Optimizing Chemical Storage Utilizing the Welch-Powell Algorithm

Naomi Risaka Sitorus – 13523122[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523122@std.stei.itb.ac.id, naomi.risaka@gmail.com

***Abstract*—Nowadays, chemicals are used in many fields of human activity. Each chemical has its own properties that has to be considered when storing them to avoid unwanted incidents, such as chemical explosions. Factors to consider when storing chemicals include their incompatibilities and storage temperature. Allocating storage for chemicals can be challenging, especially when dealing with a large number of chemicals and limited available space. This study focuses on the application of the Welch-Powell graph coloring algorithm for optimizing chemical storage. The chemicals and their relationships are represented as a graph, with chemicals as vertices and incompatibilities as edges. The developed program processes the graph in adjacency matrix form, where adjacency indicates incompatibility between chemicals. The program allocates different storage units, represented colors in graph coloring theory, ensuring only chemicals that are compatible and have the same storage temperature are placed in the same unit. This study provides valuable insights into optimizing chemical storage by applying the Welch-Powell algorithm, contributing to a safer and more efficient chemical storage practices that can be applied in similar allocation systems.**

***Keywords*—Chemical Storage Optimization, Graph Theory, Graph Coloring, Welch-Powell Algorithm.**

I. Introduction

In today's world, chemicals are widely used across many fields, from industrial production to academic research. Each chemical has its own unique properties, with some being stable, whereas others are potentially hazardous when stored improperly or within close range to incompatible substances. Accidents like chemical explosions highlights the importance of safe chemical storage practices. Proper chemical storage is not only important for safety reasons, but also substance integrity and space efficiency, ensuring that the available space is used effectively while complying with the regulatory standards.

Many factors are to be taken into consideration when storing chemicals, including incompatibilities, temperature requirements, and other specific handling conditions. Organizing chemicals systematically can be especially challenging when the number of chemicals involved are high, yet the storage space is limited. This issue is particularly significant in environments such as warehouses and laboratories, where safety and efficiency are essential.

This paper delves deeper on the application of the Welch-Powell algorithm, a technique used for graph coloring, in optimizing chemical storage. By representing chemicals and their incompatibilities as a graph, the algorithm can assign chemicals to their appropriate storage containers while considering their specific requirements. The insights from this study can be used to improve chemical storage strategies, enhancing safety and space efficiency. The methods discussed in this paper can also be applied in other fields that requires proper storage management of hazardous materials.

II. Theoretical Basis

A. Graph

A graph is a mathematical structure used to represent discrete elements and the relationships between them. It is composed of vertices or nodes, typically denoted as circles, and edges, which are the lines connecting the vertices.

$$G = (V, E)$$
$$V = \{ v_1, v_2, \dots, v_n \}$$
$$E = \{ e_1, e_2, \dots, e_n \}$$

Fig. 2.1 Definition of A Graph. (Source: [1])

For a structure to be considered a graph, there must be at least one vertex in it. An empty graph or a null graph is defined as a graph without any edges, where the vertices are isolated from each other as in not connected and having no edges between them.

Fig. 2.2 An Empty Graph. (Source: [2])

In a graph, a loop is an edge that connects a vertex to itself. Additionally, two vertices can be connected by more than one edge, these edges are referred to as multiple edges or parallel edges. Based on the presence of loops or multiple edges, graphs can be classified into two main categories: simple graphs and unsimple graphs.

1. Simple graphs

   A simple graph is a graph that does not contain loops or multiple edges between the same pair of vertices.

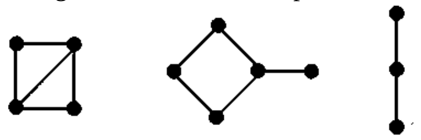


Fig. 2.3 Simple Graphs. (Source: [1])

2. Unsimple graphs

   An unsimple graph is a graph that contains either loops or multiple edges. This type of graph is further divided into two types: multi-graph, a graph containing multiple edges, and pseudo-graph, a graph containing at least one loop.

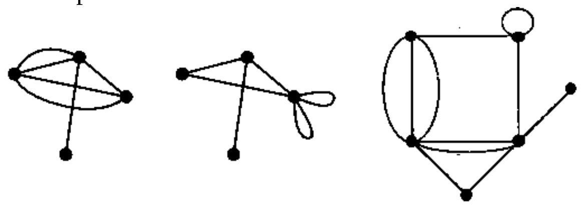


Fig. 2.4 Unsimple Graphs. (Source: [1])

Graphs can also be categorized into two types based on the direction of the edges. An undirected graph is a graph whose edges do not have any direction. On the other hand, a directed graph is a graph whose edges have a specific direction in connecting the vertices.

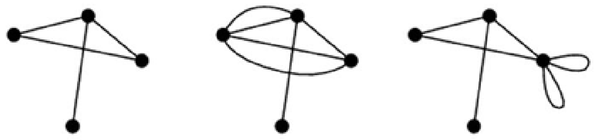


Fig. 2.5 Undirected Graphs. (Source: [1])

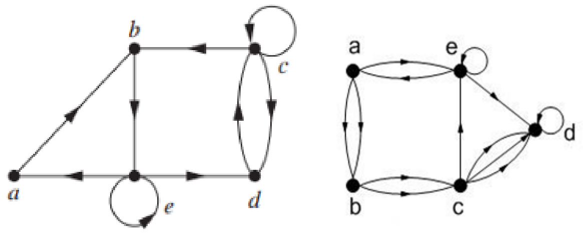


Fig. 2.6 Directed Graphs. (Source: [1])

In graph theory, several terminologies are commonly used to describe the relationships and properties of graphs.

