Link to Working Application: https://d3mhat8u0brqys.cloudfront.net/

Link to Code Repository: https://github.com/omchs10/Hotel-Booking

Note: Delay of 1 sec has been added in app to make it more user friendly, delay is not because of algorithm.

Candidate Details:

Name: Omprakash Gupta

Email: omchs10@gmail.com

Contact: +91 7355655802

Solutions:

As per given assessment, I will use below algorithm to calculate **Room Proximity or Travel Time** when all required Rooms are not available on same floor:

Let required number of Rooms are N and Travel Time is T and N possible available rooms set is ListOfRooms, where structure of item of ListOfRooms is { number: 201, booked: false}

Than

IF N<2 that is N==1 Then Travel Time T = 0 { Ex: Travelling from Room Number 101 to 101 will take 0 min }

ELSE LOOP through ListOfRooms that is i = 1 to i < ListOfRooms.length and T =0

Let current room for which Travel Time is being calculated is **currRoom,** previous room from which travelled to current room is **prevRoom,** current floor where currRoom present is **currFloor** and previous floor from which we are travelling to current floor is **prevFloor**

THEN

currRoom = ListOfRooms[i]

prevRoom = ListOfRooms[i-1]

AND

prevFloor = Math.floor(prevRoom.number / 100) { as Room number is 101-1007}

currFloor = Math.floor(currRoom.number / 100)

Let **prevRoomNumber** is room count of previous room number **prevRoom** and **currRoomNumber** is room count of current room number **currRoom,**

THEN

const prevRoomNumber = prevRoom.number % 100 { let prevRoom is 203 then room count will be 3}

const currRoomNumber = currRoom.number % 100

NOW

IF prevFloor == currFloor that is same Floor **THEN**

T+= absolute of (currRoomNumber - prevRoomNumber)*1 { same floor travelling}

ELSE

T += absolute of (currFloor - prevFloor) * 2 { Time taken to travel different floor}

T+= absolute of (currRoomNumber + prevRoomNumber) – 1 {Time taken to travel different rooms on different floor}

RETURN (T) {Here T is required travel time taken for given set of Rooms}

Time Complexity: O(N)

Now we have Travel time T for given set of rooms, I will use this algorithm to find Travel Time between Rooms in below algorithm to find best set of required Rooms.

Algorithm to find best set of Required N rooms so that Travel Time T is minimum:

STEP:1 (Time Complexity: O(No of Floor * N))

If N rooms are available on same floor, priorities these set of Rooms

Algorithm:

I will use logic of Sliding Window of Window size N to find best set of Rooms.

LOOP through Each Floor that is f = 1 to f = 10,

For each floor f, find best combination of rooms **bestRoomsOnFloor** and minimum travel time **minTravelTimeOnFloor,**

SORT available rooms on floor f, let sorted list is roomsOnFloor

LOOP from i = 0 to i = roomsOnFloor.length – N

let travelTime = roomsOnFloor [i + N - 1].number - roomsOnFloor [i].number

IF travelTime < minTravelTimeOnFloor **THEN**

minTravelTimeOnFloor = travelTime **AND**

bestRoomsOnFloor = Combination of Rooms from roomsOnFloor[i] to roomsOnFloor[i+N]

After execution of all loops, return **bestRoomsOnFloor** as result that is

RETURN bestRoomsOnFloor

STEP:2 (Time Complexity: O(No of available rooms^2 * N))

If N rooms are not available on same floor but available across the floors, I will use dynamic programming to find minimum travel time between the rooms and hence best set of N rooms,

Pseudocode of implementation of logic (rooms is all rooms from 101-1007 and numRooms is number of rooms required that is N):

```
function findBestRooms(rooms, numRooms):
  availableRooms = getAvailableRooms(rooms)
```

```
if length(availableRooms) < numRooms:
   return null

dp = initializeDPTable(length(availableRooms), numRooms)
path = initializePathTable(length(availableRooms), numRooms)

dp[0][0] = 0

for i from 1 to length(availableRooms):
   for j from 1 to numRooms:
      for k from 0 to i-1:
         travelTime = calculateTravelTime(availableRooms[k:i])
         if dp[k][j-1] + travelTime < dp[i][j]:
            dp[i][j] = dp[k][j-1] + travelTime
            path[i][j] = k

minTravelTime = Infinity
bestEnd = -1

for i from 1 to length(availableRooms):
   if dp[i][numRooms] < minTravelTime:
      minTravelTime = dp[i][numRooms]
      bestEnd = i

bestCombination = []
current = bestEnd
```

```
remainingRooms = numRooms

while remainingRooms > 0:
    start = path[current][remainingRooms]
    bestCombination.prepend(availableRooms[start:current])
    current = start
    remainingRooms -= 1

return bestCombination
```

Assessments for SDE 3

As part of the recruitment process, you are required to complete the assessment mentioned below. A few things to note while attempting the assessments.

