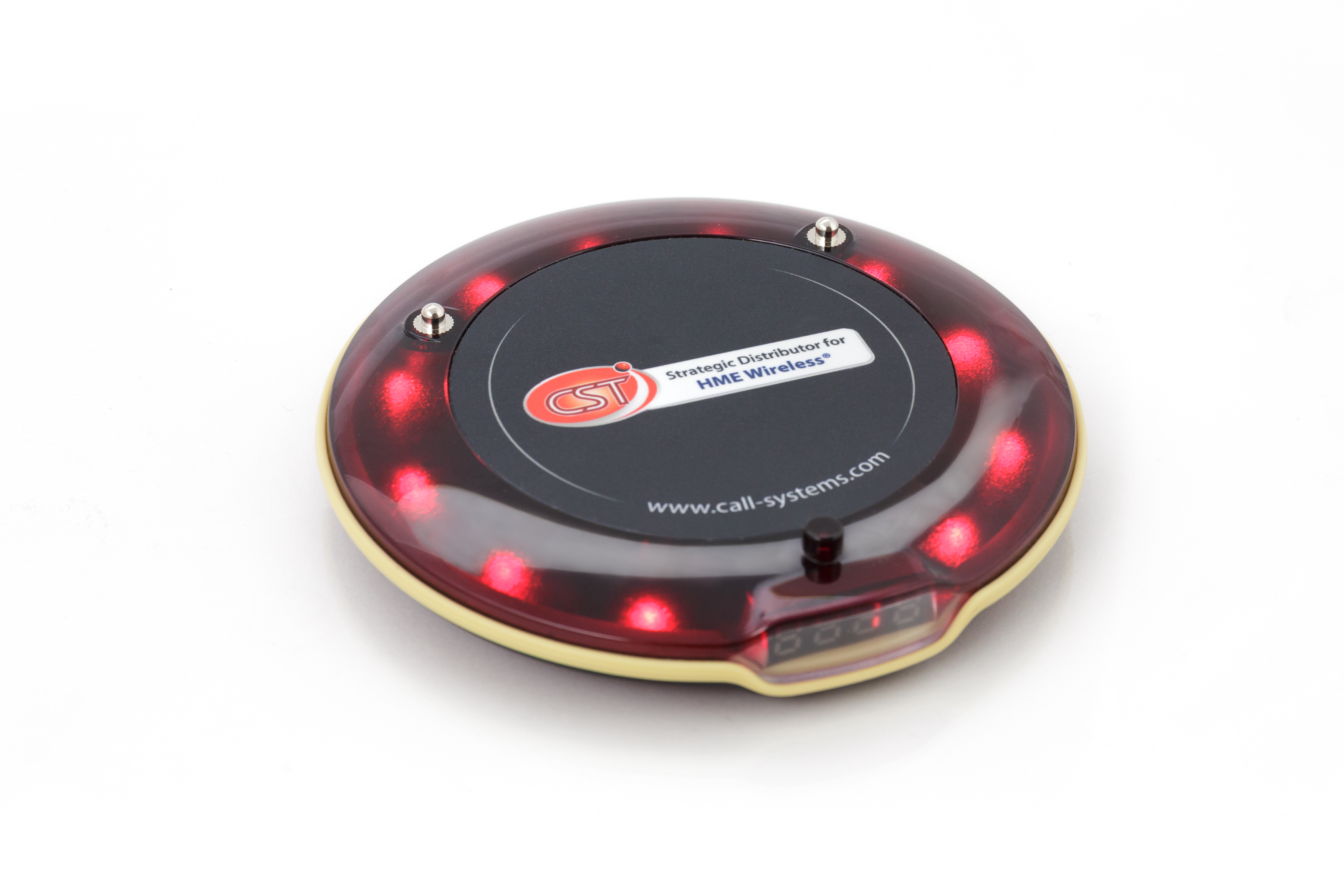
|  |
| --- |
| **Editorial พีทซิลล่าสั่งอาหาร (PZ\_Pager)** |