1. Adjacency

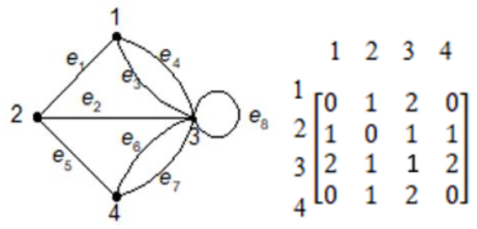
   Two vertices are considered adjacent if there is at least one edge connecting them directly.
2. Incidence

   An edge is said to be incident to an edge if the edge is connected to the said vertex.
3. Degree

   The degree of a vertex refers to the number of edges that are incident to it. In other words, it indicates the number of direct connections a vertex has.
4. Connectivity

   A graph is a connected graph if all its vertices are connected to one another, either directly or indirectly, meaning there is no isolated vertices.

An adjacency matrix is a way to represent a graph based on the adjacency relationship between vertices. The size of the matrix is determined by the number of vertices in the graph. In an undirected graph, the matrix element is set to zero if the corresponding vertices are not connected to one another. If two vertices are connected, the matrix element reflects the number of edges connecting them. For a loop, the matrix element for it is counted as two.



$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 2 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 2 & 1 & 1 & 2 \\ 4 & 0 & 1 & 2 & 0 \end{array}$$

Fig. 2.7 Adjacency Matrix of A Graph. (Source: [3])

B. Graph Coloring

Graph coloring is the process of assigning colors to the vertices of a graph such that no two adjacent vertices share the same color. The number of colors used in the coloring, referred to as the chromatic number, has to be at a minimum. The chromatic number of graph G is denoted as χ(G).

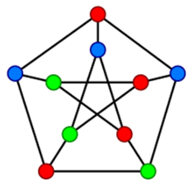


Fig. 2.8 Graph Coloring. (Source: [4])

Certain special types of graphs have predetermined chromatic numbers due to their unique characteristics.

1. Empty graphs

   The chromatic number of an empty graph is one, as none of the vertices are adjacent due to the absence of edges.
2. Complete graphs

   A complete graph is a simple graph where each vertex is connected to every other vertices. The chromatic number of a complete graph with *n* vertices is *n*, since all the vertices are adjacent to one another.
3. Circle graphs

   A circle graph is a simple graph where each vertex has a degree of two. The chromatic number of a circle graph is three if it has an odd number of vertices, and two if the number of vertices is even.
4. Bipartite graphs

   A bipartite graph is a graph with vertices that can be split into two subsets V_1 and V_2, so each edge of the graph connects a vertex in V_1 to a vertex in V_2. The chromatic number of a bipartite graph is two, with one color assigned to the vertices in V_1 and the other to those in V_2.

C. Welch-Powell Algorithm

The Welch-Powell algorithm is used to determine the chromatic number for graphs that do not fall into the categories described previously. The graph coloring process using the algorithm begins by sorting all the vertices in descending order based on their degrees. The first color is assigned to the first vertex and to the remaining vertices that are not adjacent to the already assigned ones. The process is repeated until all vertices are assigned a color, ensuring no two adjacent vertices have the same color. The chromatic number is determined by the amount of colors used in the process.

D. Chemical Storage

A chemical is a substance that has a defined composition. It can either occur naturally or be created through man-made processes. Chemicals can be found in three state of matter: solid, liquid, and gas. It can also can be categorized as pure substances or mixtures.

Each chemical has its own unique properties, which are the characteristics of a substance that can be observed or measured during or after undergoing a chemical reaction. Chemicals are often classified based on their properties, such as flammability, oxidizing potential, corrosivity, reactivity, and toxicity.

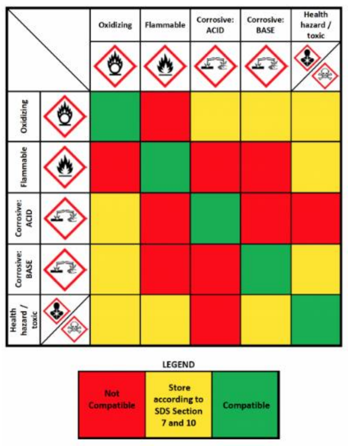
Flammable chemicals are substances that easily ignites through friction, absorption of moisture, or heat, and has the potential to cause explosions. Examples include gasoline, alcohol, acetone, toluene, and acetic acid. Oxidizers are chemicals that promote combustion by yielding combustible materials or supporting the ignition of other materials, such as oxygen and halogens like fluorine and chlorine.

Corrosive chemicals are substances capable of causing destruction or irreversible damage to other materials, including living tissues. They can be acidic, with a pH below 7, or basic, with a pH above 7. Examples include sulfuric acid, sodium hydroxide, and calcium carbonate. Reactive chemicals are substances prone to violent chemical reactions when in contact with incompatible materials. Certain reactive chemicals are also reactive with themselves or water, such as sodium metal. Toxic chemicals are substances harmful to humans through ingestion, inhalation, or physical contact. Examples include mercury, cyanide, and lead [5].

In general, all chemicals must be stored in a safe location, away from direct sunlight, and sources of heat. Proper chemical storage is essential not only for safety, but also substance integrity and stability. Each chemical must also be stored in consideration of its properties.

Flammable substances and oxidizers must always be stored separately to prevent accidental ignition or combustion. Flammables must be kept in fire-resistant cabinets with proper ventilation and away from direct heat sources. Oxidizers, which can intensify combustion, should also be stored in a well-ventilated area. Corrosive chemicals require resistant storage containers to prevent leakage or corrosion. Acids and bases should be stored apart, as accidental mixing could result in dangerous reactions. Corrosives should also be placed on secondary containment to contain spills.

