

Assignment #D: 图 & 散列表
Updated 2042 GMT+8 May 20, 2025
2025 spring, Compiled by 李振硕、信息管理系

1. 题目

M17975: 用二次探查法建立散列表

<http://cs101.openjudge.cn/practice/17975/>

<mark>需要用这样接收数据。因为输入数据可能分行了，不是题面描述的形式。

OJ 上面有的题目是给 C++设计的，细节考虑不周全。</mark>

```
```python
import sys
input = sys.stdin.read
data = input().split()
index = 0
n = int(data[index])
index += 1
m = int(data[index])
index += 1
num_list = [int(i) for i in data[index:index+n]]
```
```

思路:

代码:

```
```python
```

```
```
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

M01258: Agri-Net

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

思路:

代码:

状态: Accepted

源代码

```
import sys
import heapq

def read_case():
    lines = []
    while True:
        line = sys.stdin.readline()
        if not line:
            break
        if line.strip().isdigit():
            if lines:
                yield lines
                lines = []
            lines.append(line.strip())
        if lines:
            yield lines

def parse_matrix(lines):
    N = int(lines[0])
    data = ' '.join(lines[1:]).split()
    matrix = []
    idx = 0
    for i in range(N):
        row = list(map(int, data[idx:idx+N]))
        matrix.append(row)
        idx += N
    return N, matrix

def prim(N, matrix):
    visited = [False] * N
    min_cost = [float('inf')] * N
    min_cost[0] = 0
    heap = [(0, 0)] # (cost, node)
    total = 0

    while heap:
        cost, u = heapq.heappop(heap)
        if visited[u]:
            continue
        visited[u] = True
        total += cost
        for v in range(N):
            if not visited[v] and matrix[u][v] < min_cost[v]:
                min_cost[v] = matrix[u][v]
                heapq.heappush(heap, (matrix[u][v], v))
    return total

# 主处理逻辑
for case_lines in read_case():
    N, matrix = parse_matrix(case_lines)
    print(prim(N, matrix))
```

基本信息

#: 49290119
题目: 01258
提交人: 24n2300093007
内存: 4832kB
时间: 28ms
语言: Python3
提交时间: 2025-05-28 21:31:35

M3552.网络传送门旅游

bfs, <https://leetcode.cn/problems/grid-teleportation-traversal/>

思路:

代码:

题目描述通过 × 题解提交记录

全部提交记录


通过 608 / 608 个通过的测试用例
LEE 李振硕 提交于 2025.05.28 21:56

写题解

面向在校学生的专享特惠
完成认证享 7 折 Plus 会员, 享受更多学业及职业成长帮助

执行用时分布
2886 ms | 击败 82.54%
复杂度分析

消耗内存分布
58.65 MB | 击败 92.92%



</> 代码

Python3 智能模式

```
1 class Solution:
2     def minMoves(self, matrix: List[str]) -> int:
3         if matrix[-1][-1] == '#':
4             return -1
5
6         m, n = len(matrix), len(matrix[0])
7         pos = defaultdict(list)
8         for i, row in enumerate(matrix):
9             for j, c in enumerate(row):
10                 if c.isupper():
11                     pos[c].append((i, j))
12
13         DIRS = [(0, -1), (0, 1), (-1, 0), (1, 0)]
14         dis = [[inf] * n for _ in range(m)]
15         dis[0][0] = 0
16         q = deque([(0, 0)])
17
18         while q:
19             x, y = q.popleft()
20             d = dis[x][y]
21
22             if x == m - 1 and y == n - 1: # 到达终点
23                 return d
24
25             c = matrix[x][y]
26             if c in nos:
```

已存储

行 23, 列 25

M787.K 站中转内最便宜的航班

Bellman Ford, <https://leetcode.cn/problems/cheapest-flights-within-k-stops/>

思路:

代码:

题目描述通过 × 题解提交记录

全部提交记录

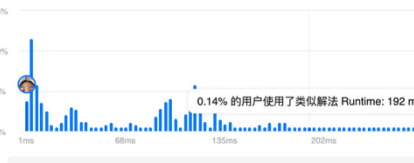
通过 56 / 56 个通过的测试用例
LEE 李振硕 提交于 2025.05.28 21:57

官方题解 写题解

面向在校学生的专享特惠
完成认证享 7 折 Plus 会员, 享受更多学业及职业成长帮助

执行用时分布
0 ms | 击败 100.00%
复杂度分析

消耗内存分布
19.13 MB | 击败 26.76%
复杂度分析



</> 代码

Python3 智能模式

