UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
NÚCLEO DE COMPUTAÇÃO - NUCOMP

ALGORITMOS E ESTRUTURAS DE DADOS I – 2018/1
PROF. FLÁVIO JOSÉ MENDES COELHO

PROJETO PRÁTICO 2 - versão 2

1 Objetivos

Este **Projeto Prático 2** – **PP2**, tem o objetivo de exercitar e avaliar suas habilidades em:

- Codificar um ou mais dos tipos abstratos de dados LISTAS, PILHAS, FILAS, TABELAS HASH, ÁRVORES N-ÁRIAS, ÁRVORES BINÁRIA DE BUSCA, ÁRVORES AVL e ÁRVORES RED-BLACK (com possíveis combinações entre eles) na linguagem de programação exigida neste enunciado, e solucionar problemas empregando estas estruturas de dados.
- Desenvolver código de qualidade com boas práticas de programação: boa indentação, boa nomeação, refinamentos sucessivos, POO (exceto herança), programação genérica, etc.
- Explicar com segurança e lógica suas decisões ao escrever o código do projeto, respondendo de forma coerente às perguntas do professor.
- Aprender a lidar com desenvolvimento de software em equipe.

2 Descrição do problema

Interpretador da Linguagem Kinojo

O iluminado prof. Kaninchen criou a mini linguagem de programação **Kinojo** com o intuito de programar caramujos. Sim, caramujos[1]! Fatos recentes revelam que os Sábios Monges Ninja-Veganos da Sagrada Montanha da Ordem Polinomial do Norte decobriram que esses bichinhos são pequenos computadores aliens deixados na Terra há milhares de anos! E, pasmem: os caramujos conseguem processar pequenos programas de manipulação de tabelas hash! Mas, com que fim os caramujos-aliens processariam tabelas hash? Só Deus e eles próprios o sabem. Mas, o inenarrável prof. Kaninchen acredita haver algo muito importante e sinistro por trás destes

[1] **Eca! Que nojo!**

caramujos-aliens, e precisa descobrir como botar esses bichinhos para processar tabelas hash! Veja, abaixo, um exemplo de um programa na linguagem criada pelo incompilável professor:

```
1. TH alfa = 6 asda3s1das qw0eq1 srt12qf tyutyuty 2f45gdf8gdfg rt4rut5565i .
2. TH beta = 7 vcbaqqq posd pavn6a qweqweqe tyutyuty uiouiuo cv12cv1bcvb.
3. TH gama = 5 poduvnahs8 oqpfhaixxzzi aiurqlm330 k31238jkkmasoo o992o3iiap .
4. UNION uni = alfa beta .
5. PRINT uni KEY asda3s1das .
6. INTER intersec = beta gama .
7. MINUS m = intersec uni .
8. PRINT m KEY pavn6a .
9. FIM
```

Está é a linguagem de programação Kinojo! Veja como é simples de entender Kinojo: na linha 1 o comando Kinojo `TH` está criando uma tabela hash chamada `alfa`. Após o operador `=` vem o número de chaves a serem inseridas na tabela, seguido de uma sequência de uma ou mais chaves. Todo comando Kinojo termina com um símbolo de ponto, com exceção do comando `FIM`. Nas linhas 2 e 3 ocorrem a criação das tabelas hash `beta` e `gama`, respectivamente, juntamente com suas chaves. Na linha 4 o comando Kinojo `UNION` processa a união das tabelas hash `alfa` e `beta` e armazena a tabela hash resultante na variável `uni`. Na linha 5, o comando `PRINT` imprime as chaves colididas (em ordem crescente) acessíveis pela chave `asda3s1das` na tabela hash `uni`. Veja que a palavra `KEY` precede a chave de busca `asda3s1das`. Na linha 6, o comando `INTER` processa a interseção entre as tabelas hash `beta` e `gama` e coloca a tabela resultante na variável `intersec`. Em seguida, na linha 7, o comando `MINUS` processa a diferença entre as tabelas `intersec` e `uni`, isto é, subtrai `uni` de `intersec`, e armazena a tabela resultante em `m`. Na linha 8 o comando `PRINT` imprime da mesma forma como explicado na linha 5, e o programa termina sua execução com o comando `FIM`, na linha 9. Os comando de união, interseção e subtração têm a mesma semântica destas operações aplicadas em teoria dos conjuntos. Nas seção seguinte, será descrito como estas operações devem ser processadas para as tabelas hash Kinojo.

