EMERGENCY RESPONSE RESOURCE ALLOCATION IN HOSPITALS USING BIPARTITE GRAPH MATCHING

Team Details:

Name	Reg.no.
V.Sreenivasulu	99220041409
S.Praneeth gowd	99220041376
T.Santhosh kumar reddy	99220041391
K.Ajith Kumar Y.Gopi krishna	99220041231 99220041596

Introduction:

- Emergency response resource allocation in hospitals is a critical problem that involves efficiently assigning limited resources, such as doctors, nurses, and medical equipment, to incoming patients, especially in situations of high demand like pandemics or natural disasters.
- Hospitals often have limited numbers of doctors, nurses, medical equipment (like ventilators), ICU beds, and other critical resources.
- In all the hospitals there will be having resources but they don’t know how to allocate them in the emergency issues.
- Time will be taken more to allocate them to the patients. At that time we can use the Bipartite graph matching by using some algorithms.
- Hospitals will be facing resource exceeds supply, in such cases decisions are used to allocate resources fairly and effectively.

- Emergency situations require quick decisions. Normally hospitals use real time information system to track resource availability.
- Computer simulation models are used to test different resource allocation strategies based on different scenarios.
- Computer simulation models:
- Time based analysis
- Weather forecasting
- When the hospitals have limited resources by using some strategies and algorithms we can get the better allocation in emergency situations.

Abstract:

- We will be using the Bipartite graph with the **Hopcroft-Karp algorithm with BFS** and **DFS** which is useful for better matching by using augmenting paths. By pairing available staff with emergency cases efficiently, this approach enhances response times and overall effectiveness in managing urgent medical situations.

Objectives:

•**Minimize Response Time**: Reduce the time it takes to allocate resources to patients, leading to faster emergency response and treatment.
•**Improve Hospital Resource Utilization**: Maximize the efficient use of hospital resources by ensuring that all available resources are utilized optimally during emergency scenarios.
•**Increase Patient Care Efficiency**: Ensure that patients receive the necessary resources in a timely manner, improving overall care quality in emergencies

s.no.	Paper title	Author	Abstract	Published year	Resource
1.	Predicting Emergency Medical Service Demand with Bipartite Graph Convolutional Networks	Xin Liu	Emergency Medical Services (EMS) are critical for providing timely care during life-threatening situations, but imbalances in supply and demand can lead to resource shortages and delays. This study models EMS demand using a hospital-region bipartite graph based on demographic, socioeconomic, and hospital data. We propose a bipartite graph convolutional neural network to predict EMS demand between hospital and region pairs in Tokyo.	June 5, 2020	IEEE
2.	Fair Allocation of Vaccines, Ventilators and Antiviral Treatments: Leaving No Ethical Value Behind in Healthcare Rationing	Ruth Faden, Nancy Kass, and colleagues	Traditional priority systems for allocating limited life-saving resources, such as vaccines and ventilators, often fail to address ethical concerns and may disadvantage vulnerable communities. The COVID-19 pandemic highlighted these issues. This paper proposes a reserve system as an alternative to traditional priority rules. By focusing on modifying restrictive features of existing systems, our approach aims to improve fairness and equity. We present a general theory of reserve design and discuss how this new method could better address allocation challenges in future emergencies, offering potential improvements in policy and practice.	20 Sep 2023	InformsPubs online

s.no.	Paper title	Author	Abstract	Published year	Resource
3.	Optimal resource allocation response to a smallpox outbreak	David M. Hyman and Daniel J. McGlynn	This paper presents a model for distributing limited vaccines across multiple cities during a smallpox outbreak to minimize fatalities. The model uses mixed integer programming to decide on vaccine allocation and control measures (isolation, ring, or mass vaccination). An efficient heuristic is developed to handle large-scale problems. Results show that this approach can save more lives compared to simple prorated allocation strategies.	2007	Elsevier
4.	Resource allocation for healthcare organisations	Viswanadham and Balaji	This study presents a two-phase method for allocating resources to elective surgeries in hospitals. It first distributes resources to all surgical requests and then uses competitive bidding to optimize allocation. The model addresses the challenge of efficiently assigning operating rooms, nurses, and equipment to maximize revenue and operational efficiency.	2014	Research gate

s.no.	Paper title	Author	Abstract	Published year	Resource
5.	Resource allocation in an emergency medical service system using computer simulation	J Emerg	This study uses a computer simulation to analyze Taipei City's Emergency Medical Service (EMS) system, aiming to improve ambulance utilization and reduce costs. The simulation, based on data from December 2000, shows that reducing the number of ambulances to one per response unit increases utilization from 8.78% to 15.47%. Despite the reduction in ambulances, service quality remains stable, with only slight increases in patient wait times. The findings suggest that EMS can operate more efficiently and cost-effectively without compromising care.	Nov, 2022	PubMed
6.	Allocation of emergency medical resources for epidemic diseases considering the areas	Bin Liu	This paper proposes an optimization model for allocating emergency medical resources during epidemics, focusing on the differing needs of various regions. By considering the heterogeneity of epidemic areas, the model ensures more effective and efficient resource distribution, improving overall epidemic response	2023	PubMed

