Ciência da Computação e Informática

Plágio

O uso de software copiado no Departamento de Ciência da Computação

Adotado pelo Subdepartamento de Ciência da Computação, 27 de fevereiro de 1997

Plágio envolve pegar ideias ou outras formas de esforço intelectual e passá-las como se fossem suas [Guralnik, 68]. Um exemplo específico de plágio é usar um programa encontrado em um livro didático como *solução* para uma tarefa de programação, ou seja, sem mencionar o autor original e a fonte do programa.

A questão do plágio é particularmente preocupante no departamento de Ciência da Computação, devido à importância atribuída à criação de programas de computador na maioria dos cursos. De acordo com os regulamentos da Universidade, as sanções para alunos que praticam plágio podem ser bastante severas. Essas sanções podem incluir a reprovação do aluno e, nos casos mais graves, a expulsão do aluno da Universidade. Como a maioria das definições e diretrizes relacionadas ao plágio se referem a trabalhos escritos e falados, os docentes do Departamento de Ciência da Computação acreditaram que seria útil interpretar algumas dessas diretrizes no contexto da programação de computadores.

O Código de Ética Estudantil da Universidade de Indiana fornece a seguinte definição de plágio como parte de sua discussão sobre má conduta acadêmica [IU, 93, p17]:

O aluno não deve adotar ou reproduzir ideias, palavras ou declarações de outra pessoa sem o devido reconhecimento. O aluno deve dar o devido crédito à originalidade dos outros e reconhecer a sua dívida sempre que fizer qualquer um dos seguintes:

- Cita palavras reais de outra pessoa, orais ou escritas
- Parafraseia as palavras de outra pessoa, orais ou escritas
- Usa a ideia, opinião ou teoria de outra pessoa

- Toma emprestados fatos, estatísticas ou outro material ilustrativo, a menos que a informação seja de conhecimento comum.

Podemos interpretar o exposto acima como significando que, ao usar programas e fragmentos de programas de terceiros, ou ao usar programas baseados em ideias, teorias ou algoritmos de terceiros, deve-se dar o devido crédito ao criador desses esforços intelectuais.

O restante desta nota discutirá *como* o devido crédito deve ser dado, que tipos de coisas é permitido copiar (com uma breve discussão sobre a lei de direitos autorais) e quando pode haver um conflito entre copiar o código de outra pessoa e os objetivos educacionais de uma tarefa.

Como é dado o devido crédito?

O devido crédito é dado indicando claramente no código-fonte:

1. O autor original do código ou algoritmo.
2. A fonte de onde este código ou algoritmo foi obtido. Se a fonte for um livro didático, forneça uma citação bibliográfica completa, incluindo o nome do texto, o autor, a editora, a edição (se não for a primeira), a data e o número da página do algoritmo/código. Se a fonte for outro aluno ou o instrutor, o nome do programador original deve ser fornecido.
3. Uma descrição de quaisquer alterações feitas no código ou algoritmo pelo programador atual.

Em geral, um programador deve documentar seu código de forma a indicar claramente quais partes do programa foram escritas por ele e quais foram escritas por outros. Além disso, é costume fornecer documentação indicando se o programador está utilizando código *próprio* , obtido de um programa previamente escrito. Dessa forma, o leitor pode distinguir entre trabalho recém-criado e reutilizado. O Apêndice fornece diversos programas de exemplo que ilustram o uso dessas diretrizes de citação.

Cópia e objetivos educacionais

Em geral, cada tarefa de um curso tem um conjunto de objetivos educacionais associados. Atualmente, existem apenas casos raros em que o objetivo é localizar e combinar os trabalhos de outros, adicionando pouco ou nada de novo. A maioria das tarefas visa exigir uma grande dose de expressão inovadora por parte do aluno.

