Aula de hoje:

1. Dúvidas do T1;
2. Dúvidas sobre os exercícios de lista encadeada (aula passada);
3. Desempenho dos algoritmos de lista encadeada e lista com arranjo;
4. Exercícios de revisão do conteúdo visto até agora.

IMPORTANTE: Fazer avaliação intermediária de disciplina!

1. Dúvidas do T1

Primeira entrega do T1:

Entrega individual:

- Questionário

Entrega em dupla:

- - Diagrama de Classe UML
  Ou
- - Especificar as Classes com seus atributos e assinaturas dos métodos

```
public class Acidente {
        private String nomeLogradouro
        private int feridos
        ...
        public Acidente (.....)
        public String getNomeLogradouro
        ....
```

```
}

public class LinkedListOfAcidente {
        private class Node {
                public Acidente acidente
                public Node next
        }
        private Node head
        private Node tail
        private Node count
        // métodos do “TAD Lista” conforme vimos em aula
}

public class StreetList {
        private class Node {
                        prevStret
                        nextStreet
                        String nomeLogradouro
                        LinkedListOfAcidente
      }
        Atributos...
        Métodos...
}
```

A leitura do arquivo está pronta. Considerem que vocês vão trabalhar com o array de Strings, e em cada posição do array tem um registro.

Lista

[“Av Ipiranga 1000”;1;2;0;nublado;norte” | Av Ipiranga 1000”;1;2;0;nublado;norte” | Av Ipiranga 1000”;1;2;0;nublado;norte”]

nomeLogradouro = “Ipiranga”

feridos = 1

3. Desempenho dos algoritmos de lista encadeada e lista com arranjo

Lista encadeada X Lista com arranjo

Em Java:

- LinkedList
- ArrayList

Duas formas de percorrer uma lista (em Java):

```
for(int i=0; i<lista.size(); i++)
        lista.get(i)

for(Integer elem : lista) // for-each
        elem
```

No caso do “for-each” é usado Iterator, que é a “ideia” de ter um atributo “current” e um método “next()”.

Para percorre uma **lista encadeada** do início ao fim de forma mais otimizada, é importante usar “for-each” ou ter um atributo “current” e um método “next()”.

Para o T1 é necessário ter este atributo “current” e métodos “prev()” e “next()” que permitirão percorrer a lista nos dois sentidos de forma mais otimizada.