Para ajudar o prof. Kaninchen, você e sua equipe decidiram programar um interpretador Kinojo. O seu programa receberá como entrada um programa Kinojo e processará as saídas corretas deste programa.

3 Elementos do interpretador Kinojo

Os seguintes elementos fazem parte do interpretador a ser implementado:

1. As tabelas hash Kinojo devem utilizar árvores AVL para resolver colisões, em lugar de listas encadeadas. Que nojo!
2. O tamanho M de uma tabela hash será determinado em função do número de chaves que a tabela armazenar. O cálculo para determinar M será: o maior primo menor que $N/4$, onde N é o número de chaves a serem armazenadas na tabela.

   A Figura 1, ilustra uma tabela hash Kinojo. Veja que o vetor $T = [T_0, T_1, \ldots, T_{m-1}]$ armazena as entradas da tabela hash Kinojo, sendo cada entrada $T_0, T_1, \ldots, T_{m-1}$ um

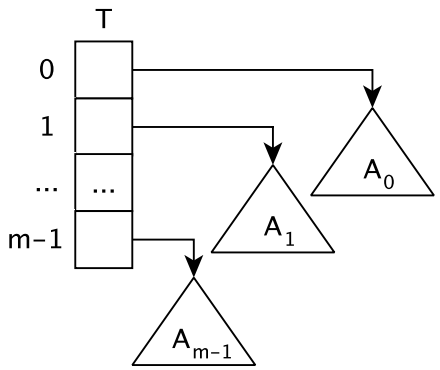

Figura 1: Tabela hash Kinojo: usando árvores AVL para resolver colisões.

ponteiro para uma árvore AVL $A_0, A_1, \ldots, A_{m-1}$, correspondente.

3. Será preciso armazenar em memória as tabelas hash de um programa Kinojo, pois cada linha de um programa Kinojo faz alguma operação sobre estas tabelas. Você tem a opção de:

   (a) Armazenar cada tabela hash Kinojo em uma AVL.

   (b) Armazenar cada tabela hash Kinojo em uma tabela hash (usando listas encadeadas ou AVLs para resolver colisões; você decide).

   (c) Armazenar cada tabela hash em uma lista encadeada.

   Importante 1: cada tabela hash Kinojo a ser armazenada terá um nome (sua chave) que é o mesmo nome da variável que a representa no programa Kinojo.

   Importante 2: as opções de armazenamento (a) e (b), acima, renderão maior pontuação na inspeção do código; a opção (c) renderá uma pontuação menor por ser mais simples e menos eficiente.

4. **Variáveis Kinojo**. São uma sequência de letras minúsculas e dígitos, iniciada por uma letra minúsculas. Armazenam as tabelas hash Kinojo e são sua chave de identificação. Se definidas com o comando `TH` não podem ser redefinidas por nenhum outro comando, podendo ser usadas como operandos (parâmetros) dos demais comandos (com execeção do `FIM`). Se definidas com um comando `UNION`, `INTER` ou `MINUS` não podem ser redefinidas com estes comandos ou com o `TH`, podendo ser usadas como operandos (parâmetros) dos demais comandos (com execeção do `TH` e `FIM`).

5. **O comando** `TH`. Cria ou define uma tabela hash Kinojo juntamente com suas chaves e um nome (que é o nome da variável Kinojo associada à tabela criada).

   Para as operações abaixo, considere que R, P e Q são tabelas hash Kinojo e que $|R|, |P|$ e $|Q|$ são os tamanhos de R, P e Q, respectivamente, considerando como tamanho o nú-

mero de células do vetor de cada tabela. Considere, ainda, que uma tabela hash Kinojo vazia $\emptyset$ possui tamanho $|\emptyset| = 0$.

6. **O comando** `UNION`. A união entre P e Q não vazias resultará em uma tabela hash Kinojo R contendo todas as chaves que estão na tabela P ou na Q (ou inclusivo), sem repetição de chaves. Além disso: (a) P `UNION` P resulta em P; (b) P `UNION` $\emptyset$ resulta em P (P não vazia); e (c) $\emptyset$ `UNION` Q resulta em Q (Q não vazia). Sendo P e Q não vazias, o tamanho de R será o maior primo menor ou igual a $|P| + |Q|$.