As implicações práticas disto são que tanto o aluno quanto o instrutor têm uma responsabilidade compartilhada, embora diferente, em garantir que os objetivos da tarefa sejam cumpridos e em determinar o que e quanto código pode ser copiado para ainda atender aos objetivos do curso. A responsabilidade do instrutor é deixar os objetivos claros, bem como fornecer diretrizes sobre os pedaços de código que podem, e possivelmente *devem* , ser copiados de outros. O instrutor também é responsável por monitorar a cópia e citação feitas pelos alunos. A responsabilidade do aluno é interpretar os objetivos e diretrizes fornecidos pelo instrutor, buscar ativamente

esclarecimentos, se necessário, se envolver e encorajar outros a se envolverem em comportamento ético referente à cópia (incluindo o instrutor) e se automonitorar suas ações a esse respeito.

Como orientação geral, antes de incluir o trabalho de outra pessoa na tarefa de programação, é prudente primeiro verificar com o instrutor.

Aqui estão dois exemplos em que a cópia desempenha um papel radicalmente diferente em relação aos objetivos educacionais de uma tarefa. No primeiro caso, não é apropriado, enquanto no segundo, é apropriado.

1. Uma tarefa é dada na qual você deve escrever uma sub-rotina que recebe um `int` como parâmetro e gera a codificação binária desse parâmetro. Suponha que você encontre uma solução para este problema em um livro didático da biblioteca. Você deve usar essa solução, citando-a corretamente? Embora não possamos ter certeza sobre o objetivo específico desta tarefa, na ausência de informações em contrário, podemos presumir que o instrutor deseja que você desenvolva a função sozinho, com base em sua compreensão de representações numéricas. Portanto, usar o programa do livro didático violaria os objetivos educacionais da tarefa.
2. Uma tarefa é dada na qual você deve desenvolver uma classe `de grafos` que forneça operações como e . Suponha que, ao realizar esta tarefa, você queira reutilizar o código que desenvolveu na classe de estruturas de dados para listas encadeadas e tabelas de hash. É justo presumir que os objetivos desta tarefa não *incluem* o desenvolvimento de implementações de listas encadeadas e tabelas de hash; em vez disso, os objetivos são que você use estruturas de dados conhecidas para construir um programa maior e mais complexo. Portanto, neste exemplo, copiar o código escrito anteriormente está dentro dos objetivos educacionais. `print_nodes` `current_edge_value`

Permissões e direitos autorais

Uma questão adicional diz respeito à *permissão* , explícita ou implícita, de um indivíduo para usar o código ou as ideias de outra pessoa. A discussão a seguir pressupõe que o código em discussão esteja publicado em um livro didático ou outro formato impresso, ou seja publicamente legível online via protocolo FTP ou HTTP.

Podemos usar como guia as leis de direitos autorais existentes nos Estados Unidos. De acordo com a lei de direitos autorais, *as ideias* em si não são passíveis de direitos autorais e, portanto, qualquer pessoa pode usar livremente as ideias de outra pessoa. Como indicado acima, no ambiente acadêmico, o uso das ideias de outra pessoa deve sempre incluir uma citação.

Para o código obtido do instrutor em uma tarefa, a permissão é implicitamente dada aos alunos para usar esse código e, de fato, o uso do código do instrutor geralmente é necessário.

Código obtido da internet ou publicado em textos pode ter permissões explícitas, permitindo que outra pessoa o copie e use, embora possa haver restrições. Por exemplo, a Free Software Foundation, que distribui o compilador GNU C++ usado nos computadores dos departamentos, fornece livre acesso ao seu código-fonte sob o que eles chamam de acordo *copyleft* , que prevê cópias significativas, desde que a distribuição subsequente do código copiado forneça os devidos créditos e não imponha outras restrições à cópia. É muito mais comum que o código não tenha permissões ou restrições explícitas, ou seja protegido por direitos autorais. Como regra geral, trate o código sem permissões explícitas como código protegido por direitos autorais.

