Assignment #C: bfs & dp

Updated 1436 GMT+8 Nov 25, 2025

2025 fall, Complied by 郑文萱 基础医学院

说明：

1）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

2）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

3）如果不能在截止前提交作业，请写明原因。

1. 题目

sy321迷宫最短路径

bfs, https://sunnywhy.com/sfbj/8/2/321

思路： 求最短路径了一定是bfs，感觉对于迷宫类的搜索都比较模板化。但是这个题要求写出路径，这个最好是找出最短后返回去走，参考了题解，记录每个前置点最后读取（有点像链表的那种查找方式），这个思维非常有用。

代码：

```
from collections import deque
n,m=map(int,input().split())
puzzle=[]
for _ in range(n):
    puzzle.append(list(map(int,input().split())))
queue=deque()
queue.append((0,0))
prev_point=[[None]*m for _ in range(n)]
prev_point[0][0]=(-1,-1)
directions=[(0,1),(0,-1),(1,0),(-1,0)]
while queue:
    x,y=queue.popleft()
    if x==n-1 and y==m-1:
        break
    for dx,dy in directions:
        nx,ny=x+dx,y+dy
        if 0<=nx<n and 0<=ny<m and puzzle[nx][ny]==0 and prev_point[nx][ny]==None:
            prev_point[nx][ny]=(x,y)
            queue.append((nx,ny))
back=[]
x,y=n-1,m-1
```

```
while (x,y)!=(-1,-1):
    back.append((x,y))
    x,y=prev_point[x][y]
back.reverse()
for (x,y) in back:
    print(x+1,y+1)
```

代码运行截图 （至少包含有"Accepted"）

代码书写 Python

```
from collections import deque
n,m=map(int,input().split())
puzzle=[]
for _ in range(n):
    puzzle.append(list(map(int,input().split())))
queue=deque()
queue.append((0,0))
prev_point=[[None]*m for _ in range(n)]
prev_point[0][0]=(-1,-1)
directions=[(0,1),(0,-1),(1,0),(-1,0)]
while queue:
    x,y=queue.popleft()
    if x==n-1 and y==m-1:
        break
    for dx,dy in directions:
        nx,ny=x+dx,y+dy
        if 0<=nx<n and 0<=ny<m and puzzle[nx][ny]==0 and
            prev_point[nx][ny]=(x,y)
            queue.append((nx,ny))
back=[]
x,y=n-1,m-1
while (x,y)!=(-1,-1):
```

测试输入 提交结果 历史提交

完美通过 查看题解

100% 数据通过测试 详情

运行时长: 0 ms

sy324多终点迷宫问题

bfs, https://sunnywhy.com/sfbj/8/2/324

思路：这个题就跟上个题内核是一样的，只不过不用往回走但是要记录步数，可以说是更经典的BFS，写一个函数然后对迷宫内的每个点应用就可以，不写函数更好因为避免了多次从头调用超时的问题，只要把步数记下来就行。（于是直接对上一题代码进行了修改）

代码：

```
from collections import deque
n,m=map(int,input().split())
puzzle=[]
for _ in range(n):
    puzzle.append(list(map(int,input().split())))
queue=deque()
queue.append((0,0))
count=[[-1]*m for _ in range(n)]
count[0][0]=0
directions=[(0,1),(0,-1),(1,0),(-1,0)]
while queue:
    x,y=queue.popleft()
    for dx,dy in directions:
        nx,ny=x+dx,y+dy
        if 0<=nx<n and 0<=ny<m and puzzle[nx][ny]==0 and count[nx][ny]==-1:
            count[nx][ny]=count[x][y]+1
            queue.append((nx,ny))
for i in range(n):
    output=' '.join(str(count[i][j]) for j in range(m))
    print(output)
```

代码运行截图 （至少包含有"Accepted"）

代码书写 Python

```python
from collections import deque
n,m=map(int,input().split())
puzzle=[]
for _ in range(n):
    puzzle.append(list(map(int,input().split())))
queue=deque()
queue.append((0,0))
count=[[-1]*m for _ in range(n)]
count[0][0]=0
directions=[(0,1),(0,-1),(1,0),(-1,0)]
while queue:
    x,y=queue.popleft()
    for dx,dy in directions:
        nx,ny=x+dx,y+dy
        if 0<=nx<n and 0<=ny<m and puzzle[nx][ny]==0 and
            count[nx][ny]=count[x][y]+1
            queue.append((nx,ny))
for i in range(n):
    output=' '.join(str(count[i][j]) for j in range(m))
    print(output)
```

测试输入 提交结果 历史提交

完美通过 查看题解

100% 数据通过测试 详情

运行时长: 0 ms

M02945: 拦截导弹

dp, greedy http://cs101.openjudge.cn/pctbook/M02945

思路：一看到此题时想到的思路是把从每一个开始拦截然后能够拦截的个数都算一下然后取最大值，不过在算个数的时候就需要dp。状态转移方程的思路是每个都取前面比他高或与其相等的高度最大值加一，最后的输出比较有趣，不是特定值而是最大值，这在《算法笔记》里面亦有记载。

代码：

```
n=int(input())
nums=list(map(int,input().split()))
dp=[1]*n
for i in range(n):
    for j in range(i):
        if nums[j]>=nums[i]:
            dp[i]=max(dp[i],dp[j]+1)
print(max(dp))
```

代码运行截图 （至少包含有"Accepted"）

状态：Accepted

源代码

```
n=int(input())
nums=list(map(int,input().split()))
dp=[1]*n
for i in range(n):
    for j in range(i):
        if nums[j]>=nums[i]:
            dp[i]=max(dp[i],dp[j]+1)
print(max(dp))
```