Existing algorithm:

- **Greedy Algorithm for Resource Allocation**
- **Description:** This algorithm makes a series of decisions based on immediate availability without considering future consequences. It selects the next best available resource (doctor, room, equipment) for an appointment.
- **Implementation:**
  - *Maintain lists of available doctors and resources.*
- **Priority-Based Scheduling Algorithm**
- **Description:** This algorithm assigns priorities to patients based on the urgency of their medical needs (e.g., emergencies vs. routine check-ups). The system schedules appointments for higher-priority patients first.
- **Implementation:**
  - *Create a priority queue to manage incoming appointment requests.*

Proposing

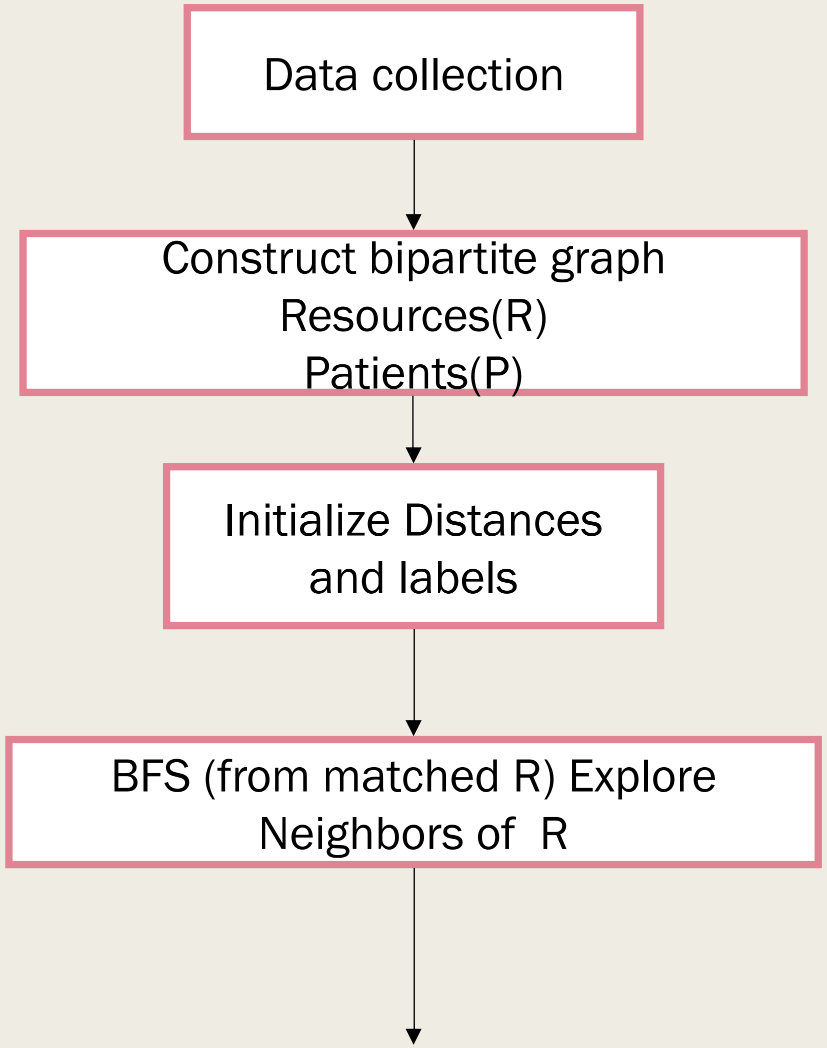
- **Bipartite Graph Matching**

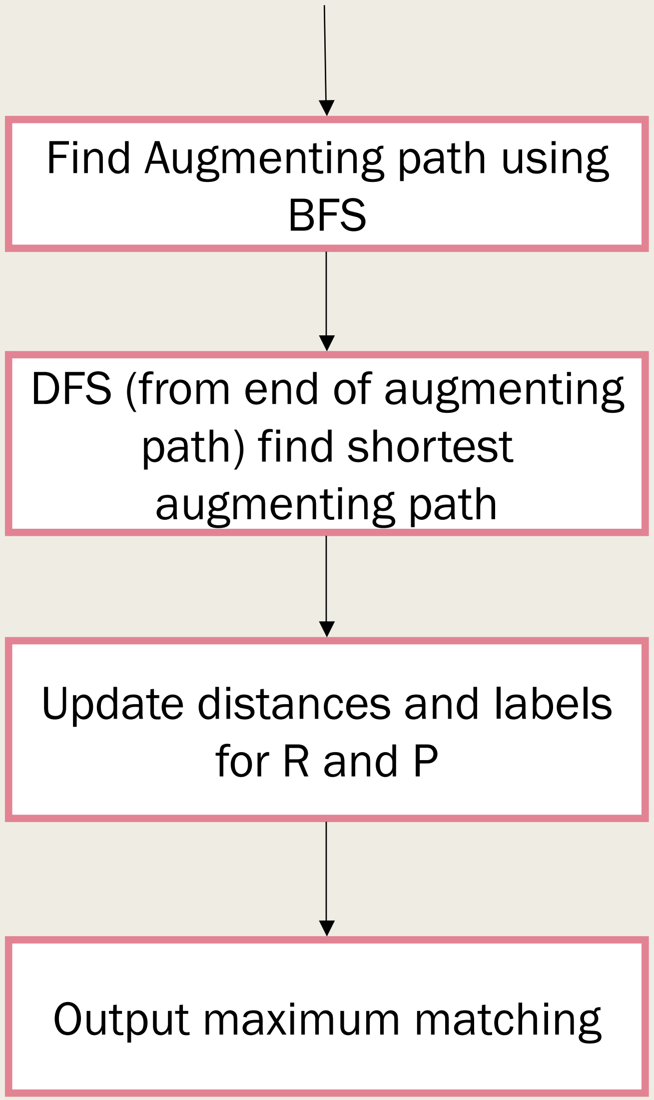
- **Description**: This is used to find the maximum matching in a bipartite graph, which can represent the relationships between resources (e.g., doctors, equipment) and patients.
- **Use Case:** Efficiently assigns staff to patients or equipment to procedures based on availability and requirements.
- **Implementation**: Algorithms like the Hopcroft-Karp algorithm can be used to find maximum matchings.
- **Algorithm:**Hopcroft-Karp   with BFS and DFS for best matching.

Why Hopcraft algorithm :

- **Time Complexity:**
- Greedy Algorithm: O(n^2 * m)
- Priority-Based Scheduling: O(n log n)
- Hopcroft-Carp Algorithm (BFS/DFS approach): O(n + m)
- **Space complexity:**
- Greedy Algorithm : O(n + m)
- Priority-Based Scheduling Algorithm: O(n)