A lei de direitos autorais estabelece que somente o detentor dos direitos autorais (por exemplo, o autor do software, a empresa de software, o editor do texto) tem permissão para fazer cópias do material protegido por direitos autorais. No entanto, há uma disposição nesta lei que permite que indivíduos façam suas próprias cópias do trabalho de outrem sob certas circunstâncias. Esta disposição é chamada de *Estatuto do Uso Justo* . Grande parte da discussão a seguir é uma paráfrase de trechos do artigo " *Uso Justo: Visão Geral e Significado para o Ensino Superior"* , de Kenneth Crews, Professor Associado da IUPUI e Diretor do Centro de Gestão de Direitos Autorais. Este artigo pode ser obtido em

```
http://www.iupui.edu/it/copyinfo/highered.html
```

A cópia de código (ou qualquer outro material protegido por direitos autorais) sob o Estatuto de Uso Justo requer a consideração de quatro fatores:

Propósito:

Qual é a finalidade da cópia? Em geral, a cópia para fins educacionais e de pesquisa privada tem sido protegida pelos tribunais.

Natureza:

Qual é a natureza da obra protegida por direitos autorais? Os tribunais têm se mostrado favoráveis à cópia de obras de não ficção atualmente impressas, o que se aplica à maioria dos códigos que os alunos desejam copiar.

Quantia:

Quanto da obra é copiado? Em geral, a cópia de pequenas quantidades de obras impressas (em relação ao tamanho da obra como um todo) tem proteção judicial.

Efeito de mercado:

Qual será o impacto do trabalho no mercado? Crews escreve

Este fator significa fundamentalmente que se você fizer um uso para o qual uma compra de um original teoricamente deveria ter ocorrido — independentemente de sua vontade pessoal ou capacidade de pagar por tal compra — então este fator pode pesar contra o uso justo. ... Se o seu propósito for pesquisa ou bolsa de estudos, o efeito de mercado pode ser difícil de provar. ... Cotações ocasionais ou fotocópias podem não ter efeitos adversos no mercado, mas reproduções de software e fitas de vídeo podem fazer incursões diretas nos mercados potenciais para essas obras.

Em resumo, na maioria dos casos de uso de software protegido por direitos autorais por alunos em trabalhos de programação, todas essas condições são atendidas e o aluno pode fazer as cópias. No entanto, em casos em que o código pode ser posteriormente comercializado ou em que ocorra cópia substancial, deve-se ter mais cautela.

Informações adicionais sobre direitos autorais, especialmente no que se refere ao ensino superior, podem ser encontradas no URL fornecido pelo Centro de Gerenciamento de Direitos Autorais da IUPUI:

```
http://www.iupui.edu/it/copyinfo/home.html
```

Resumo

O uso de código copiado em trabalhos de programação deve ser claramente documentado no código, de modo a permitir que o leitor identifique quem escreveu cada parte do código, e deve ser feito dentro dos limites dos objetivos educacionais do trabalho de programação. Caso tenha alguma dúvida sobre esta política, não hesite em discuti-la com qualquer um dos docentes.

Referências

Guralnik, 68

David Guralnik, Editor. Novo dicionário mundial da língua americana de Webster. The World Press Publishing Company, 1968.

UI, 93

Código de Ética Estudantil da Universidade de Indiana, 1993

Apêndice

1. Um aluno é solicitado a escrever uma classe em C++ que implemente uma lista encadeada. Grande parte da implementação foi retirada de um livro didático de estruturas de dados. Veja como a citação pode ser:

```
    // int_list.cc

    // Classe: Lista de inteiros
    //
    // Programador: Jake deStudent
    // Data de criação: 3 de dezembro de 94
    // Última modificação: 16 de março de 95
    //
    // Adaptado da classe de lista do texto
    // _Estruturas de Dados e Análise de Algoritmos em C++_
    // por Mark Weiss, publicado por Benjamin Cummings, 1994, pp. 62-71.
    //
    // Alterações: Construtor de cópia adicionado, #define INT_LIST, jdt
16/03/95
    // Todo o outro código usado literalmente de Weiss
    .
    .
    .
    /* ========
         Ação: Retorna se a lista está vazia ou não
         Programador: Mark Allen Weiss
         Fonte: _Estruturas de Dados e Análise de Algoritmos em C++_
                       publicado por Benjamin Cummings, 1994, p.62
         Adaptação: Alterado do procedimento inline pelo jdt para ocultar
informações.
         Parâmetros: Nenhum passado ou retirado
         Observações: Assume um nó de cabeçalho fictício
    */
    int_List::Is_Empty()
    {
      retornar (List_Head -> Próximo == NULL);
    }
```