- You may upload a google doc with the relevant links (working app link, link to your code repository and a google doc with the solution)
- Please ensure that the privacy of the relevant google docs/files is set to “Anyone with the link”
- In case you have any questions, you may reach out to us at careers@unstop.com
- The deadline to complete the assessment is 3 days from the date of registration. In case you need an extension please drop us an email at careers@unstop.com
- Name your file as: YourName_AssessmentSubmission.
- Please avoid sharing your assessment over email and submit only on the link shared via mail https://unstop.com/jobs/software-development-engineer-unstop-942370

Important Notes:

- Verify that all links are accessible before submitting.
- Late submissions will not be considered.

Assignment : Hotel Room Reservation System

Problem Statement:

A hotel has **97 rooms** distributed across **10 floors**:

- **Floors 1-9:** Each floor has **10 rooms**, numbered sequentially (e.g., Floor 1: 101-110, Floor 2: 201-210, and so on).
- **Floor 10 (Top Floor):** Has only **7 rooms**, numbered **1001-1007**.
- **Building Structure:**
    1. A staircase and lift are located on the **left side** of the building.
    2. Rooms on each floor are arranged sequentially from **left to right**, with the **first room** on each floor being closest to the stairs/lift.
- **Room Proximity (Travel Time):**
    1. **Horizontal travel:** Moving between two adjacent rooms on the same floor takes **1 minute per room**.
    2. **Vertical travel:** Moving between floors takes **2 minutes per floor** using the stairs/lift.
- **Booking Rules:**
    1. A single guest can book up to **5 rooms** at a time.
    2. Priority is to book rooms on the same floor first.
    3. If rooms are not available on the same floor, Priority is to book rooms that minimize the **total travel time** between the first and last room in the booking.
    4. If the required number of rooms is unavailable on one floor, booking should span across floors, prioritizing rooms that minimize the combined vertical and horizontal travel time.

Example Scenario:

1. **Available Rooms:**
    - Floor 1: 101, 102, 105, 106
    - Floor 2: 201, 202, 203, 210
    - Floor 3: 301, 302
2. A guest wants to book 4 rooms:
    - Rooms **101, 102, 105, 106** on Floor 1 will be selected because they minimize total travel time.
3. If only **2 rooms** are available on Floor 1 (e.g., 101, 102):
    - The system will select rooms **201, 202** from Floor 2, as this minimizes vertical (2 minutes) and horizontal travel times.

Your task is to create a room reservation system that dynamically calculates the total travel time between booked rooms and optimally assigns rooms based on these rules.

Deliverables:

Submit a live url with following functionality:

- Interface to enter no of rooms and book them.

- Visualization of booking

- Button to generate random occupancy on rooms,

- Button to reset entire booking


No of Rooms
Book
Reset
Random

OMPRAKASH GUPTA

omchs10@gmail.com | 7355655802 | Noida 201301

Summary

Talented professional brings proven abilities in end-to-end project management, specializing in Amazon Connect and React. Strong understanding of domain-specific requirements and Very quick learner. Highly organized leader with great attention to detail, adept at working in hybrid environments. Committed to continuous learning and staying updated with the latest industry trends and technologies. I am having expertise in AWS services, Chatbot development and Fullstack development.

Experience

06/2021 - Current
Cognizant Technology Solutions India Pvt Ltd
Noida

Associate Projects

•Developed 10+ Solutions to automate the gaps present in AWS platform and in Contact center using Gen AI and Chatbot tools like Amazon connect, Genesys and Kore AI
Skills: Cloud, AWS, GCP, Reactsjs, Javascript, Nodejs, Java, Kore AI, Product development, Agile Methodology, chatbot, NLP, Generative AI, CI/CD, Git, Contact center tools, Amazon Lex

•Developed GenAI based PowerBI analytics tool to generate PowerBI visuals for user prompt.
Skills: Reactjs, Javascript, AWS Lambda, Bedrock, ML, GenAI, AWS Services, HTML5

Skills

- Language: Javascript, C++, Java, HTML5
- Framework and Library: Reactsjs, Nodejs, Springboot, Maven
- Tools: Kore AI, Git, Amazon Connect, VS code
- Cloud: AWS, GCP, Genesys, Salesforce
- Client communications
- Project Management
- Leadership skills
- Agile Development
- Project Planning
- Collaborative mindset
- Software Development
- Project development
- Contact Center
- GenAI
- Machine Learning

Education and Training

2017-2021

BTech in Computer Science and Engineering
National Institute of Technology Delhi

Certifications

- Reactjs by Udemy
- Javascript
- AWS Cloud Practitioner by AWS (957c29e015b440a7818a9ee85411556)
- Java

Languages

English, Hindi