*ที่มา: ข้อสอบท้ายค่ายสองศูนย์ ม.บูรพา รุ่น 17*

**Problem Statement**

ระบบเพจเจอร์ เรียกคิวเป็นระบบที่เริ่มใช้กันอย่างแพร่หลาย ในกลุ่มงานที่ไม่มีการใช้งานเฉพาะด้าน เช่นร้านอาหาร

หรือสถานที่ที่มีการใช้คิวธรรมดา ไม่มากนัก โดยเน้นไปในการช่วยอำนวยความสะดวก และ ลดปัญหาจากระบบคิวเดิมๆ เช่นการเรียกชื่อ ที่อาจจะไม่ได้ยินกรณีที่ผู้รับบริการอยู่ในสถานที่ ที่มีเสียงรบกวน และลบจุดด้อยในค่าใช้จ่ายจากอุปกรณ์สิ้นเปลืองต่างๆ ซึ่งแตกต่างกับระบบคิวแบบทั่วไป โดยระบบคิวแบบเพจเจอร์เป็นระบบคิวที่สามารถเตือนผู้ที่รอคิวได้ทันทีเมื่อถึงคิว ทำให้ระบบคิว แบบเพจเจอร์ เป็นอีกหนึ่งระบบที่กำลังได้รับความรับนิยมในหมู่งานระบบโรงพยาบาล และ ระบบร้านอาหาร



วันนี้พีทซิลล่าก็ได้เป็นเจ้าของร้านอาหารชื่อดังแห่งหนึ่งในบียูยูแลนด์ ได้นัดหมายกับลูกค้าจำนวน N คนไว้โดยแต่ละคนอยากสั่งอาหารที่เวลา L หน่วยและได้รับอาหารที่เวลา R หน่วยที่จำเพาะสำหรับแต่ละคน แต่เนื่องจากช่วงนี้เป็นช่วงโควิด ลูกค้าแต่ละคนไม่สามารถถือเพจเจอร์ไว้กับตัวเองได้ พีทซิลล่าเลยจำเป็นต้องกองเพจเจอร์ของลูกค้าแต่ละคนไว้เป็นกองซ้อน หรือ Stack โดยจะหยิบเข้ากองทันทีที่ลูกค้าเริ่มสั่งอาหาร แล้วตั้งเวลาไว้ว่า ณ เวลา R หน่วยเครื่องจะดังทันที เมื่อเครื่องดังพีทซิลล่าก็นำเพจเจอร์ออกจากกองซ้อนและนำอาหารไปให้พนักงานเพื่อส่งของทันที แต่ถ้าเครื่องเพจเจอร์ที่ไม่ใช่เครื่องที่อยู่บนสุดดังจะทำให้

พีทซิลล่าไม่สามารถนำเครื่องเพจเจอร์ออกจากกองซ้อนได้ ทำให้การสั่งอาหารนั้นเป็นโมฆะทันที

**ตัวอย่าง** เช่นมีลูกค้าจำนวน 4 คนอยากได้อาหารดังตาราง

|  |  |  |  |
| --- | --- | --- | --- |
| **ลูกค้าคนที่** | **เวลาสั่งอาหาร** | **เวลาส่งอาหาร** | **ราคา** |
| 1 | 1 | 10 | 1,000 |
| 2 | 2 | 5 | 200 |
| 3 | 3 | 7 | 2,000 |
| 4 | 5 | 9 | 500 |

เมื่อพิจารณาการสั่งอาหารของลูกค้าแต่ละคนพบว่าลูกค้าคนที่ 1, 2 และ 4 สามารถสั่งอาหารพร้อมกันได้ ดังนี้

