Problemas - Hashing

Prof. Dr. Juliano Henrique Foleis

Resolva os exercícios sugeridos abaixo. Note que para serem eficientes, as soluções abaixo devem ser implementadas usando *hashing* (em `C++` você pode usar as classes `unordered_map` e `unordered_set`). Juntamente com cada exercício, um protótipo em `C++` é fornecido.

Exercícios

1. Dado um vetor de inteiros `arr`, conte a frequência de cada inteiro.

```
#include <unordered_map>
#include <vector>

/*Retorna: um dicionario com a frequencia de cada inteiro, onde a chave é o inteiro
e o valor é a frequência.*/
std::unordered_map<int,int> contarFrequencia(std::vector<int>& arr);
```

2. Dado um vetor de inteiros `arr`, retorne a posição do primeiro elemento do vetor que não tem duplicata. Se todos os elementos ocorrerrem pelo menos duas vezes, retorne `-1`.

```
#include <unordered_map>
#include <vector>

int primeiroUnico(std::vector<int>& arr);
```

3. Dado um vetor de inteiros `arr`, retorne um par de inteiros que somados resultam em um valor `k` dado.

```
#include <unordered_set>
#include <vector>
#include <utility>

std::pair<int, int> twosum(std::vector<int>& arr, int k);
```

4. Dado um vetor de inteiros `arr`, retorne `true` se `arr` contém algum elemento repetido, e `false` caso contrário.

```
#include <unordered_set>
#include <vector>

bool temDuplicatas(std::vector<int>& arr);
```

5. Dado um vetor de inteiros `arr`, retorne um vetor com os elementos de `arr`, mas sem elementos repetidos. Os elementos no vetor de saída precisam estar na mesma ordem que os elementos do vetor de entrada.

```
#include <unordered_set>
#include <vector>

std::vector<int> semDuplicatas(std::vector<int>& arr);
```

6. Dado um vetor de strings `palavras`, retorne um vetor de vetores, onde cada vetor interno contém todas as palavras que são anagramas entre si. Por exemplo, se `palavras = {"ate", "eat", "tea", "bat", "tab"}`, então a saída deve ser `{{"ate", "eat", "tea"}, {"bat", "tab"}}`.

```
#include <vector>
#include <string>
#include <unordered_map>
#include <unordered_set>

std::vector<std::vector<std::string>> agruparAnagramas(std::vector<std::string>& palavras);
```

7. Dado um vetor `arr` não-ordenado de inteiros, retorne o comprimento da maior sequência de inteiros consecutivos em `arr`. Por exemplo, se `arr = {1, 9, 3, 10, 4, 20, 2}`, então a saída deve ser `4`, pois a maior sequência de inteiros consecutivos é `{1, 3, 4, 2}`. Note que os elementos desta sequência não precisam estar um ao lado do outro no vetor `arr`.

```
#include <unordered_set>
#include <vector>

int maiorSequenciaConsecutivos(std::vector<int>& arr);
```

8. Dado um vetor de inteiros `arr`, retorne `true` se existe pelo menos um subvetor de `arr` cuja soma dos elementos é igual a 0, ou `false` caso contrário.

```
#include <unordered_set>
#include <vector>

bool temSubvetorSomaZero(std::vector<int>& arr);
```

9. Dado um vetor de inteiros `arr`, retorne `true` se existe pelo menos um subvetor de `arr` cuja soma dos elementos é igual a um valor `k` dado, ou `false` caso contrário.

```
#include <unordered_set>
#include <vector>

bool temSubvetorSomaK(std::vector<int>& arr, int k);
```

10. Dado um par de vetores, `arr1` e `arr2`, retorne `true` se `arr1` e `arr2` contém os mesmos elementos, ou `false` caso contrário. Considere que os vetores não contém elementos duplicados e que não necessariamente os elementos estão na mesma ordem.

```
#include <unordered_set>
#include <vector>

bool saoIguais(std::vector<int>& arr1, std::vector<int>& arr2);
```

11. Dado um par de vetores `arr1` e `arr2`, retorne um vetor contendo os elementos que estão em `arr1` ou em `arr2`. Em outras palavras, sua função deve implementar a operação de união de conjuntos.

```
#include <unordered_set>
#include <vector>

std::vector<int> uniao(std::vector<int>& arr1, std::vector<int>& arr2);
```

12. Dado um par de vetores `arr1` e `arr2`, retorne um vetor contendo os elementos que estão em `arr1` e em `arr2`. Em outras palavras, sua função deve implementar a operação de interseção de conjuntos.

```
#include <unordered_set>
#include <vector>
```

```
std::vector<int> intersecao(std::vector<int>& arr1, std::vector<int>& arr2);
```

13. Dado um par de vetores `arr1` e `arr2`, retorne um vetor contendo os elementos que estão em `arr1` mas não estão em `arr2`. Em outras palavras, sua função deve implementar a operação de diferença de conjuntos.

```
#include <unordered_set>
#include <vector>

std::vector<int> diferenca(std::vector<int>& arr1, std::vector<int>& arr2);
```

