CLUB DE PROGRAMACIÓN COMPETITIVA UMSA

INF – 143
TALLER DE PROGRAMACIÓN

FEBRERO 2022

UMSA
la mejor

Presentación

Univ. Grover Osvaldo Rodriguez Apaza

Carrera de Informática
Facultad de Ciencias Puras y Naturales

18 de Febrero de 2022

Contenido

1 Conjuntos

¿Qué es un conjunto?

Un conjunto es la agrupación de diferentes elementos que comparten entre sí características y propiedades semejantes

Ejemplo

Conjunto de números primos

$S = \{2, 3, 5, 7, 331, 1000000007, \ldots, 19\}$

Lo que generalmente se realiza, es operaciones sobre dicho conjunto.

Problema de motivación

Subconjuntos

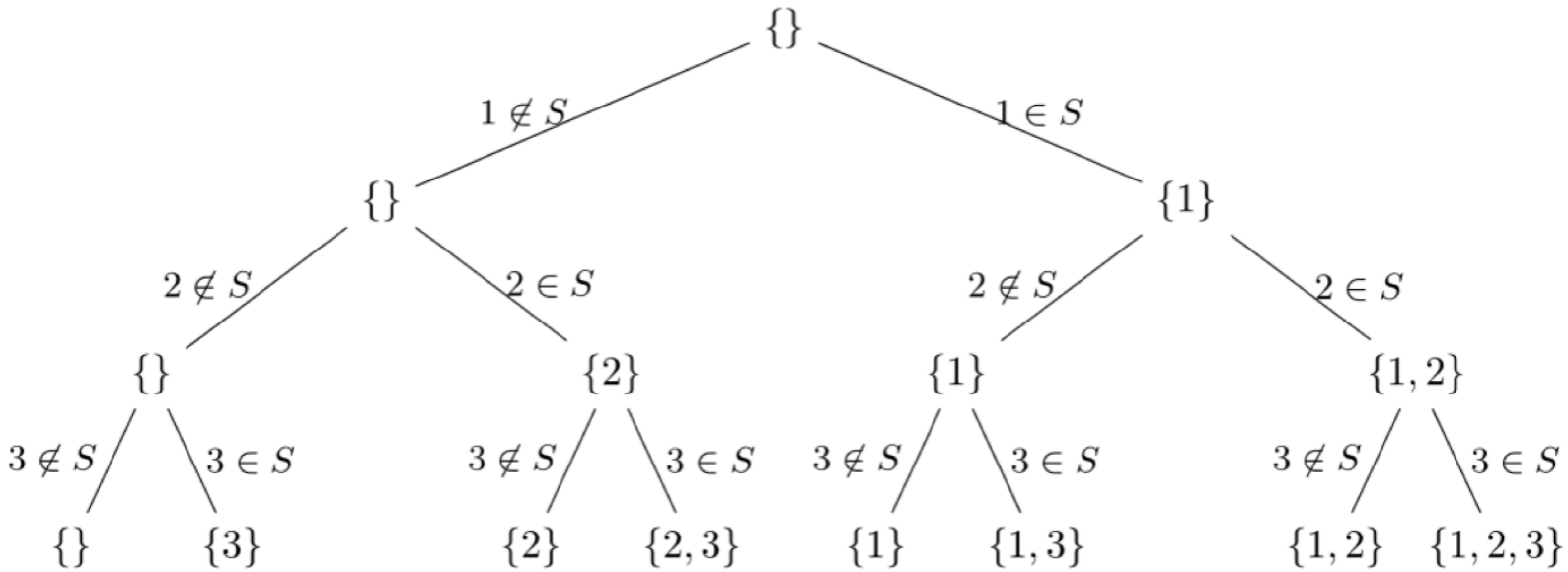
Descripción

Dado un conjunto $|S|$, se quiere hallar todos los subconjuntos de S, además se sabe que la cantidad de subconjuntos que tiene un conjunto de tamaño n es 2^n.

Ejemplo

Si $n = 3$, y el conjunto S es: $S = \{1, 2, 3\}$
los subconjuntos de S son: $\{\emptyset\}$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$

Solución



Código

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int v[N];
bool estado[N];
int n;
void subset(int i){
    if(i == n){
        cout << "{";
        for(int i = 0; i < n; i++)
            if(estado[i])
                cout << v[i] << ", ";
        cout << "}\n";
        return;
    }

    subset(i + 1);
    estado[i] = 1;
    subset(i + 1);
    estado[i] = 0;
}
```

Código

```
1  int main(){
2
3      cin >> n;
4      for(int i = 0; i < n; i++)
5          cin >> v[i];
6      subset(0);
7      return 0;
8  }
```

Complejidad $O(2^n \cdot n)$

Ejemplo 1

Conjuntos

Descripción

Dado un conjunto S de n elementos, contar cuantos subconjuntos tienen un numero impar.

Entrada

La entrada consiste de un número entero n $(1 \leq n \leq 22)$, el tamaño del conjunto, luego vienen los s_i números $(1 \leq s_i \leq 10^{18})$. los elementos del conjunto

Salida

Contar cuantos subconjuntos del conjunto S tiene un número impar.

Ejemplo Entrada	Ejemplo Salida
3 1 2 3	6

Código

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int A[N];
bool estado[N];
int n;
void subconjuntos(int i, int &ans){
    if(i == n){
        for(int i = 0; i < n; i++){
            if(estado[i]){
                if(A[i] % 2 == 1){
                    ans++;
                    break;
                }
            }
        }
        return;
    }
    // no tomar al elemento
    estado[i] = 0;
    subconjuntos(i + 1, ans);
    // tomar al elemento
    estado[i] = 1;
    subconjuntos(i + 1, ans);
}
```

