HEAVEN'S LIGHT IS OUR GUIDE

RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

CSE-2102

LAB-3

Discrete Mathematics Sessional

Submitted To:
Suhrid Shakhar Ghosh
Asst. Professor
Dept. of Computer Science & Engineering

Submitted By :
Kaif Ahmed Khan
ID: 2103163
Dept. of Computer Science & Engineering

December 21, 2023

1 Equality of Sets

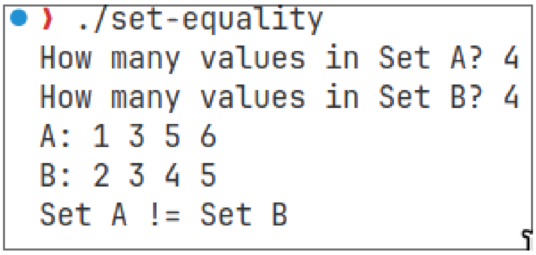
Take two sets A, B as input from user. Find whether these two sets are equal or not.

1.1 Source Code

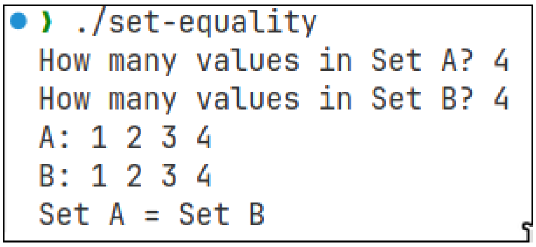
```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n;
  cout << "How many values in Set A? ";
  cin >> n;
  int p;
  cout << "How many values in Set B? ";
  cin >> p;
  int setA = 0;
  int setB = 0;
  cout << "A: ";
  while(n--){
    int x;
    cin>> x;
   setA |= (1<<x);
  }
  cout << "B: ";
  while(p--){
    int x;
    cin>> x;
   setB |= (1<<x);
  }
  if(setA == setB) cout << "Set A = Set B" << endl;
  else cout << "Set A != Set B" << endl;
}
```

1.2 Output



(a) Two sets are not equal



(b) Two sets are equal

Figure 1: Output of Source Code 1.1

1.3 Analysis

The code successfully checks whether the two sets A and B are equal or not and prints the relevant output. For storing the inputs as set members I have used the bit manipulation method, where each bit represent the decimal number from 0 to 32. For example, if 0000 is a 4 bit integer, then by using the left shift operator I can shift 1 to desired position and ORed with the integer. so `setA |= (1<<2)` will shift 1 to bit position 2 i.e. set A contains the element 2.

Finally if `setA,setB` will be same integer if the sets are equal. Otherwise they are not equal. Limitation of this process is that only numbers from 0 to 31 can be stored as set elements.

The complexity of the code is $\mathcal{O}(max(n, p))$ where n and p is the size of set A and B respectively.

2 Power Sets

Take a set A as input from user. Print the power set of A as output.

2.1 Source Code

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n;
  cout << "How many values in Set A? ";
  cin >> n;
  int setA = 0;
  cout << "A: ";
  while(n--){
    int x;
    cin>> x;
    setA |= (1<<x);
  }
  int b = 0;
  cout << "{ ";
  do{
    cout << "{";
    for(int i = 0; i<32;i++){
      if(b & (1<<i)) cout << i << " ";
    }
    cout << "}" <<endl;
  }while((b=(b-setA)&setA));

  return 0;
}
```

2.2 Output

```
❯ ./powerset
How many values in Set A? 3
A: 1 2 3
{ {}
{1 }
{2 }
{1 2 }
{3 }
{1 3 }
{2 3 }
{1 2 3 }
```

Figure 2: Power set of $A = \{1, 2, 3\}$

2.3 Analysis

The program generates the power set of a set A. The input of set is taken similarly as in source code 1.1 using bit manipulation. In each iteration of the do-while loop the assumed subset `b` is subtracted from the set A and then the intersection (using bitwise AND) with set A is set to `b` thus the subsets are generated.

The complexity of the code is $\mathcal{O}(2^n)$ where n is the size of set A.

3 Cartesian Products

Take two sets A,B as input. Print the Cartesian product as output.

3.1 Source Code

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n;
  cout << "How many values in Set A? ";
  cin >> n;
  int p;
  cout << "How many values in Set B? ";
  cin >> p;
  int setA = 0, setB = 0;
  cout << "A: ";
  while(n--){
    int x; cin>> x;
   setA |= (1<<x);
  }
  cout << "B: ";
  while(p--){
    int x; cin>> x;
   setB += (1<<x);
  }
```

```cpp
    cout << "Cartesian Product:\n" << "{" << endl;
    for(int i = 0; i < 32; i++){
      if(setA & (1<<i)){
        for(int j = 0; j < 32; j++){
          if(setB & (1<<j)){
            printf("\t(%d,%d)\n", i,j);
          }
        }
      }
    }
    cout << "}" << endl;
}
```

3.2 Output

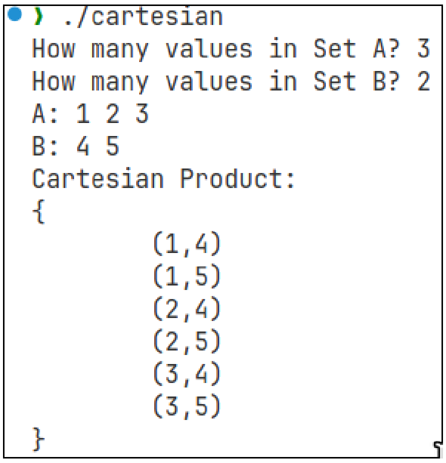


Figure 3: Cartesian product of $\mathcal{A} \times \mathcal{B}$

3.3 Analysis

The program prints the Cartesian product of two sets $\mathcal{A}, \mathcal{B}$. The input process of set is similar to the source code 1.1, and 1.2. According to the definition of Cartesian product of set, $(\mathcal{A} \times \mathcal{B})$ is a set of ordered pairs (x, y) where $x \in \mathcal{A}$ and $y \in \mathcal{B}$. The for loop iterating from $0 \ldots 31$ firstly checks whether the bit of `setA` is 1 or 0. if it is 1 then it similarly iterates over the `setB` and prints the ordered pair (x, y).

The complexity of the code is $\mathcal{O}(max(n, p))$ where n and p is the size of set A and B respectively.