ที่เวลา 1 หน่วย : พีทซิลล่านำเพจเจอร์ของลูกค้าคนที่ 1 ใส่ไปในกอง

ที่เวลา 2 หน่วย : พีทซิลล่านำเพจเจอร์ของลูกค้าคนที่ 2 ใส่ไปในกอง

ที่เวลา 5 หน่วย : พีทซิลล่านำเพจเจอร์ของลูกค้าคนที่ 2 ออกจากกอง พร้อมทั้งนำเพจเจอร์ของลูกค้าคนที่ 4 เข้าไปในกอง ( กล่าวคือ ถ้ามีคำสั่งนำเพจเจอร์เข้าและออกในหน่วยเวลาเดียวกัน จะทำการดึงเพจเจอร์ออกให้หมดก่อน แล้วนำเพจเจอร์เข้าในเวลาเสี้ยวหน่วยจนแทบไม่ได้ใช้เวลา )

ที่เวลา 9 หน่วย : พีทซิลล่านำเพจเจอร์ของลูกค้าคนที่ 4 ออกจากกอง

ที่เวลา 10 หน่วย : พีทซิลล่านำเพจเจอร์ของลูกค้าคนที่ 1 ออกจากกอง

แต่พีทซิลล่าไม่สามารถรับรายการสั่งอาหารของลูกค้าคนที่ 2 และ 3 พร้อมกันได้เนื่องจากที่เวลา 5 หน่วย เพจเจอร์ของลูกค้าคนที่ 2 ซึ่งอยู่ด้านล่างของซ้อนดังและต้องนำออก ซึ่งไม่ใช่เพจเจอร์ที่อยู่ด้านบน ทำให้การสั่งอาหารของลูกค้าคนที่ 2 และ 3 เป็นโมฆะ

แต่อย่างไรก็ตามพีทซิลล่าอยากรับรายการสั่งอาหารของลูกค้าที่ทำให้พีทซิลล่าได้เงินมากที่สุด ดังนั้นพีทซิลล่าจึงต้องรับรายการอาหารเฉพาะลูกค้าคนที่ 1 และ 3 ที่ได้เงินรวมจำนวน 3,000 บาท

**งานของคุณ**

จงเขียนโปรแกรมเพื่อช่วยหาว่าสามารถรับรายการอาหารอย่างดีที่สุดแล้วได้เงินมากสุดเท่าใด

**ข้อมูลนำเข้า**

บรรทัดแรก รับจำนวนเต็มบวก 1 จำนวน คือ Q แทนจำนวนคำถาม โดยที่ 1 <= Q <= 3

ในแต่ละคำถาม

บรรทัดแรก รับจำนวนเต็มบวก 1 จำนวน คือ N แทนจำนวนลูกค้า โดยที่ 1 <= N <= 200

อีก N บรรทัดต่อมา รับจำนวนเต็มบวก 3 จำนวนคือ L, R และ C โดยที่ 1 <= L < R

และ C เป็นราคาของอาหารที่ลูกค้าสั่ง มีค่าไม่เกินขอบเขตของตัวแปร int โดยไม่มีลูกค้าสองคนใดๆ ที่สั่งอาหารและรับอาหารที่เวลาเดียวกันพอดี

**ข้อมูลส่งออก**

มี Q บรรทัด โดยแต่ละบรรทัดให้แสดงจำนวนเงินที่มากสุดที่พีทซิลล่าได้จากการรับรายการอาหาร

**ตัวอย่างที่ 1**

|  |  |
| --- | --- |
| **ข้อมูลนำเข้า** | **ข้อมูลส่งออก** |
| 2  4  1 10 1000  2 5 200  3 7 2000  5 9 500  3  10 12 200  10 15 400  13 17 400 | 3000  600 |