Código

```
1  int main(){
2      cin >> n;
3      for(int i = 0; i < n; i++)
4          cin >> A[i];
5
6      int ans = 0;
7      subconjuntos(0, ans);
8
9      cout << ans << '\n';
10
11     return 0;
12 }
```

Complejidad $O(2^n \cdot n)$

Ejemplo 2

Conjuntos 2

Descripción

Dado un conjunto S de n elementos, contar cuantos numeros diferentes se pueden formar con los subconjuntos del mismo conjunto, donde el resultado de cada subconjunto es la suma de sus elementos.

Entrada

La entrada consiste de un número entero n $(1 \leq n \leq 22)$, el tamaño del conjunto, luego vienen los s_i números $(1 \leq s_i \leq 10^5)$. los elementos del conjunto

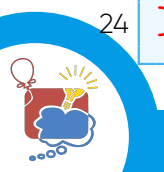
Salida

Contar cuantos números diferentes se pueden formar.

Ejemplo Entrada	Ejemplo Salida
3 1 2 3	6

Código

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
const int MX = 2200005;
int A[N], n;
bool estado[N], FRE[MX];
void subconjuntos(int i){
    if(i == n){
        int cur = 0;
        for(int i = 0; i < n; i++){
            if(estado[i]){
                cur += A[i];
            }
        }
        FRE[cur] = true;
        return;
    }
    // no tomar al elemento
    estado[i] = 0;
    subconjuntos(i + 1);
    // tomar al elemento
    estado[i] = 1;
    subconjuntos(i + 1);
}
```



Código

```
1  int main(){
2      cin >> n;
3      for(int i = 0; i < n; i++)
4          cin >> A[i];
5
6      subconjuntos(0);
7      int ans = 0;
8      for(int i = 1; i < MX; i++)
9          ans += FRE[i];
10
11     cout << ans << '\n';
12
13     return 0;
14 }
```

Complejidad $O(2^n \cdot n)$

2 Permutaciones

¿Qué es una permutacion?

Una permutación es la variación del orden o posición de los elementos de un conjunto ordenado o una tupla

Ejemplo

Permutacion de n elementos diferentes $n = 3$ $S = \{1, 2, 3\}$ la cantidad de permutaciones es: $n!$, entonces $3! = 6$, y las permutaciones son:

$$\{1, 2, 3\}$$
$$\{1, 3, 2\}$$
$$\{2, 1, 3\}$$
$$\{2, 3, 1\}$$
$$\{3, 1, 2\}$$
$$\{3, 2, 1\}$$

Problema de motivación

Permutaciones

Descripción

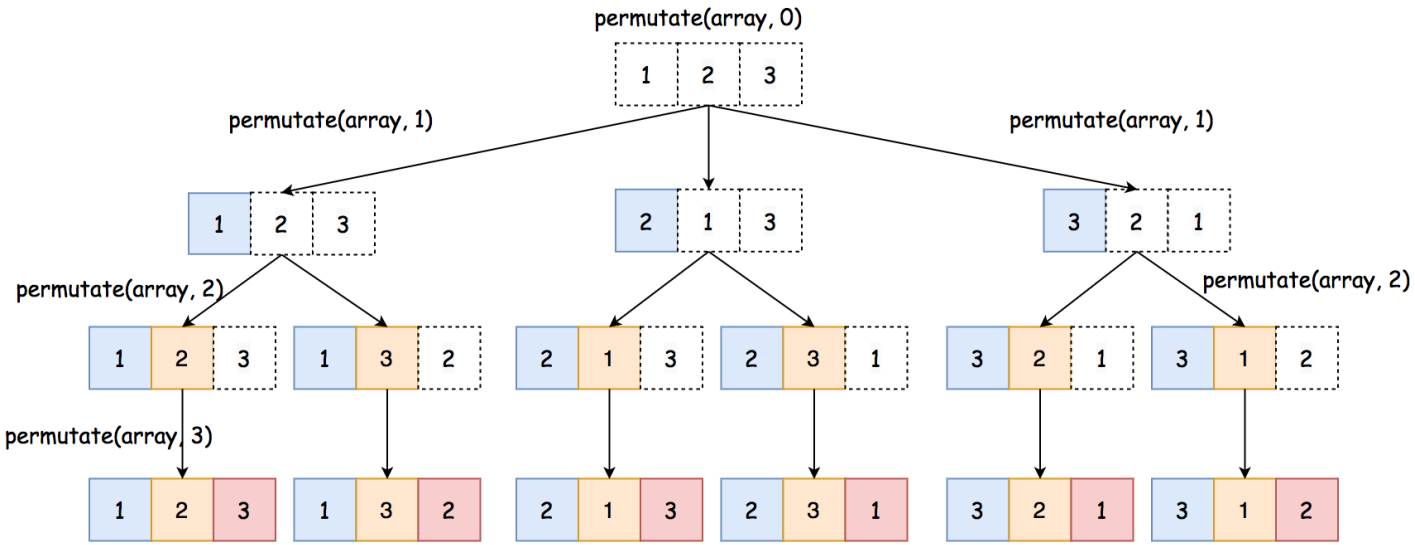
Dado un conjunto $|S|$, se quiere hallar todas las permutaciones de S, se garantiza que los elementos seran diferentes

Ejemplo

Si $n = 3$, y el conjunto S es: $S = \{1, 2, 3\}$
todas las permutaciones son:

$$\{1, 2, 3\}$$
$$\{1, 3, 2\}$$
$$\{2, 1, 3\}$$
$$\{2, 3, 1\}$$
$$\{3, 1, 2\}$$
$$\{3, 2, 1\}$$

Solución


permutate(array, 0)
1 2 3
permutate(array, 1)
permutate(array, 1)
1 2 3
2 1 3
3 2 1
permutate(array, 2)
permutate(array, 2)
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
3 1 2
permutate(array, 3)
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
3 1 2

Código

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int v[N];
bool estado[N];
int n;
void permutacion(vector<int> &A){
    if(A.size() == n){
        for(int i = 0; i < n; i++)
            cout << A[i] << ", ";
        cout << '\n';
        return;
    }
    for(int i = 0; i < n; i++){
        if(estado[i] == false){
            A.push_back(v[i]); // tomar parte de la permutacion
            estado[i] = true;
            permutacion(A);
            A.pop_back(); // quitar de la permutacion
            estado[i] = false;
        }
    }
}
```