Reactive substances need to be stored away from incompatible substances. They may require to be stored under inert conditions to maintain stability, such as in a sealed airtight container or under mineral oil. Toxic chemicals require tightly sealed and resistant containers for storage. Such containers must be placed in a controlled area with proper ventilations. Toxic chemicals must also be stored below eye level [6], [7].



	Oxidizing	Flammable	Corrosive: ACID	Corrosive: BASE	Health hazard / toxic
Oxidizing	🟩	🟥	🟨	🟨	🟨
Flammable	🟥	🟩	🟥	🟥	🟨
Corrosive: ACID	🟨	🟥	🟩	🟥	🟥
Corrosive: BASE	🟨	🟥	🟥	🟩	🟨
Health hazard / toxic	🟨	🟨	🟥	🟨	🟩

LEGEND

🟥	🟨	🟩
Not Compatible	Store according to SDS Section 7 and 10	Compatible

Fig. 2.9 Chemical Incompatibility Chart (Source: [7])

Moreover, specific storage conditions such as temperature and humidity, should be considered based on the chemical's requirements. Containers used for storage must to be in good condition, securely holding the chemicals without any leaks. Proper labeling with information regarding the chemical and its hazards is crucial. Those containers are stored in storage facilities requiring regular checks and maintenance. Room temperature substances are usually kept on shelves in cabinets, whereas chemicals needing lower temperatures are placed in specialized refrigerators or freezers.

III. Methodology

The implementation of the source code for this paper utilizes the Python programming language due to the features it offers. In specific, Python provides a built-in matrix data type for graph representation. It also provides sets for handling chemical information.

The chemicals and their relationships are illustrated as a graph, with vertices representing chemicals and edges representing incompatibilities. Each chemical is assigned to an index, starting from 1, based on the order of name input, which must be unique.

The following source code allows for the input of chemical names:

```
def input_chemical_info():
  # the number of chemicals input
  while True:
   try:
    chemical_amt = int(input("Enter the number of
                   chemicals: "))
    if chemical_amt <= 0:
      print("The number of chemicals must be
       positive.")
      continue
    break

   except ValueError:
    print("The number of chemicals must be an
     integer.")

  chemicals = {}
  chemical_names = []
  # chemical name input and validation
  for i in range(chemical_amt):
   while True:
    name = input(f"Enter the name of chemical {i+1}:
            ").strip()
    if not name:
     print("Chemical name cannot be empty. Please
      try again.")

    elif name in chemical_names:
     print(f"The'{name}' has already been added.
      Please enter a different chemical.")

    else:
     chemical_names.append(name)
     break
  ...
```

The chemical information considered for chemical storage allocation using the Welch-Powell graph coloring algorithm include incompatibility relationships and storage temperature. Incompatibilities are based on user input, using the predefined chemical indices. Two storage temperatures are available: cold and room. Cold storage is intended for chemicals requiring refrigeration or freezing, whereas room storage is for those suitable in cabinets or shelves. Validation is done on both incompatibility indices and storage temperature input.

The following source code allows for the input of chemical information:

```
def validate_chemical_indices(chemical_indices,
chemical_amt):
  # no incompatible chemicals
  if not chemical_indices:
   return True

  for idx in chemical_indices:
  # index out of range
   if not (1 <= idx <= chemical_amt):
    return False

   return (len(chemical_indices) == len(set(
    chemical_indices)))

def validate_temperature(temp):
  return (temp in ["cold", "room"])

def input_chemical_info():
  ...
  for i in range(chemical_amt):
   print(f"\nChemical {i+1} ({chemical_names[i]})")

   # chemical incompatibility input and validation
   while True:
    incompatible_str = input(f"Enter the indices of
                            incompatible chemicals: ")
    if incompatible_str:
     try:
      incompatible_chemicals = list(map(int,
                     incompatible_str.split(',')))
      if not validate_chemical_indices
      (incompatible_chemicals, chemical_amt):
        print(f"Invalid chemical indices. Please
         try again.")
        continue
      break
     except ValueError:
      print("Indicies must be separated by commas.
       Please try again.")

    else:
     incompatible_chemicals = []
     break

   # chemical storage temperature input and
     validation
   while True:
    storage_temperature = input("Enter the required
         storage temperature (cold/room): ").lower()
     if validate_temperature(storage_temperature):
      break
     else:
      print("Invalid temperature. Please try
       again.")

    chemicals[i] = {
      "name": chemical_names[i],
      "incompatible_with": incompatible_chemicals,
      "temperature": storage_temperature
    }

  return chemicals, chemical_names
```

The graph is represented by an adjacency matrix for processing. Based on the incompatibility indices from the input, the matrix element is set to zero if the corresponding chemicals are compatible, as it indicates that they are not adjacent. When there is incompatibility, the matrix element is set to one. Storage temperature is not considered in this case.

The following source code creates the adjacency matrix for the chemicals:

```
def create_adjacency_matrix(chemicals):
  num_chemicals = len(chemicals)
  mat = [[0] * num_chemicals for _ in
            range(num_chemicals)]

  for i in range(num_chemicals):
   for j in chemicals[i]["incompatible_with"]:
    mat[i][j-1] = 1  # incompatible
    mat [j-1][i] = 1  # incompatible

  return mat
```

The first step in the Welch-Powell algorithm is to sort vertices, in this case chemicals, by their degree in descending order before performing the traversal coloring process. As the graph is represented using an adjacency matrix, the degree of a vertex is the sum of all the elements in its row.