2. Um instrutor fornece aos alunos um programa principal e pede que escrevam diversas funções que são chamadas por main. Ao entregar a tarefa de casa, cada aluno deve citar o instrutor e a si mesmo como programadores:

```
    // lição_de_casa_4.cc

    //
    // Programadora: Sally Estudante e Jane Instrutora
    // Data de criação: 5 de fevereiro de 97
    //
    // Este programa recebe como entrada ...
    // { Restante da descrição do programa }

    //
    // O programa principal foi escrito por Jane Instructor e distribuído
para
    // os alunos em 22 de janeiro de 97 na descrição da Tarefa de casa 4
para
    // C201. As funções prompt_for_input e calc_payment foram escritas
por
    // Sally Student. A função display_payments foi escrita por Jane
    // Instrutor.
```

Para indicar claramente a autoria, às vezes é necessário documentar quem criou cada função individual, de maneira semelhante à seguinte, além de documentar os diferentes programadores no cabeçalho do programa. Isso é particularmente comum em projetos grandes criados por vários programadores diferentes.

```
    // Programadora: Sally Student
    // {Descrição da função e outras informações do cabeçalho}
    void prompt_for_input (double saldo_inicial, int número_de_anos,
                           taxa de juros dupla)
    {
       /* corpo da função */
    }

    // Programadora: Jane Instructor
    // Fonte: Descrição no folheto da Tarefa de Casa 4
    // Adaptações: Nenhuma
    // {Descrição da função e outras informações do cabeçalho}
    void display_payments (double initial_balance, int número_de_anos,
                           taxa de juros dupla)
    {
       /* corpo da função */
    }
```

3. Você quer escrever um programa de jogo da velha para Windows para seu próprio prazer. Você encontra uma solução bacana na internet, escrita em Java por alguém, que você adapta para C++ e X-Windows. Isso reforçará a prática profissional de citar as fontes de onde você obteve o código e documentar as alterações feitas.

```
    // jogo da velha.cc

    // Programadores: Joe Student e Jose Javaguru
    //
    // Este programa é uma adaptação de Joe Student de um programa Java
    // escrito por Jose Javaguru. O programa Java foi obtido do
    // site http://www.javajunkie.joe/~javaguru/waycool/tictactoe
    //
    // O desenho do programa principal e das principais sub-rotinas foram
tomados
    // do código do Javaguru. A portabilidade do Java para C++/X-windows
foi feita
    // por Joe Student. Devido a esta porta, as funções display_board,
    // input_move e new_board tiveram que ser completamente reescritos.
    //
    // { Mais detalhes descrevendo as mudanças que precisavam ser feitas
}
    //
```

4. Você precisa escrever um programa que recebe como entrada um conjunto de pares de disciplinas, onde um par **<C1,C2>** indica que a disciplina **C1** é um pré-requisito da disciplina **C2** , e informa se há um *ciclo* nos pré-requisitos para qualquer conjunto de disciplinas, ou seja, se alguma disciplina é um pré-requisito para si mesma. Você escreve um programa baseado no algoritmo *de ordenação topológica* descrito em um de seus livros didáticos, mas para o qual o código não é explicitamente apresentado.

```
// course_cycle.cc
//
// Programador: Chris van Student
// Data de criação: 9 de abril de 96
// Última modificação: 16 de abril de 96

// {Descrição do problema geral}
//
// A sub-rotina central deste programa é check_for_cycle,
// que é uma implementação do algoritmo de classificação topológica
// descrito no texto
// _Fundamentos da Ciência da Computação, edição C_
// por Alfred Aho e Jeffrey Ullman, publicado pela Computer Science
// Imprensa, 1995, pp. 497-498.
```