7. **O comando** `INTER`. A interseção entre P e Q não vazias resultará em uma tabela hash Kinojo R contendo todos as chaves que estão na tabela P e também na tabela Q. O tamanho de R deverá ser o tamanho da menor tabela entre P e Q, ou o tamanho de qualquer uma delas se ambas tiverem o mesmo tamanho. A tabela R será vazia se P e Q não vazias forem disjuntas (não possuirem uma ou mais chaves em comum), ou, se $P = \emptyset$ ou $Q = \emptyset$ (ou inclusivo).

8. **O comando** `MINUS`. A subtração entre as tabelas P e Q não vazias resultará em uma tabela hash Kinojo R contendo todos as chaves que estão na tabela P e não estão em Q ($R = P - Q$ ou $R = P\ Q$). O tamanho de R deverá ser o maior primo menor ou igual à $|P| - |Q|$. No caso de $|P| - |Q| < 2$, deve ser mostrado na tela "Operação inválida". Além disso: (a) Se P e Q não vazias forem disjuntas (não possuirem uma ou mais chaves em comum), então P `MINUS` Q resulta em P; (b) P `MINUS` P resulta em $\emptyset$; (c) $\emptyset$ `MINUS` Q resulta em $\emptyset$ (Q não vazia); (d) P `MINUS` $\emptyset$ resulta em P (P não vazia).

9. **O comando** `PRINT`. Seja K uma chave de busca em uma tabela hash Kinojo T. O comando `PRINT` seleciona uma tabela Kinojo T e imprime as chaves colididas em ordem crescente, no endereço hash de K em T. Se a chave K não existir em T, o comando imprime "Chave K não encontrada em Tabela T". Se T estiver vazia, o comando imprime "Tabela T vazia".

4 Entradas e saídas

Cada entrada será um programa Kinojo, como descrito nos itens 2 e 3. Cada saída deverá ser apresentada corretamente conforme explicado sobre o comando `PRINT` descrito nos itens 2 e 3.

5 Requisitos do projeto

1. **Equipes**. Este projeto deve ser desenvolvido por uma equipe de **dois** estudantes. Não serão aceitas equipes com número menor ou maior de participantes, a não ser que isso seja permitido pelo professor, e discutido/decidido antes do início do projeto. Qualquer equipe com um número diferente de dois participantes (sem a justificativa explicada acima), terá seu projeto valendo no máximo 80% da pontuação total obtida.

2. **Ferramentas e técnicas**. O projeto deve ser codificado em C++14. Será permitido programação procedural, mas todos as estruturas de dados (TADS) principais deverão ser programadas orientadas a objeto, generalizadas (templates) e encapsuladas.

3. **Padrões de codificação**. Siga o *CamelCase* do Java como padrão de nomeação de identificadores. A indentação e posicionamento de chaves deve seguir o padrão K&R, variante de Stroustrup (`en.wikipedia.org/wiki/Indent_style#K.26R_style`). Veja as aulas iniciais de AED 1 sobre boas práticas de nomeação, etc.

4. **Compilador**. Indica-se o uso do compilador *GCC - the GNU Compiler Collection* (http://gcc.gnu.org), ou sua variante *Mingw* para para Microsoft Windows, ou ainda o compilador *clang* (https://clang.llvm.org). **Sempre teste seu projeto em vários compiladores online!** Às vezes, o projeto pode funcionar bem na sua máquina, mas pode apresentar erros em outros compiladores.

5. **Submissão**. Todo o projeto deverá ser submetido ao juiz online (a ser informado pelo professor) em um único arquivo fonte com extensão `cpp`.

6. **Bibliotecas e funções**. Sua equipe não deve utilizar nenhuma estrutura de dados pronta (listas, pilhas, filas, vectors, etc) da *Standard Template Library* - STL, ou de qualquer outra biblioteca C++. É suficiente o uso de `iostream` e `cstdlib` (para uso de qualquer outro arquivo de cabeçalho, fale com o professor). O uso de estruturas de dados prontas ou funções de bibliotecas conforme explicado acima, implicará na atribuição da nota mínima ao projeto.

6 Pontuação

O **PP1** vale no mínimo 0.0 e no máximo 10.0 (ou no máximo 8,0, no caso explicado na seção 6, item 1). As notas são distribuídas em duas partes:

(1) A **avaliação funcional** avalia o quanto foi implementado do que foi pedido, e se as saídas corretas são obtidas. Esta parte vale de 0.0 à 5.0 pontos. O código do projeto será submetido a um juiz online com N casos de testes (N será informado pelo professor). Cada caso de teste vale $5,0/N$.