14. Dado um par de vetores `arr1` e `arr2`, retorne `true` se `arr1` é um subconjunto de `arr2`, ou `false` caso contrário. Considere que os vetores não contém elementos duplicados. Por definição `arr1` é subconjunto de `arr2` se todos os elementos de `arr1` estão em `arr2`.

```
#include <unordered_set>
#include <vector>

bool ehSubconjunto(std::vector<int>& arr1, std::vector<int>& arr2);
```

15. Dado um par de vetores `arr1` e `arr2`, retorne `true` se `arr1` e `arr2` são disjuntos, ou `false` caso contrário. Considere que os vetores não contém elementos duplicados. Por definição `arr1` e `arr2` são disjuntos se não existe nenhum elemento em comum entre eles.

```
#include <unordered_set>
#include <vector>

bool saoDisjuntos(std::vector<int>& arr1, std::vector<int>& arr2);
```

16. Dado um par de vetores `arr1` e `arr2`, retorne um vetor contendo os elementos que estão em `arr1` ou em `arr2`, mas não em ambos. Em outras palavras, sua função deve implementar a operação de diferença simétrica de conjuntos.

```
#include <unordered_set>
#include <vector>

std::vector<int> diferencaSimetrica(std::vector<int>& arr1, std::vector<int>& arr2);
```

17. Dado um vetor de inteiros, retorne o elemento majoritário do vetor. O elemento majoritário do vetor é aquele que ocorre mais do que $\frac{n}{2}$ vezes, onde n é o tamanho do vetor. Se o vetor não contém um elemento majoritário, retorne `-1`.

```
#include <unordered_map>
#include <vector>

int elementoMajoritario(std::vector<int>& arr);
```

18. Dado um vetor de inteiros e um inteiro `k`, encontre o número de elementos distintos em cada janela de tamanho h. Por exemplo, se `arr = {1, 2, 1, 3, 4, 2, 3}` e `k = 4`, então a saída deve ser `{3, 4, 4, 3}`. Note que a primeira janela é `{1, 2, 1, 3}`, que contém 3 elementos distintos, a segunda janela é `{2, 1, 3, 4}`, que contém 4 elementos distintos, e assim por diante.

```
#include <unordered_map>
#include <vector>

std::vector<int> contarElementosJanela(std::vector<int>& arr, int k);
```

19. Dado um vetor de strings, crie um programa que conta a frequência de cada string. Além disto, a função recebe um dicionário (hash) contendo as palavras que não devem ser contadas (*stopwords*). Por

exemplo, se `arr = {"the", "day", "is", "sunny", "the", "the", "the", "sunny", "is", "is"}` e `stopwords = {"the", "is"}`, então a saída deve ser `{{"day", 1}, {"sunny", 2}}`.

```
#include <unordered_map>
#include <unordered_set>
#include <vector>
#include <utility>

std::unordered_map<std::string, int>
    contarFrequencia(std::vector<std::string>& arr, std::unordered_set<std::string>& stopwords);
```

20. Encontre o comprimento do maior subvetor de um vetor binário (que contém apenas 0 e 1) cujo número de 0s é igual ao número de 1s. Por exemplo, se `arr = {0, 0, 1, 0, 1, 0, 0, 1, 0, 0}`, então a saída deve ser `4`, pois o maior subvetor com o mesmo número de 0s e 1s é `{0, 1, 0, 1}`.

```
#include <unordered_map>
#include <vector>

int comprimentoMaiorSubvetorIgualZerosUns(std::vector<int>& arr);
```

21. Encontre o comprimento do maior subvetor de um vetor binário (que contém apenas 0 e 1) cuja soma dos elementos é igual a 0. Por exemplo, se `arr = {0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0}`, então a saída deve ser `6`, pois o maior subvetor com soma igual a 0 é `{0, 1, 0, 1, 0, 1}`.

```
#include <unordered_map>
#include <vector>

int comprimentoMaiorSubvetorSomaZero(std::vector<int>& arr);
```

22. Dado um vetor de strings, encontre as strings no vetor que podem ser formadas concatenando **exatamente duas outras** strings do mesmo vetor. Por exemplo, se `arr = {"cat", "dog", "catdog", "fish", "dogfish", "fishdog"}`, então a saída deve ser `{"catdog", "dogfish", "fishdog"}`.

```
#include <unordered_set>
#include <vector>

std::unordered_set<std::string>
    stringsConcatenadas(std::vector<std::string>& arr);
```

Exercícios Adicionais (Em Inglês)

1. UVA 10226 - Hardwood Species
2. UVA 10282 - Babelfish
3. UVA 11286 - Conformity
4. UVA 10420 - List of Conquests
5. UVA 11572 - Unique Snowflakes
6. UVA 11849 - CD
7. UVA 11991 - Easy Problem from Rujia Liu?

BONS ESTUDOS!