This following source code sorts the chemicals by their degree:

```
def sort_chemicals_degree(adj_mat):
  degrees = [sum(row) for row in adj_mat]
  sorted_chemicals = sorted(range(len(adj_mat)),
                   key=lambda x: degrees[x],
                   reverse=True)
   return sorted_chemicals
```

Graph coloring, used to allocate storage, is done in an iterative process. It goes through the all the vertices and checks whether the vertex is adjacent with ones assign to the current color. In addition to incompatibility, the storage temperature is evaluated when assigning the color, representing a storage unit. Chemicals with different storage temperature from the initial chemical cannot be assigned to the same storage unit. The process continues until all chemicals are assigned to a storage unit. The assignment is stored in an array.

This following source code checks a chemical and assigns chemicals to storage units:

```
def  check_chemical(adj_mat,  chemical,  chemicals,
chemical_storage, curr_unit):
   for i in range(len(adj_mat)):
    # checks adjacency
    if (adj_mat[chemical][i] == 1) and
    (chemical_storage[i] == curr_unit):
      return False

    # checks temperature
    if (chemicals[chemical]["temperature"] !=
    chemicals[i]["temperature"]) and
    (chemical_storage[i] == curr_unit):
      return False

  return True

def assign_storage(adj_mat, chemicals,
chemical_storage, chemical_amt):
  sorted_chemicals = sort_chemicals_degree(adj_mat)
  for chemical in sorted_chemicals:
   for curr_unit in range(chemical_amt):
    if check_chemical(adj_mat, chemical, chemicals,
    chemical_storage, curr_unit):
      chemical_storage[chemical] = curr_unit
      break

  return chemical_storage
```

The assignment array is processed to group chemicals into storage units. It is initially split based on storage temperatures, then further split based on incompatibility. Chemicals that are compatible and share the same storage temperature are kept in the same storage unit. All storage units are stored in an array.

The following code groups the chemicals based on their requirements:

```
def group_chemicals(adj_mat, chemicals,
chemical_names, chemical_storage):
  result = [[] for _ in range(max(chemical_storage)
           + 1)]
  storage_temp = ['' for _ in range(len(result))]

  # split based on temperatures
  for i in range(len(adj_mat)):
   idx = chemical_storage[i]
   result[idx].append(chemical_names[i])

   if storage_temp[idx] == '':
    storage_temp[idx] = chemicals[i]["temperature"]

   else:
    if storage_temp[idx] !=
    chemicals[i]["temperature"]:
     result.append([chemical_names[i]])
     storage_temp.append(chemicals[i]
      ["temperature"])
     result[idx].remove(chemical_names[i])

  # split based on incompatiblity
  final_result = []
  final_storage_temp = []

  for i in range(len(result)):
   if result[i]:
    temp = storage_temp[i]
    found = False

    for j in range(len(final_result)):
     if final_storage_temp[j] == temp:
      can_merge = True

      for chem in result[i]:
       for existing_chem in final_result[j]:
        idx1 = chemical_names.index(chem)
        idx2 = chemical_names.index(existing_chem)
        if adj_mat[idx1][idx2] == 1:
         can_merge = False
         break

       if not can_merge:
        break

      if can_merge:
       final_result[j].extend(result[i])
       found = True
       break

    if not found:
     final_result.append(result[i])
     final_storage_temp.append(temp)

  return final_result, final_storage_temp
```

The following source code shows the overall flow of the program:

```
def main():
  print("-------------- Chemical Storage Allocator
   --------------")

  chemicals, chemical_names = input_chemical_info()
  # adjacency matrix for the graph
  adj_mat = create_adjacency_matrix(chemicals)

  chemical_amt = len(chemicals)
  # stores the storage number for each chemical
  chemical_storage = [0 for i in range
                     (chemical_amt)]
  chemical_storage = assign_storage(adj_mat,
         chemicals, chemical_storage, chemical_amt)

  print("\nChemical storage allocation:")
  result, storage_temp = group_chemicals(adj_mat,
       chemicals, chemical_names, chemical_storage)

  print(f"The number of storage units needed is
   {len(result)}.")

  for i, color_list in enumerate(result):
   print(f"Storage unit {i+1} ({storage_temp[i]})
    contains: {', '.join(color_list)}")

  print("\nAllocation completed.")
```

IV. RESULTS AND ANALYSIS

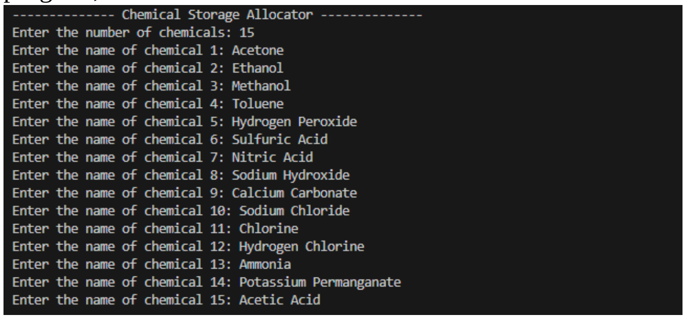
Testing is conducted on the source code in order to evaluate its functionality and generate results. These results are then further analyzed to provide insights on the application of the Welch-Powell graph coloring algorithm in utilizing chemical storage. The test case is comprised of 15 chemicals, each defined by their properties and storage temperature. The chemicals used in this test case are as follows:

1. Acetone (flammable)
   Incompatible with: hydrogen peroxide, sulfuric acid, nitric acid, and potassium permanganate
   Storage temperature: cold
2. Ethanol (flammable)
   Incompatible with: hydrogen peroxide, sulfuric acid, nitric acid, and potassium permanganate
   Storage temperature: cold
3. Methanol (flammable)
   Incompatible with: hydrogen peroxide, sulfuric acid, nitric acid, and potassium permanganate
   Storage temperature: cold
4. Toluene (flammable)
   Incompatible with: hydrogen peroxide, sulfuric acid, nitric acid, and potassium permanganate
   Storage temperature: cold
5. Hydrogen peroxide (reactive)
   Incompatible with: flammable chemicals, corrosive

chemicals, and oxidizers
Storage temperature: cold

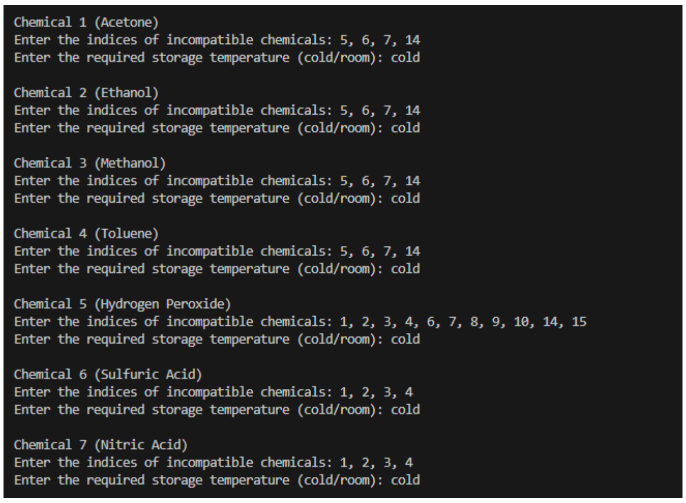
6. Sulfuric acid (corrosive)
   Incompatible with: flammable alcohols (acetone, ethanol, and methanol) and toluene
   Storage temperature: cold
7. Nitric acid (corrosive)
   Incompatible with: flammable alcohols (acetone, ethanol, and methanol) and toluene
   Storage temperature: cold
8. Sodium hydroxide (corrosive)
   Incompatible with: corrosive acids (sulfuric acid and nitric acid)
   Storage temperature: room
9. Calcium carbonate (corrosive)
   Incompatible with: corrosive acids (sulfuric acid and nitric acid)
   Storage temperature: room
10. Sodium chloride (corrosive)
    Incompatible with: corrosive acids (sulfuric acid and nitric acid)
    Storage temperature: room
11. Chlorine (toxic)
    Incompatible with: flammable chemicals, corrosive chemicals, reactive chemicals, and oxidizers
    Storage temperature: room
12. Hydrogen chloride (toxic)
    Incompatible with: flammable chemicals, corrosive chemicals, reactive chemicals, and oxidizers
    Storage temperature: room
13. Ammonia (toxic)
    Incompatible with: flammable chemicals, corrosive chemicals, reactive chemicals, and oxidizers
    Storage temperature: room
14. Potassium permanganate (oxidizer)
    Incompatible with: toxic chemicals, flammable chemicals, corrosive chemicals, reactive chemicals, and oxidizers
    Storage temperature: room
15. Acetic acid (flammable)
    Incompatible with: flammable alcohols (acetone, ethanol, and methanol), toluene, corrosive chemicals, reactive chemicals, and oxidizers
    Storage temperature: room

Information regarding the chemicals is input into the program, as shown below:



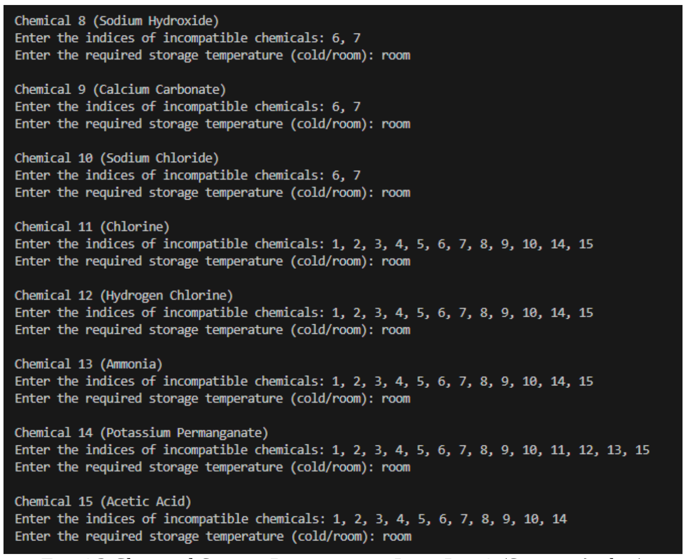
```
------------- Chemical Storage Allocator -------------
Enter the number of chemicals: 15
Enter the name of chemical 1: Acetone
Enter the name of chemical 2: Ethanol
Enter the name of chemical 3: Methanol
Enter the name of chemical 4: Toluene
Enter the name of chemical 5: Hydrogen Peroxide
Enter the name of chemical 6: Sulfuric Acid
Enter the name of chemical 7: Nitric Acid
Enter the name of chemical 8: Sodium Hydroxide
Enter the name of chemical 9: Calcium Carbonate
Enter the name of chemical 10: Sodium Chloride
Enter the name of chemical 11: Chlorine
Enter the name of chemical 12: Hydrogen Chlorine
Enter the name of chemical 13: Ammonia
Enter the name of chemical 14: Potassium Permanganate
Enter the name of chemical 15: Acetic Acid
```