**คำอธิบายตัวอย่างที่ 1**

ในคำถามที่ 1 เลือกลูกค้าคนที่ 1 และ 3 ทำให้ได้เงินจำนวนมากที่สุด

ในคำถามที่ 2 เลือกลูกค้าคนที่ 1 และ 2 หรือ ลูกค้าคนที่ 1 และ 3 ก็ได้

**แนวคิด**

แต่ละปัญหาย่อยจะมีวิธีทำต่างกันดังนี้

**ปัญหาย่อยที่ 1** จะมี N ไม่เกิน 15

ในปัญหาย่อยนี้เราจะทำการเรียงสับเปลี่ยน ( permutation ) ในการเลือกลูกค้าในทุกแบบเพื่อหาแบบที่ดีที่สุดที่เป็นไปได้แล้วจำลองการนำเพจเจอร์เข้ากองซ้อนและออกกองซ้อนตามโจทย์โดยใช้ **std::priority\_queue** ที่ใช้ในการเรียงเวลาการนำเพจเจอร์เข้าออก



**ปัญหาย่อยที่ 2** จะมี N ไม่เกิน 200

เราจะใช้ dynamic programming ในการแก้ปัญหา โดยเราจะนิยาม  ว่ารายการอาหารที่อยู่ในช่วงเวลาตั้งแต่ L ถึง R นั้นสามารถทำจำนวนเงินได้มากสุดเท่าใด จะได้ว่า 

โดย  แทนราคาของรายการอาหารที่ตรงกับช่วง พอดีโดยถ้าไม่มีรายการอาหารนั้นจะให้ 

แต่เนื่องจาก 1 <= L, R ทำให้ไม่สามารถใช้ index เป็นเวลาที่ให้ในโจทย์ โดยเราจะทำการ hash เวลาที่โจทย์กำหนดให้เป็นจำนวนเต็ม  โดย  เป็นจุดเวลาที่แตกต่างกันทั้งหมด ทำให้ต้องใช้หน่วยความจำเพียง 

โดยเรามีฟังก์ชันในการหาค่าของ  ดังนี้ เมื่อตอนก่อนเรียกฟังก์ชันเราจะให้ทุกช่องของ 

**long long finddp(long long L,long long R){**

**if(dp[L][R]!=-1) return dp[L][R];**

**long long mx=0,K;**

**for(K=L+1;K<=R-1;K++){**

**mx = max(mx,finddp(L,K)+finddp(K,R));**

**}**

**return dp[L][R]= mx+cost[L][R];**

**}**

 ในการหาค่า  โดยเราจะตอบค่าของ 

**Full Source Code**

|  |
| --- |
| **#include<bits/stdc++.h>**  **using namespace std;**  **#define N 205**  **long long l[N],r[N],c[N],dp[2\*N][2\*N],cost[2\*N][2\*N],cnt;**  **unordered\_map<long long,long long> mp;**  **vector<long long> g;**  **long long finddp(long long L,long long R){**  **if(dp[L][R]!=-1) return dp[L][R];**  **if(L+1>=R) return dp[L][R]=cost[L][R];**  **long long mx=0,K;**  **for(K=L+1;K<=R-1;K++){**  **mx = max(mx,finddp(L,K)+finddp(K,R));**  **}**  **return dp[L][R]= mx+cost[L][R];**  **}**  **int main(){**  **long long q,n,i;**  **scanf("%lld",&q);**  **while(q--){**  **memset(cost,0,sizeof cost);**  **memset(dp,-1,sizeof dp);**  **mp.clear();**  **g.clear();**  **scanf("%lld",&n);**  **for(i=0;i<n;i++){**  **scanf("%lld %lld %lld",&l[i],&r[i],&c[i]);**  **if(mp[l[i]]!=1) mp[l[i]]=1,g.push\_back(l[i]);**  **if(mp[r[i]]!=1) mp[r[i]]=1,g.push\_back(r[i]);**  **}**  **sort(g.begin(),g.end());**  **cnt=0;**  **for(auto x : g){**  **mp[x]=cnt++;**  **}**  **for(i=0;i<n;i++){**  **cost[mp[l[i]]][mp[r[i]]]=c[i];**  **}**  **printf("%lld\n",finddp(0,cnt-1));**  **}**  **return 0;**  **}** |

