Alexandria University
Faculty of Engineering
CSED 2025

Discrete Lab 2

Report

Submitted to

Eng. Omar Elsherif

Faculty of Engineering, Alex. University

Submitted by

Islam Yasser Mahmoud 20010312

Mkario Michel Azer 20011982

Faculty of Engineering, Alex. University

Nov. 2022

Problem Statement:

Generating all possible subsets of a set represented as array list of distinct strings using:

I. Iterative approach
II. Recursive approach

Used data structures:

- **Iterative approach:**

Vector of strings (Universe) which stores the elements of the Universe set.

- **Recursive approach:**

Vector of strings (Universe) which stores the elements of the Universe set

Vector of strings (PossibleSet) which stores the subsets dynamically to print them

Algorithms used:

- **Iterative approach:**
  - Get the set and its size(n) from the user.
  - Get the size of the power set which equals: (**2^n**) which indicates our total combinations.
  - Iterate on the combinations one by one and make a bit test on the first n bits each one of them to see which bit is set then print the corresponding element of the bit's place in the set. Then by the end of this iterations you will get all the subsets. (Bitmasking approach)

- **Recursive approach:**
  - Get the set and its size(n) from the user
  - Then call the recursive function with default value of index equal 0
  - If index became equal n, then print current PossibleSet
  - Otherwise call the function recursively twice (Backtracking approach)

Code Snippets:

```
8   int n;
9   vector<string> Universe;
10
11  // iterative approach
12  void powerSet_it ()
13  {
14      int set_size = n;
15
16      int power_set_size = pow(2,set_size);
17
18      for(int i=0 ; i<power_set_size ; i++)
19      {
20          for(int j=0 ; j<set_size ; j++)
21          {
22              if(i & (1 << j))                    // if the bit is set
23                  cout << Universe[j] << " ";
24          }
25          cout << endl;
26      }
27  }
```

```
29  // recursive approach
30  vector<string> PossibleSet;
31
32  void GenerateSets(int index=0){
33      if(index == n){
34          for(int i=0;i<PossibleSet.size();i++){
35              cout << PossibleSet[i] << " ";
36          }
37          cout << endl;
38          return;
39      }
40
41      PossibleSet.push_back(Universe[index]);
42      GenerateSets(index +1);
43      PossibleSet.pop_back();
44      GenerateSets(index+ 1);
45  }
```

```
47  int main()
48  {
49      cout << "Enter the number of elements of the set" << endl;
50      cin >> n;
51      cout << "Enter elements of the set" << endl;
52      string element;
53      for(int i=0;i<n;i++){
54          cin >> element;
55          Universe.push_back(element);
56      }
57      short x;
58      cout << "Enter 1 for iterative approach & 2 for recursive approach:" << endl;
59      cin >> x;
60      cout << "The subsets are:" << endl;
61      switch(x)
62      {
63          case 1:
64              powerSet_it();
65              break;
66          case 2:
67              cout << endl;
68              GenerateSets();
69              break;
70      }
71      return 0;
72  }
```

Sample runs:

- **Iterative approach:**

```
Enter the number of elements of the set
1
Enter elements of the set
4
Enter 1 for iterative approach & 2 for recursive approach:
1
The subsets are:

4

Process returned 0 (0x0)   execution time : 7.165 s
Press any key to continue.
```

```
Enter the number of elements of the set
2
Enter elements of the set
1
2
Enter 1 for iterative approach & 2 for recursive approach:
1
The subsets are:

1
2
1 2

Process returned 0 (0x0)   execution time : 15.901 s
Press any key to continue.
```

Note that:

the space on the first line of the subsets indicates the empty set.

```
Enter the number of elements of the set
4
Enter elements of the set
0
1
2
3
Enter 1 for iterative approach & 2 for recursive approach:
1
The subsets are:

0
1
0 1
2
0 2
1 2
0 1 2
3
0 3
1 3
0 1 3
2 3
0 2 3
1 2 3
0 1 2 3

Process returned 0 (0x0)   execution time : 12.745 s
Press any key to continue.
```

```
Enter the number of elements of the set
0
Enter elements of the set
Enter 1 for iterative approach & 2 for recursive approach:
1
The subsets are:


Process returned 0 (0x0)   execution time : 8.091 s
Press any key to continue.
```

- **Recursive approach**

```
Enter the number of elements of the set
1
Enter elements of the set
1
Enter 1 for iterative approach & 2 for recursive approach:
2
The subsets are:

1


Process returned 0 (0x0)   execution time : 21.117 s
Press any key to continue.
```

```
Enter the number of elements of the set
3
Enter elements of the set
1
2
3
Enter 1 for iterative approach & 2 for recursive approach:
2
The subsets are:

1 2 3
1 2
1 3
1
2 3
2
3


Process returned 0 (0x0)   execution time : 14.903 s
Press any key to continue.
```

Assumptions and necessary details to be known:

- **Iterative approach**
  - Bitwise and , left shift
  - Bitmasking approach
- **Recursive approach**
  - Backtracking approach