Fig. 4.1 Chemical Names Input (Source: Author)



```
Chemical 1 (Acetone)
Enter the indices of incompatible chemicals: 5, 6, 7, 14
Enter the required storage temperature (cold/room): cold

Chemical 2 (Ethanol)
Enter the indices of incompatible chemicals: 5, 6, 7, 14
Enter the required storage temperature (cold/room): cold

Chemical 3 (Methanol)
Enter the indices of incompatible chemicals: 5, 6, 7, 14
Enter the required storage temperature (cold/room): cold

Chemical 4 (Toluene)
Enter the indices of incompatible chemicals: 5, 6, 7, 14
Enter the required storage temperature (cold/room): cold

Chemical 5 (Hydrogen Peroxide)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 6, 7, 8, 9, 10, 14, 15
Enter the required storage temperature (cold/room): cold

Chemical 6 (Sulfuric Acid)
Enter the indices of incompatible chemicals: 1, 2, 3, 4
Enter the required storage temperature (cold/room): cold

Chemical 7 (Nitric Acid)
Enter the indices of incompatible chemicals: 1, 2, 3, 4
Enter the required storage temperature (cold/room): cold
```

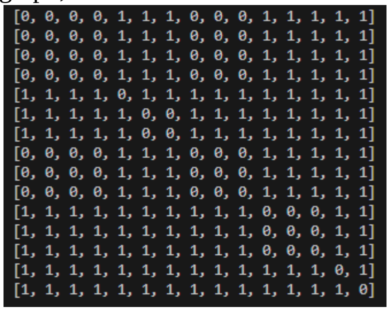
Fig. 4.2 Chemical Storage Requirements Input Part 1 (Source: Author)



```
Chemical 8 (Sodium Hydroxide)
Enter the indices of incompatible chemicals: 6, 7
Enter the required storage temperature (cold/room): room

Chemical 9 (Calcium Carbonate)
Enter the indices of incompatible chemicals: 6, 7
Enter the required storage temperature (cold/room): room

Chemical 10 (Sodium Chloride)
Enter the indices of incompatible chemicals: 6, 7
Enter the required storage temperature (cold/room): room

Chemical 11 (Chlorine)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15
Enter the required storage temperature (cold/room): room

Chemical 12 (Hydrogen Chlorine)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15
Enter the required storage temperature (cold/room): room

Chemical 13 (Ammonia)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15
Enter the required storage temperature (cold/room): room

Chemical 14 (Potassium Permanganate)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15
Enter the required storage temperature (cold/room): room

Chemical 15 (Acetic Acid)
Enter the indices of incompatible chemicals: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14
Enter the required storage temperature (cold/room): room
```

Fig. 4.3 Chemical Storage Requirements Input Part 2 (Source: Author)

Based on the chemical incompatibilities, an adjacency matrix —which is not displayed in the actual program—is formed to represent the graph, as seen below:



```
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
```

Fig. 4.4 Chemical Incompabilities Represented as An Adjacency Matrix (Source: Author)

The chemicals are then allocated to storage units based on their incompatibilities, represented in the adjacency matrix, and storage temperature, using the Welch-Powell graph coloring algorithm.

The following figures illustrate the stages of manual graph coloring to allocate chemical storage using the algorithm. Chemicals are denoted as C<index>, for example, chemical 1 as C1. In this representation, an edge is drawn between two chemicals if they are incompatible with one another chemical. The number of storage units required to store the chemicals is seven, which is the chromatic number derived from the manual process of graph coloring.

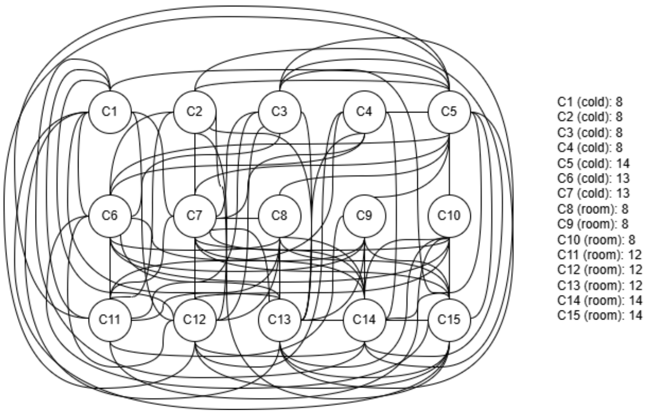


Fig. 4.5 Manual Chemical Storage Allocation Initial State (Source: Author)

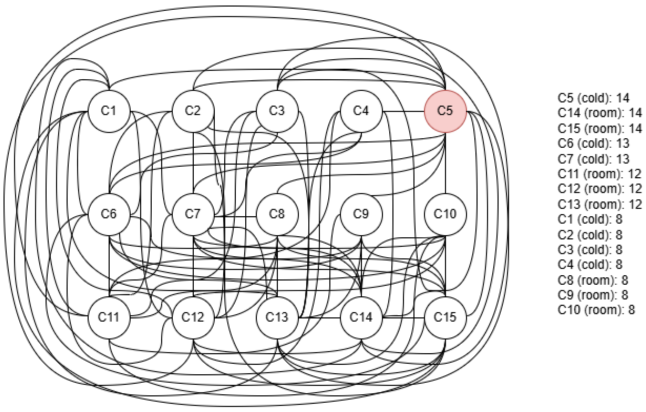


Fig. 4.6 Manual Chemical Storage Allocation Stage 1 (Source: Author)