- Hopcroft-Carp Algorithm (BFS/DFS approach): O(n + m)
- If time complexity is the primary concern, the Hopcroft-Carp Algorithm (BFS/DFS approach) is the best choice.
- If space complexity is the primary concern, all three algorithms have similar space complexity

Flow chart:


Data collection
Construct bipartite graph
Resources(R)
Patients(P)
Initialize Distances
and labels
BFS (from matched R) Explore
Neighbors of R


Find Augmenting path using BFS
DFS (from end of augmenting path) find shortest augmenting path
Update distances and labels for R and P
Output maximum matching

Methodology:

- **Modeling the Problem Using Bipartite Graphs**
- **Inputs:**
- A set of resources ‘R’
- A set of emergency cases ‘E’
- Create a bipartite graph G=(U∪V)
- Where : U represents the set of resources R.
- V represents the set of emergency cases E.
- **Formulate the Matching Problem**
- Use a suitable matching algorithm to find the optimal assignment of resources to emergency cases.
- The **Hopcroft-Karp algorithm** improves on finding the maximum matching

BFS (Breadth first search) – to find the no. of inputs

DFS(Depth first search) -- to find augment paths

- An **augmenting path** is a special kind of path in a bipartite graph that helps you find a bigger set of matching pairs between two sets .
- **Evaluate the Solution**
- Implement the chosen algorithm to compute the optimal matching.
- Based on effectiveness and efficiency we will be getting the accuracy of the solution.
- **Deployment and Monitoring**
- Implementing the algorithm and we will be monitoring the performance.

