# Lógica Computacional 117366
# Descrição do Projeto
# Formalização de Algoritmos para busca e ordenação

15 de junho de 2011

Prof. Mauricio Ayala-Rincón

O aluno Thiago Mendonça Ferreira Ramos (`thiagomendoncaferreiraramos@yahoo.com.br`) dará suporte aos alunos no desenvolvimento do projeto. Laboratórios do LINFO (6) tem instalado o software necessário (PVS com as livrarias PVS da NASA).

## 1 Introdução

Algoritmos de busca e ordenação são fundamentais em ciência da computação. Busca é um mecanismo essencial em estruturas de dados e ordenação é relevante para diminuição do tempo na busca em diversas estruturas de dados. Neste projeto considerar-se-ão algoritmos de busca e ordenção sobre o tipo abstrato de dados `list` como especificado no assistente de demonstração PVS.

O objetivo do projeto da disciplina é introduzir os mecanismos básicos de manuseio de tecnologias de verificação e formalização que utilizam técnicas dedutivas lógicas, como as estudadas na disciplina, para garantir que objetos computacionais são lógicamente corretos.

## 2 Descrição do Projeto

Com base na *teoria* PVS `sorting` (arquivos de especificação e prova `sorting.pvs` e `sorting.prf`) disponível na página da disciplina, especificada na linguagem do assistente de demonstração PVS (`pvs.csl.sri.com`) executável em plataformas Unix/Linux os alunos deverão formalizar propriedades de especificações para busca e ordenação em listas de naturais.

### 2.1 Busca em listas de naturais

Para a especificação de busca em listas:

```
search(i : nat, l : list[nat]) : RECURSIVE nat =
  IF null?(l) THEN length(l)
  ELSIF car(l) = i THEN 0
     ELSE 1 + search(i, cdr(l)) ENDIF
MEASURE length(l)
```

a teoria inclui demonstrações de propriedades da função `search` como p. ex., o fato de retornar o comprimento da lista no caso no qual o valor buscado não ocorre na lista e no caso no qual este occorre, efetivamente retornar um índice da lista de entrada no qual o valor buscado ocorre, respectivamente:

```
not_in_l_gives_lenght_l : LEMMA
FORALL(l : list[nat], i: nat) :
  (FORALL( k : below[l`length]):
```

```
      NOT nth(l, k) = i)  =>
   search(i, l) = l`length

search_works : LEMMA
FORALL (l : list[nat], k : nat) :
  member(k, l) => nth(l, search(k, l)) = k
```

Concretamente para a função `search` devem ser formalizadas as seguintes propriedades:

**Questão 01** *Para toda lista e valor buscado o índice que a função* `search` *retorna é o menor índice no qual o valor buscado ocorre na lista*

```
search_min_index : CONJECTURE
FORALL (l : list[nat], k : nat) :
   FORALL (m : below[l`length]) : nth(l, m) = k =>
          search(k, l) <= m
```

**Questão 02** *O índice obtido na busca sobre o reverso da lista é o maior índice no qual o valor ocorre*

```
search_in_rev_max_index  : CONJECTURE
FORALL (l : list[nat], k : nat) :
   FORALL (m : below[l`length]) : nth(l, m) = k =>
          (length(l) - 1) - search(k, reverse(l)) >= m
```

## 2.2 Ordenação de listas de naturais

Para a função de ordenação de listas especificada no estilo de *maxsort* deverá ser formalizado o fato da lista resultante estar ordenada.

A ordenação é especificada utilizando uma função auxiliar que recursivamente intercâmbia a chave mínima da lista até deixa-la no fim da lista:

```
switching_min(l : list[nat] | NOT null?(l)) : RECURSIVE list[nat] =
 IF l`length < 2 THEN l
 ELSIF car(l) < car(cdr(l)) THEN
    cons(car(cdr(l)), switching_min(cons(car(l), cdr(cdr(l)))))
 ELSE cons(car(l), switching_min(cdr(l)))
 ENDIF
MEASURE length(l)
```

Lemas adicionais sobre a correção desta função serão necessários.

A função de busca, recursivamente, utiliza a função de intercâmbio para detetar o mínimo, que é colocado no início da lista revertendo-a. O processo recursivo continua sobre o resto do reverso da lista excluindo o elemento mínimo.

```
sorting_min(l : list[nat] | NOT null?(l)) : RECURSIVE list[nat] =
 IF l`length < 2 THEN l
 ELSE LET rev_sw_min = reverse(switching_min(l)) IN
       cons(car(rev_sw_min), sorting_min(cdr(rev_sw_min)))
 ENDIF
MEASURE length(l)
```

Lemas adicionais sobre a correção desta função serão precisos, sendo a correção (parcial) sintetizada na seguinte conjectura.
**Questão 03** *A lista gerada pela função* `sorting_min` *é uma lista ordenada não decrescentemente*

```
sorting_min_works  : CONJECTURE
 FORALL (l : list[nat], k : below[l`length]) :
    0 <= k AND k <= length(l) - 2 => nth(l, k) <= nth(l, k+1)
```

# 3 Etapas do desenvolvimento do projeto

Os alunos deverão definir grupos de trabalho limitados a **quatro** membros. O projeto será dividido em duas etapas como segue:

- A primeira etapa do projeto e a de Verificação das Formalizações. Os grupos deverão ter prontas as suas formalizações na linguagem do assistente de demonstração PVS o dia **04.07.2011**. Na semana de **04-07.07.2011**, durante os dias de aula, realizar-se-á a verificação do trabalho para a qual os grupos deverão, em acordo com o monitor e professor, determinar um horário (de uma hora) no qual todos membros do grupo deverão comparecer.
  **Avaliação (peso 6.0)**:
  - Um dos membros, selecionado por sorteio, explicará os detalhes da formalização em máximo 20 minutos.
  - Os quatro membros do grupo poderão complementar a explicação inicial em máximo 10 minutos.
  - A formalização será testada nos seguintes 30 minutos.
- A segunda etapa do projeto consiste da apresentação dos resultados finais e conclusões do estudo do problema.
  **Avaliação (peso 4.0)**: Cada grupo de trabalho devera entregar um Relatório Final inédito, editado em Latex, limitado a oito páginas (12 pts, A4, espaçamento simples) do projeto até o dia **07.07.2011** com o seguinte conteúdo:
  - Introdução e contextualização do problema.
  - Explicação da soluções.
  - Especificação do problema e explicação do método de solução.
  - Descrição da formalização.
  - Conclusões.
  - Referências.