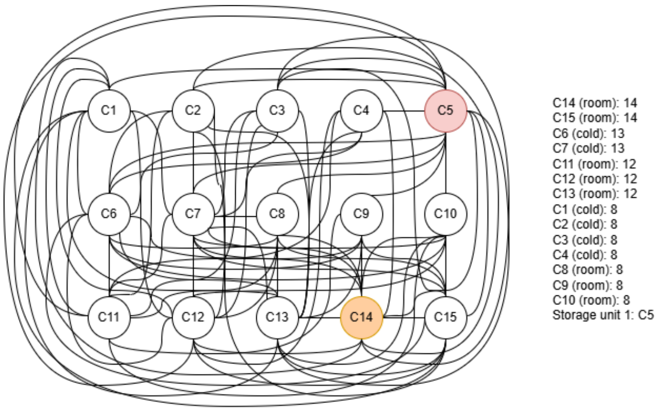


Fig. 4.7 Manual Chemical Storage Allocation Stage 2 (Source: Author)

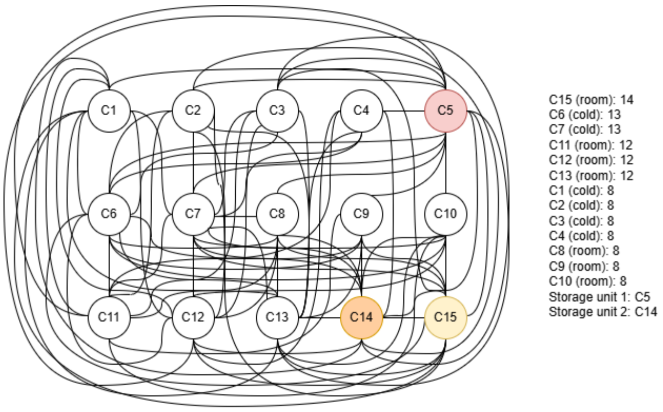


Fig. 4.8 Manual Chemical Storage Allocation Stage 3 (Source: Author)

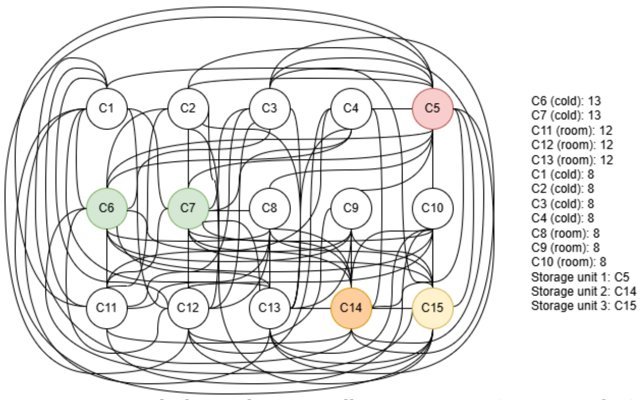


Fig. 4.9 Manual Chemical Storage Allocation Stage 4 (Source: Author)

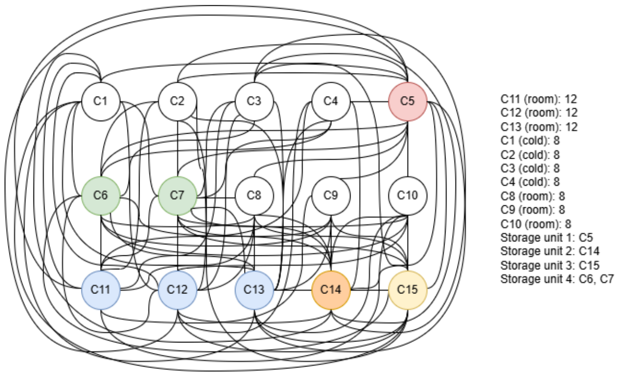


Fig. 4.10 Manual Chemical Storage Allocation Stage 5 (Source: Author)

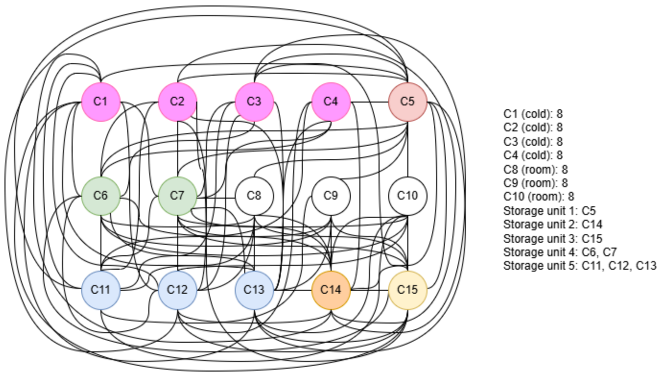


Fig. 4.11 Manual Chemical Storage Allocation Stage 6 (Source: Author)

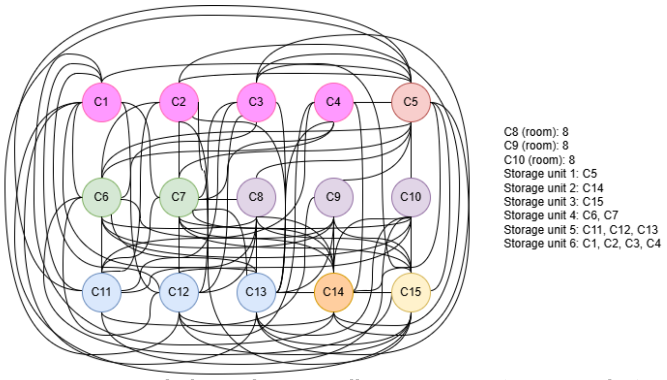


Fig. 4.12 Manual Chemical Storage Allocation Stage 7 (Source: Author)