```
1 from collections import defaultdict, deque
2 from typing import List
3
4 class Solution:
5     def findCheapestPrice(self, n: int, flights: List[List[int]],
6                           src: int, dst: int, k: int) -> int:
7         # 建图: 邻接表
8         graph = defaultdict(list)
9         for u, v, cost in flights:
10             graph[u].append((v, cost))
11
12         # 队列元素为: (当前城市, 当前总价格, 已用中转次数)
13         queue = deque()
14         queue.append((src, 0, 0))
15
16         # 到达每个城市的最小费用记录 (防止不必要的搜索)
17         prices = [float('inf')] * n
18         prices[src] = 0
19
20         while queue:
21             city, cost, stops = queue.popleft()
22
23             if stops > k:
24                 continue
25
26             for neighbor, price in graph[city]:
27                 new_cost = cost + price
28                 # 只有当到达 neighbor 的费用更便宜时, 才继续向外扩展
29                 if new_cost < prices[neighbor]:
30                     prices[neighbor] = new_cost
31                     queue.append((neighbor, new_cost, stops + 1))
```

已存储

行 33, 列 66

M03424: Candies

Dijkstra, <http://cs101.openjudge.cn/practice/03424/>

思路:

代码:

状态: **Accepted**

源代码

```
import heapq
from collections import defaultdict

def solve_with_dijkstra(n, m, constraints):
    graph = defaultdict(list)
    for a, b, c in constraints:
        graph[a].append((b, c)) # candies[b] <= candies[a] + c -> edge

    dist = [float('inf')] * (n + 1)
    dist[1] = 0 # snoopy (node 1) gets 0 candies

    pq = [(0, 1)] # (distance, node)
    while pq:
        d, u = heapq.heappop(pq)
        if d > dist[u]:
            continue
        for v, w in graph[u]:
            if dist[v] > dist[u] + w:
                dist[v] = dist[u] + w
                heapq.heappush(pq, (dist[v], v))

    return dist[n] - dist[1]

# 主程序输入处理
if __name__ == "__main__":
    import sys
    input = sys.stdin.read
    data = input().split()

    n, m = int(data[0]), int(data[1])
    constraints = []
    idx = 2
    for _ in range(m):
        a, b, c = int(data[idx]), int(data[idx+1]), int(data[idx+2])
        constraints.append((a, b, c))
        idx += 3

    print(solve_with_dijkstra(n, m, constraints))
```

基本信息

#: 49296646
题目: 03424
提交人: 24n2300093007
内存: 61760kB
时间: 311ms
语言: Python3
提交时间: 2025-05-29 17:49:48

M22508:最小奖金方案

topological order, <http://cs101.openjudge.cn/practice/22508/>

思路:

代码:

状态: Accepted

基本信息

#: 49296670
题目: 22508
提交人: 24n2300093007
内存: 4240kB
时间: 24ms
语言: Python3
提交时间: 2025-05-29 17:51:20

源代码

```
from collections import defaultdict, deque

def min_total_award(n, m, matches):
    # 构建反向图 (败方 -> 胜方)
    graph = defaultdict(list)
    indegree = [0] * n
    for a, b in matches:
        graph[b].append(a) # b -> a 表示 a 的奖金必须比 b 高
        indegree[a] += 1

    # 初始化拓扑排序队列
    queue = deque()
    dist = [0] * n # 奖金增量部分 (最终是 100 + dist[i])
    for i in range(n):
        if indegree[i] == 0:
            queue.append(i)

    # 拓扑排序 + DP 计算每个节点的最长路径长度
    while queue:
        u = queue.popleft()
        for v in graph[u]:
            if dist[v] < dist[u] + 1:
                dist[v] = dist[u] + 1
            indegree[v] -= 1
            if indegree[v] == 0:
                queue.append(v)

    total_award = sum(100 + d for d in dist)
    return total_award

# 主函数处理输入
if __name__ == "__main__":
    import sys
    input = sys.stdin.read
    data = input().split()

    n = int(data[0])
    m = int(data[1])
    matches = []
    idx = 2
    for _ in range(m):
        a = int(data[idx])
        b = int(data[idx + 1])
        matches.append((a, b))
        idx += 2

    print(min_total_award(n, m, matches))
```

2. 学习总结和收获

<mark> “如果发现作业题目相对简单，有否寻找额外的练习题目，如 数算 2025spring ”每日选做 、

这次作业对我来说是最难的，用二次探查法建立散列表这一道题想了很久，但现在还没 AC，想以后再补。Dijkstra, topological order 还没学好。机考前一定要复习完。