189A. Cut Ribbon

brute force/dp, 1300, https://codeforces.com/problemset/problem/189/A

思路： 标签既然是蛮力或dp那肯定还是首选dp,感觉是比较像硬币找零的，dp思路是三个小单位单独判断然后取最大的。

代码：

```
n,a,b,c=map(int,input().split())
dp=[-1]*(n+1)
dp[0]=0
for i in range(1,n+1):
    if i>=a and dp[i-a]!=-1:
        dp[i]=max(dp[i],dp[i-a]+1)
    if i>=b and dp[i-b]!=-1:
        dp[i]=max(dp[i],dp[i-b]+1)
    if i>=c and dp[i-c]!=-1:
        dp[i]=max(dp[i],dp[i-c]+1)
print(dp[n])
```

代码运行截图 （至少包含有"Accepted"）

351565705	Dec/02/2025 16:16UTC+8	zwx0208	189A - Cut Ribbon	PyPy 3	Accepted	93 ms	1400 KB

M01384: Piggy-Bank

dp, http://cs101.openjudge.cn/practice/01384/

思路：跟上一题很像，目标变为求总量最小值，状态转移方程的思路是从每种硬币出发，这个我不太能想到于是参考题解，发现是用每种硬币更新dp最小值的，学到了学到了。用python3提交超时了，换了pypy3才过了。

代码：

```
t=int(input())
for _ in range(t):
    e,f=map(int,input().split())
    total_weight=f-e
    n=int(input())
    coins=[]
    for _ in range(n):
        p,w=map(int,input().split())
        coins.append((p,w))
    dp=[float('inf')]*(total_weight+1)
    dp[0]=0
    for p,w in coins:
        for weight in range(w,total_weight+1):
            if dp[weight-w]!=float('inf'):
                dp[weight]=min(dp[weight-w]+p,dp[weight])
    if dp[total_weight]==float('inf'):
        print('This is impossible.')
    else:
        print(f'The minimum amount of money in the piggy-bank is
{dp[total_weight]}.')
```

代码运行截图 （至少包含有"Accepted"）

#51101640提交状态

状态: Accepted

源代码

```
t=int(input())
for _ in range(t):
    e,f=map(int,input().split())
    total_weight=f-e
    n=int(input())
    coins=[]
    for _ in range(n):
        p,w=map(int,input().split())
        coins.append((p,w))
    dp=[float('inf')]*(total_weight+1)
    dp[0]=0
    for p,w in coins:
        for weight in range(w,total_weight+1):
            if dp[weight-w]!=float('inf'):
                dp[weight]=min(dp[weight-w]+p,dp[weight])
    if dp[total_weight]==float('inf'):
        print('This is impossible.')
    else:
        print(f'The minimum amount of money in the piggy-bank is {dp[total_weight]}
```

M02766: 最大子矩阵

dp, kadane, http://cs101.openjudge.cn/pctbook/M02766

思路：kadane这个名字听起来挺高级的但是很好理解啊，储存前一个数据，这种方法在刚学dp做斐波那契数列的时候就已经在用了。此题不太会，参考题解，其中二维降为一维数组的解法太妙了，先固定行后在列的维度进行kadane算法，就变成一维了，学到了学到了（这个列表推导式也是非常好的）。但其实还是没有太掌握这种方法，准备多找几个kadane的题练练。

代码：

```
def max_matrix(matrix):
    def kadane(l):
        max_current=max_global=l[0]
        for x in l[1:]:
            max_current= max(x, max_current + x)
            max_global= max(max_current, max_global)
        return max_global
    rows = len(matrix)
    cols = len(matrix[0])
    max_sum = float('-inf')
```

```
    for left in range(cols):
        temp = [0] * rows
        for right in range(left, cols):
            for row in range(rows):
                temp[row] += matrix[row][right]
            max_sum = max(max_sum, kadane(temp))
    return max_sum
n = int(input())
nums = []
while len(nums)<n**2:
    nums.extend(input().split())
matrix = [list(map(int,nums[i*n:(i+1)*n])) for i in range(n)]
max_sum = max_matrix(matrix)
print(max_sum)
```

代码运行截图 （至少包含有"Accepted"）

状态：Accepted

源代码

```
def max_matrix(matrix):
    def kadane(l):
        max_current=max_global=l[0]
        for x in l[1:]:
            max_current= max(x, max_current + x)
            max_global= max(max_current, max_global)
        return max_global
    rows = len(matrix)
    cols = len(matrix[0])
    max_sum = float('-inf')
    for left in range(cols):
        temp = [0] * rows
        for right in range(left, cols):
            for row in range(rows):
                temp[row] += matrix[row][right]
            max_sum = max(max_sum, kadane(temp))
    return max_sum
n = int(input())
nums = []
while len(nums)<n**2:
    nums.extend(input().split())
matrix = [list(map(int,nums[i*n:(i+1)*n])) for i in range(n)]
max_sum = max_matrix(matrix)
print(max_sum)
```


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“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。 助教上机做了前三个题，还是会做的，迷宫属于bfs比较典型的，浇水双指针可以说是最简单的一题，解密就是递归就可以过，预测赢家看了但没有想到好的dp思路，神经网络之国和最后一题几乎没看，感觉也不太

能会了，准备把重心放在M题上。作业的话前两个bfs比较常规，课下在晴问也做了几个练手，345dp属于之前没太见过的，学到很多，最后一题好难只能先复刻题解，准备给自己一点时间再消化消化。上次作业立的 flag 竟然完成了，就是看完了《算法图解》，感觉这本书用于理解算法思维还是很有用的，而且图片很多（字大行稀）所以不知不觉就能看完。