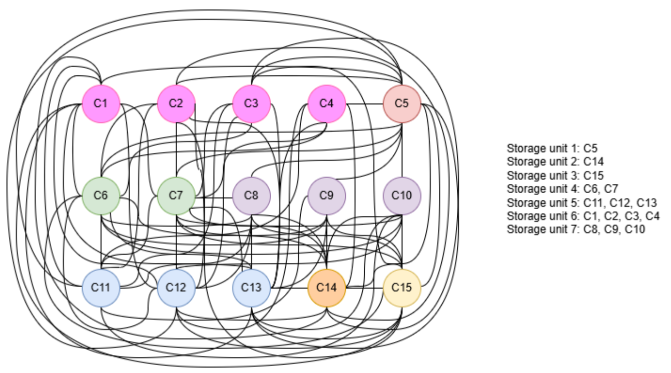
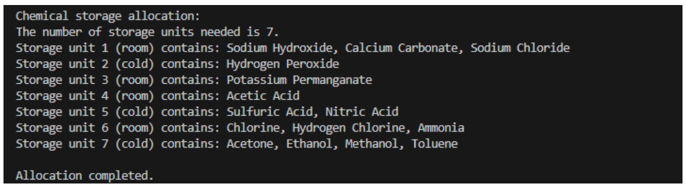


Fig. 4.13 Manual Chemical Storage Allocation Final State (Source: Author)

The storage allocation result produced by the program is as follows:



```
Chemical storage allocation:
The number of storage units needed is 7.
Storage unit 1 (room) contains: Sodium Hydroxide, Calcium Carbonate, Sodium Chloride
Storage unit 2 (cold) contains: Hydrogen Peroxide
Storage unit 3 (room) contains: Potassium Permanganate
Storage unit 4 (room) contains: Acetic Acid
Storage unit 5 (cold) contains: Sulfuric Acid, Nitric Acid
Storage unit 6 (room) contains: Chlorine, Hydrogen Chlorine, Ammonia
Storage unit 7 (cold) contains: Acetone, Ethanol, Methanol, Toluene

Allocation completed.
```

Fig. 4.14 Automated Chemical Storage Allocation (Source: Author)

The chemical groupings within each unit between the manual process and program are consistent, confirming the correctness of the program's implementation. The numbering of storage units may differ due to the chemical grouping algorithm, which processes the chemicals in two steps. Chemicals are initially grouped based on their storage temperatures, then they are further organized based on their incompatibilities.

V. Conclusion

This study successfully optimizes chemical storage using the Welch-Powell graph coloring algorithm. The developed program assigns chemicals into storage units, taking into account both their incompatibilities and storage temperature. The program has been proved to be highly efficient when handling a large number of chemicals.

Further development of the program could allow it to manage additional chemical requirements, such as humidity control. More specific needs, like temperature ranges for storage, can could also be introduced to the program. A feature for automating the assignment of incompatibilities would be a great addition to the program, eliminating the need for users to manually input each chemical's index.

In conclusion, the application of the Welch-Powell algorithm for optimizing chemical storage provides valuable insights. By automating the chemical allocation process, it reduces the likelihood of errors that occur in manual processing. The automation of chemical storage allocation also makes the process more time-efficient. This leads to a more reliable and optimized chemical storage system. The knowledge gained from this study contributes to advancements in safe chemical storage practices and can be applied to similar allocation systems.

VI. Appendix

The source code for this paper, titled *Optimizing Chemical Storage Utilizing the Welch-Powell Algorithm*, is available at:
https://github.com/naomirisaka/Makalah-Matdis

VII. Acknowledgment

First and foremost, the author expresses deep gratitude to God Almighty for His guidance and strength in completing this study. The author also wishes to extend sincere appreciation to Ir. Rila Mandala, M.Sc., Ph.D., and Dr. Ir. Rinaldi Munir, M.T., for their invaluable knowledge and guidance as the author's lecturers in Discrete Mathematics. Their insights were instrumental in the development of this work. Heartfelt thanks are also due to the author's family and friends for their unwavering support and encouragement throughout this study.

References

[1] R. Munir, "Graf Bagian 1", IF1220 Matematika Diskrit, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. [Accessed: Dec. 26, 2024].
[2] J. Harris, J. L. Hirst, and M. Mossinghoff, *Combinatorics and Graph Theory (CGT)*, University of Notre Dame. [Online]. Available: https://www3.nd.edu/~dgalvin1/40210/40210_F12/CGT_early.pdf. [Accessed: Dec. 27, 2024].
[3] R. Munir, "Graf Bagian 2", IF1220 Matematika Diskrit, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf. [Accessed: Dec. 27, 2024].
[4] R. Munir, "Graf Bagian 3", IF1220 Matematika Diskrit, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf. [Accessed: Dec. 27, 2024].
[5] Moreno Valley Fire Department, *Chemical Classification Guidelines*, Moreno Valley Government, 2023. [Online]. Available: https://moval.gov/departments/fire/pdf/dev-guides/ChemicalClassificationGuidelines.pdf. [Accessed: Dec. 28, 2024].
[6] Smithsonian Institution, "Chapter 19 Chemical Handling and Storage", Smithsonian Institution. [Online]. Available: https://www.si.edu/sites/default/files/unit/oshem/ch_19_chemical_handling.pdf . [Accessed: Dec. 28, 2024].
[7] EPFL, "Chemical Storage", EPFL. [Online]. Available: https://www.epfl.ch/campus/security-safety/en/lab-safety/hazards/chemical-hazards/chemicals-storage/. [Accessed: Dec. 28, 2024].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Januari 2025

[illegible]

Naomi Risaka Sitorus – 13523122