5. Adapte o código de outra pessoa que contém citações para o código de uma *segunda* (ou terceira, ou quarta...) pessoa. Nesse caso, você deve preservar as citações originais no seu código, adicionando comentários que indiquem quais adições específicas são suas. Às vezes, isso precisa ser feito linha por linha, se diferentes partes de uma função forem escritas por pessoas diferentes ao longo do tempo. Por exemplo, aqui está um fragmento das citações do código entregue por um aluno do curso de Compiladores C431 (adaptado excluindo vários comentários e adicionando citações para o programador original):

```
    /*
    semacts.c

    Este arquivo contém as rotinas de ação semântica para o MACRO
    compilador.

    Programador original:
                            Fornecido pelo editor (Benjamin Cummings) para
                            acompanhar o texto _Criando um Compilador com
C_ por
                            Fischer e Leblanc, 1991. Nenhum programador
                            identificado. Também pode ser obtido via FTP
anônimo
                            de kaese.cs.wisc.edu.

    15/08/92 Josh Tenenberg - Comentou sobre o assunto #ifdef TURBOC
    25/10/94 Byron Miller, - Adicionado IntConstType ao Semantic
                               declarações e rotinas
    25/10/94 Byron Miller, - Adicionado IntConstRec às declarações
semânticas
                               e rotinas
    25/03/96 Robert Wiseman - Adicionada rotina semântica convert_type
    25/03/96 Robert Wiseman - Rotina semântica process_ids adicionada
    25/03/96 Robert Wiseman - Adicionados tipos predefinidos integer e
!error para
                               Rotina semântica InitializeSemActions
    */
```

6. Você mesmo desenvolve o código, sem usar código de fontes externas. Nesse caso, liste-se como o único autor. O exemplo a seguir foi escrito na linguagem de programação Scheme, e as linhas que começam com `;' indicam comentários.

```
    ;;; Autor: Josh Tenenberg
    ;;; Data de criação: 29/10/96
    ;;;
    ;;; Ação: Recebe um problema como entrada e executa cada instância
    (defina problema de execução
      (lambda (provavelmente)
        (writeln "*** Função de teste " (problema->print-label prob) "
***")
        (nova linha) (nova linha)
        (para cada
         (lambda (inst)
           (executar-instância inst (problema->thunk prob) (problema-
>comp prob)))
         (problema->instâncias prob))))
```

7. Você desenvolve código a partir de um algoritmo de conhecimento comum em ciência da computação. Por exemplo, suponha que um programa que você está escrevendo precise de uma função para encontrar um elemento mínimo em uma árvore de busca binária. Nesse caso, use a seguinte regra prática sobre quem citar. Se você desenvolver o código sozinho, não precisará citar nenhuma outra fonte. Se usar um ou mais textos ou outras fontes de código como referência, cite-os. Aqui está o código que desenvolvi para este problema de encontrar o mínimo no Scheme, fazendo referência ao texto que usei em seu desenvolvimento:

```
;;; Autor: Josh Tenenberg
;;; Data de criação: 22/09/96
;;; Algoritmo referenciado do texto:
;;; Autor: Mark Allen Weiss
;;; Fonte: _Estruturas de Dados e Análise de Algoritmos em C++_
;;; publicado por Benjamin Cummings, 1994, p.131
;;; Adaptação: alterado de C++ para esquema
;;;
;;; Ação: Encontre um elemento mínimo em um BST

    (defina find-min
      (lambda (árvore)
        (cond
          ((árvore-vazia? árvore) '())
          ((árvore vazia? (árvore da subárvore esquerda)) (árvore raiz))
          (else (find-min (esquerda-subárvore))))))
```