Código

```
1  int main(){
2
3      cin >> n;
4      for(int i = 0; i < n; i++)
5          cin >> v[i];
6
7      vector<int> A;
8      permutacion(A);
9      return 0;
10 }
```

Complejidad $O(n!)$

Next Permutation

next_permutation(inicio, final) es una función de c++, que permite encontrar la primera permutación lexicográficamente mayor.
Para usar esta función se debe importar la libreria **algorithm**
para hallar todas las permutaciones de un array A, se debe ordenar el vector

Código

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
int v[N];
int n;
int main(){

    cin >> n;
    for(int i = 0; i < n; i++)
        cin >> v[i];
    // ordenar antes de usar next permutation para
    // obtener todas las permutaciones
    do{
        for(int i = 0; i < n; i++)
            cout << v[i] << ", ";
        cout << '\n';
    }while(next_permutation(v, v + n));
    return 0;
}
```

Complejidad $O(n!)$

3

Backtraking

¿Qué es backtraking?

Es una técnica algorítmica para resolver problemas de forma recursiva intentando construir una solución de forma incremental, pieza a pieza, eliminando aquellas soluciones que no satisfacen las restricciones del problema en algún momento.

Problema de motivación

Laberinto

Descripción

Dado un tablero A de tamaño n x m el cual representa a un laberinto, donde la entrada es la posición $A_{0,0}$ y la salida es $A_{n-1,m-1}$

Se quiere encontrar todos los posibles caminos que hay para salir del laberinto

- '.' denota un espacio libre en el laberinto
- '#' denota una pared en el laberinto

para moverse en el laberinto se puede hacer solo hacia la derecha y abajo, además solo se puede mover por espacios libres

Entrada

La entrada consiste de dos números enteros n y m, $(1 \leq n, m \leq 8)$, el tamaño del laberinto, luego vienen los n lineas, cada linea tiene m caracteres $A_{i,j} \in \{'.', '\#'\}$

Salida

Mostrar todos los caminos posibles para salir del laberinto

Ejemplo Entrada	Ejemplo Salida
3 3 ... .#.	*.. *** .#* **. **. .#* *** *.. .#*

Código

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, m;
char A[9][9];
bool estado[9][9];
void backtraking(int i, int j){
    if(i == n - 1 and j == m - 1){
        estado[0][0] = 1;
        for(int i = 0; i < n; i++){
            for(int j = 0; j < m; j++){
                if(estado[i][j])
                    cout << '*';
                else
                    cout << A[i][j];
            }
            cout << '\n';
        }
        cout << "\n";
        return;
    }
    if(i + 1 < n && A[i + 1][j] != '#'){ // ir abajo
        estado[i + 1][j] = 1;
        backtraking(i + 1, j);
        estado[i + 1][j] = 0;
    }
```

Código

```
    // ir a la derecha
    if(j + 1 < m && A[i][j + 1] != '#'){
        estado[i][j + 1] = 1;
        backtraking(i, j + 1);
        estado[i][j + 1] = 0;
    }
}
int main(){
    cin >> n >> m;
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            cin >> A[i][j];
        }
    }

    backtraking(0, 0);

    return 0;
}
```

¿Preguntas?

¡Gracias!