Universidade Federal do Ceará

Campus de Quixadá

QXD0153 - Desafios de Programação

Produza o código mais conciso possível para as seguintes tarefas:

1. Dado uma lista inteiros aleatórios, imprima os elementos distintos (únicos) de maneira ordenada.

   Exemplo de Entrada v = 40,20,10,20,30,10

   Exemplo de Saída 10 20 30 40

   Pesquise a função unique da STL do C++.

   Esse problema pode ser resolvido usando as funções sort e unique.

   Solução Ana Paula:

```
void unique_(vector<int>& n){

  sort(n.begin(), n.end());
  std::vector<int>::iterator it;
  it = unique(n.begin(), n.end());
  n.resize(distance(n.begin(), it));
}
```

   Solução Décio:

```
list <int> lista;
lista.sort();
lista.unique();
```

2. Dado uma lista ordenada de inteiros L com tamanho máximo de 1M de itens, determine quantas vezes o valor v existe em L. Pesquise a função lower_bound da STL do C++.

   Esse problema pode ser resolvido em O(log n) depois do vetor ordenado. Basta utilizar a função lower_bound e upper_bound.

   Solução João Victor:

```
sort(v.begin(),v.end());
vector<int>::iterator low,up;
low=lower_bound (v.begin(), v.end(), valor_v); //
up= upper_bound (v.begin(), v.end(), valor_v); //
printf("%d␣\n\n",(up-low));
```

3. Gere todas as permutações de $\{'A','B','C',...,'J'\}$ com as das primeiras do alfabeto. Pesquise a função next_permutation da STL do C++.

   O problema é resolvido através da simples aplicação da função next_permutation

   char letras [] = 'A','B','C','D','E','F','G','H','I','J';

   Solução Enoque:

```
while ( std::next_permutation(letras,letras+10) ){
        for(int i = 0; i < 10; i++){
          cout << letras[i] << "␣";
        }
        cout << endl;
}
```

4. Gere todos os possíveis subconjuntos de $\{0, 1, 2, \ldots, N-1\}$, para $N = 20$.

   Dica: Use as operações bit-a-bit:

   - Cada número inteiro $[0, \ldots, (1 << N) - 1]$ representa um subconjunto de um conjunto de N elementos. Por exemplo, 0 representa o conjunto vazio e $(1 << N) - 1$ representa o conjunto de todos os elementos.
   - Para verificar se o elemento j está no conjunto i, basta verificar se $(i\&(1 << j)) > 0$

   Solução Vinicius:

```
for(int i = 0; i < (1<<N); i++){
    for(int j = 0; j < N; j++){
        if((i&(1 << j)) > 0){
            cout << j+1 << "␣";
```

```
        }
    }
    cout << endl;
}
```

5. Determine se existe dois elementos distintos x e y em L tal que $x+y = S$, onde n ($1 \leq n \leq 10^6$) é o tamanho do vetor.

   Solução Pedro Olimpío:

```
sort(lista.begin(), lista.end());
vector<int>::iterator it;

//Complexidade (n log n)
for (it = lista.begin(); it != lista.end(); it++){
    if (binary_search(lista.begin(), lista.end(), s - *it)){
        printf("%d %d\n", *it, s - *it);
        return 0;
    }
}
printf("nao existe\n");
```

   Solução Raul:

```
sort(lista.begin(), lista.end());

int ini = 0;
int fim = lista.size()-1;

//Complexidade O(n)
while(ini < fim){
    if(lista[ini] + lista[fim] == S){
        cout << "Achei!" << endl;
        cout << '[' << ini << "] = " << lista[ini] << endl;
        cout << '[' << fim << "] = " << lista[fim] << endl;
```

```
            break;
        }
        else if(lista[ini] + lista[fim] > S){
            fim--;
        }
        else{
            ini++;
        }
}
```

Solução Enoque:

```
map <int, int> myMap;

for (int i = 0; i < TAM; i++){
            scanf ("%d", &vetor[i]);
            myMap[vetor[i]] = 1;
}

//Complexidade O(n log n)
for (int i = 0; i < TAM; i++){
        if (myMap[s-vetor[i]] != 0){
            count++;
            break;
        }
}

if (count == 0){
            printf ("NAO PODE");
}
else{
            printf("PODE");
}
```