Result:

```python
if __name__ == "__main__":
    # Consider 10 resources (e.g., doctors, nurses, ventilators, etc.)
    # and 10 patients in need of these resources
    num_resources = 10
    num_patients = 10

    # Sample connections (possible allocations between resources and patients)
    possible_allocations = [
        (0, 0), (0, 1), (1, 2), (1, 3), (2, 3), (2, 4),
        (3, 5), (3, 6), (4, 7), (4, 8), (5, 9), (5, 1),
        (6, 2), (6, 3), (7, 4), (7, 5), (8, 6), (8, 7),
        (9, 8), (9, 9)
    ]

    # Allocate resources
    allocate_resources_to_patients(num_resources, num_patients, possible_allocations)
```

```
Maximum number of resources allocated: 10
Resource 0 allocated to Patient 0
Resource 1 allocated to Patient 2
Resource 2 allocated to Patient 4
Resource 3 allocated to Patient 6
Resource 4 allocated to Patient 8
Resource 5 allocated to Patient 1
Resource 6 allocated to Patient 3
Resource 7 allocated to Patient 5
Resource 8 allocated to Patient 7
Resource 9 allocated to Patient 9
```

```python
# Add availability of doctors for patients
hospital.add_availability(0, 0)  # Doctor 0 is available for Patient 0
hospital.add_availability(0, 1)  # Doctor 0 is available for Patient 1
hospital.add_availability(1, 2)  # Doctor 1 is available for Patient 2
hospital.add_availability(2, 3)  # Doctor 2 is available for Patient 3
hospital.add_availability(2, 4)  # Doctor 2 is available for Patient 4

# Allocate patients to available doctors
allocated_patients = hospital.allocate_patients()
print("Allocated Patients:")
for patient, doctor in allocated_patients:
    print(f"Patient {patient} allocated to Doctor {doctor}")

# Perform BFS traversal to demonstrate graph structure
print("\nBFS Traversal:")
hospital.bfs(0)
```

```
BFS Traversal:
0 5 2 3 1 4
DFS Traversal:
0 5 2 3 1 4 Allocated Patients:
Patient 0 allocated to Doctor 0
Patient 1 allocated to Doctor 0
Patient 2 allocated to Doctor 1
Patient 3 allocated to Doctor 2
Patient 4 allocated to Doctor 2
```

```python
class BipartiteGraph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[] for _ in range(vertices)]

    def add_edge(self, u, v):
        self.graph[u].append(v)
        self.graph[v].append(u)

    def bfs(self, start_vertex):
        visited = [False] * self.V
        queue = []

        queue.append(start_vertex)
        visited[start_vertex] = True

        while queue:
            vertex = queue.pop(0)
            print(vertex, end=" ")

            for neighbor in self.graph[vertex]:
                if not visited[neighbor]:
                    queue.append(neighbor)
                    visited[neighbor] = True

    def dfs(self, start_vertex):
        visited = [False] * self.V

        self._dfs_helper(start_vertex, visited)

    def _dfs_helper(self, vertex, visited):
        visited[vertex] = True
```

```python
        print(vertex, end=" ")

        for neighbor in self.graph[vertex]:
            if not visited[neighbor]:
                self._dfs_helper(neighbor, visited)
# Define the bipartite graph with patients and doctors
bipartite_graph = BipartiteGraph(10)

# Add edges between patients and doctors
bipartite_graph.add_edge(0, 5)  # Patient 0 is assigned to Doctor 5
bipartite_graph.add_edge(1, 3)  # Patient 1 is assigned to Doctor 3
bipartite_graph.add_edge(2, 5)  # Patient 2 is assigned to Doctor 5
bipartite_graph.add_edge(3, 2)  # Patient 3 is assigned to Doctor 2
bipartite_graph.add_edge(4, 1)  # Patient 4 is assigned to Doctor 1

# Perform BFS traversal starting from Patient 0
print("BFS Traversal:")
bipartite_graph.bfs(0)

# Perform DFS traversal starting from Patient 0
print("\nDFS Traversal:")
bipartite_graph.dfs(0)
class Hospital:
    def __init__(self, num_patients, num_doctors):
        self.patients = [i for i in range(num_patients)]
        self.doctors = [i for i in range(num_doctors)]
        self.graph = [[] for _ in range(num_patients + num_doctors)]

    def add_availability(self, doctor, patient):
        self.graph[patient].append(doctor + len(self.patients))
        self.graph[doctor + len(self.patients)].append(patient)

    def allocate_patients(self):
```

Conclusion:

- Emergency Response Resource Allocation in Hospitals Using Bipartite Graph Matching" offers a practical solution to efficiently manage hospital resources during critical situations.
- By optimizing resource allocation through advanced algorithms, the system can reduce response times and improve patient care.
- As it evolves, it could play a significant role in enhancing healthcare systems worldwide, especially during large-scale emergencies or global health crises.

Future scope:

- This project could expand to include multiple hospitals, allowing better coordination of resources during emergencies.
- It could also use real-time data, like the number of incoming patients and available hospital resources, to make quicker decisions.
- The system could predict emergency situations and automatically allocate resources based on past trends. It could also be connected to ambulance services for smoother patient transfers.

References:

- IEEE papers
- Google scholar
- Get hub
- Pub Med research papers