pv:
            return False
        if self.rank[pu] > self.rank[pv]:
            self.parent[pv] = pu
        elif self.rank[pu] < self.rank[pv]:
            self.parent[pu] = pv
        else:
            self.parent[pv] = pu
            self.rank[pu] += 1
        return True

def is_valid(n, statements):
    dsu = DisjointSet(n)

    false_count = 0
```

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。

道路是一个很有趣的双指标BFS，但是题目输入数据有点坑；

食物链非常巧妙的并查集。

本次作业难度适中，但是需要刷题提高熟练度，迎接机考。