(2) **Inspeção de código**. Esta parte vale de 0.0 à 5.0 pontos, e avalia os critérios especificados à seguir:

   (a) Segurança do aluno nas respostas às perguntas do professor sobre o código desenvolvido. Uso adequado de linguagem técnica nas explicações: nomes corretos de estruturas de dados, seus componentes e algoritmos, de elementos da linguagem de programação, e das técnicas de programação utilizadas.

   (b) Indentação de código correta, utilização de padrão de nomeação, nomes adequados para identificadores, etc.

   (c) Implementação correta/coerente dos tipos abstratos de dados, seguindo o que foi apresentado nas aulas, e neste enunciado.

   (d) Implementação orientada a objetos (com encapsulamento) das estruturas de dados.

   (e) Código genérico (templates) onde for aplicável e útil.

   (f) Codificação racional, simples, eficiente e criativa do código.

Se o projeto for implementado em outra linguagem de programação, ganhará integralmente a nota mínima.

7 Datas

- Emissão deste enunciado: 24/05/2018.
- Abertura do juiz online: 30/05/2018.
- Fechamento do juiz online: 10/06/2018 às 11h59min (hora local).
- Defesas: 11/06/2018 à 15/06/2018.

8 Contribuições

Agradeço aos alunos Francisco Elio Parente Arcos Filho (GAPA), Vitor Matheus de Souza Carvalho (GAPA, monitor) e Levi da Silva Lima (GAPA, monitor) que contribuíram com ideias iniciais para este Projeto Prático, além de ajudarem na elaboração das entradas e saídas do Juiz Online).

CÓDIGO DE ÉTICA

Este projeto é uma avaliação acadêmica e deve ser concebido, projetado, codificado e testado pela equipe, com base nas referências fornecidas neste enunciado ou nas aulas de Algoritmos e Estruturas de Dados, ou por outras referências indicadas pelo professor, ou com base em orientações do professor para com a equipe, por solicitação desta. Portanto, não copie código pronto da Internet para aplicá-lo diretamente a este projeto, não copie código de outras equipes, não forneça seu código para outras equipes, nem permita que terceiros produzam este projeto em seu lugar. Isto fere o código de ética desta disciplina e implica na atribuição da nota mínima ao trabalho.

Referências

[1] COELHO, Flávio. Slides das aulas de *Algoritmos e Esruturas de Dados I*. Disponível em https://sites.google.com/a/uea.edu.br/fcoelho/. Universidade do Estado do Amazonas, Escola Superior de Tecnologia, Núcleo de Computação - NUCOMP. Semestre letivo 2018/1.

[2] C++. In: *cppreference.com*, 2016. Disponível em <http://en.cppreference.com/w/>. Acesso em: 17 abr. 2016.

[3] CORMEN, T. H., Leiserson, C. E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 3rd edition, MIT Press, 2010

[4] KNUTH, Donal E. *Fundamental Algorithms*, 3rd.ed., (vol. 1 de The Art of Computer Programming), Addison-Wesley, 1997.

[5] KNUTH, Donal E. *Seminumerical Algorithms*, 3rd.ed., (vol. 2 de The Art of Computer Programming), Addison-Wesley, 1997.

[6] KNUTH, Donal E. *Sorting and Searching*, 2nd.ed., (vol. 3 de The Art of Computer Programming), Addison-Wesley, 1998.

[7] STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Edition, Addison-Wesley, 2013.

[8] STROUSTRUP, Bjarne. *A Tour of C++*. Addison-Wesley, 2014.

[9] SZWARCFITER, Jayme Luiz et. alii. *Estruturas de Dados e seus Algoritmos*. Rio de Janeiro. 2a. Ed. LTC, 1994.

[10] WIRTH, Niklaus. *Algoritmos e Estruturas de Dados*. Rio de Janeiro. 1a. Ed. Prentice - Hall do Brasil Ltda., 1989.

[11] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Java e C++*. 2a. Edição. Cengage Learning, 2010.

[12] ZIVIANI, Nívio. *Projeto de Algoritmos com Implementação em Pascal e C*. 3a. Ed. São Paulo: Cengage Learning, 2012.