Projeto I I
Operações sobre conjuntos (Sets)

SCC0202 – Algoritmos e Estruturas de Dados I

Prof. Rudinei Goularte

Data de Entrega: 09/12/2024

Este projeto deverá ser feito em grupos de 2 alunos. A entrega deve ser realizada no Tidia (veja abaixo).

Descrição do Problema

Conjunto é um conceito básico do ramo matemático da Teoria dos Conjuntos. Um Conjunto representa uma coleção de objetos, em que cada objeto é denominado **elemento**. A relação básica entre um elemento e o conjunto é a relação de pertinência, que determina se um elemento pertence ou não ao conjunto.

Conjuntos tem diversas aplicações em sistemas computacionais, estando presentes em várias linguagens de programação (como uma Estrutura de Dados), assim como em soluções de problemas matemáticos (puros e aplicados), de otimização e estatísticos.

Objetivo do Trabalho

Implementar o TAD **Conjunto** de modo que o usuário possa escolher entre 2 a Estruturas de Dados que implementam o conjunto. Ou seja, o TAD Conjunto deve ser cliente de 2 outros TADs, sendo que um será escolhido pelo usuário em tempo de execução. A escolha das Estruturas de Dados dos TADs de suporte ao Conjunto é livre. Entretanto, as estruturas escolhidas devem possibilitar que as operações **específicas** do Conjunto tenham a **menor complexidade computacional possível**. As operações consideradas são:

- 1) **Operações básicas** para TADs em geral: Criar (um conjunto), Apagar (um conjunto), Inserir (um elemento em um conjunto), Remover (um elemento de um conjunto) e Imprimir (imprimir os elementos armazenados no Conjunto).

- 2) **Operações específicas** de conjuntos: Pertence (um elemento está presente ou não no conjunto), União entre 2 Conjuntos, Intersecção entre 2 Conjuntos. Note que essas 2 útlimas operações recebem como entrada 2 conjuntos e devolvem um terceiro conjunto como resposta.

Entrega do Trabalho

1) Arquivo TXT: um breve relatório justificando as Estruturas de Dados escolhidas para representar Conjuntos e a complexidade computacional de tempo (Big Oh) de cada uma das operações (básicas e específicas). A complexidade deve ser apresenta como uma análise justificada* das operações e não apenas o Big Oh final. Por exemplo, suponha 2 algoritmos, um é O(1n) = O(n) e o outro é O(n) + O(n) = O(2n), que no fim é O(n). Nos casos em análise no projeto, em geral, a complexidade final de um algoritmo não muda (no exemplo ambos são O(n)), mas para n muito grande, o valor da constante pode fazer diferença (no exemplo O(1n) é melhor que O(2n)). *O justificada implica em analisar e explicar a composição das operações encontrando as complexidades parciais e montando uma equação que represente a complexidade.

2) Arquivo ZIP: Crie um arquivo Nome_Grupo.zip (Nome_Gupo deve ser substituído pelo nome dos componentes do grupo. Ex.: John_Mychell.zip) contendo: o arquivo TXT (relatório), o makefile do projeto e **todo código fonte** necessário para compilar e executar o programa, incluindo arquivos *.c* (*main* e outros), arquivos *.h e arquivos de teste*. **Não adicionar arquivos executáveis nem arquivos objeto.** Os arquivos de teste devem seguir o padrão 1.in, 1.out, 2.in, 2.out,… vejam exemplos abaixo.

3) O arquivo .ZIP deve ser entregue no Tidia (Escaninho) até 09/12/2024, 23:59h. Trabalhos submetidos em datas posteriores à especificada terão 1,0 ponto de desconto por dia de atraso. Apenas um integrante do grupo precisa submeter o trabalho no Escaninho.

Observações Importantes

- O projeto deve ser desenvolvido em grupos de 2 alunos.

- A formação dos grupos deve ser colocada no Tidia (Wiki) – do mesmo modo que no Projeto 1, até dia 30/10 às 23:59.

- A implementação deve ser realizada em linguagem C, padrão C99.

- Não serão forecidos arquivos de teste. Cada grupo será responsável por desenvolver seus prórpios testes e testar seu projeto antes da entrega. **Dica 1**: façam arquivos txt com entradas variadas e usem os "operadores" < e > no shell para carregar/criar os arquivos de testes. **Dica 2**: façam testes (mais de um!) para todas as operações. Um mesmo arquivo pode testar uma combinação de operações (criação+inserções+imprimir+apgar, p.e). Atenção especial aos testes das operações específicas. **Dica 3**: os PAEs/Monitores podem ajudar, ensinando a gerar os casos de teste. Consutem-nos!!!

- **Sobre o TAD Item:** não será necessário utilizar o TAD Item neste projeto. As structs que implementam as EDs devem ter um campo *int chave;* no lugar do Item.

Este campo será usado como a chave para as inserções, buscas, remoções e outras operações que necessitem.

- Os projetos serão compilados e corrigidos utilizando o compilador GCC Linux.
- **Haverá, obviamente, verificação de plágio. Aos projetos envolvidos em plágio será atribuída nota zero.**

Critérios de Avaliação

- Aderência à especificação e ao uso de TADs (4 pontos).
- Eficácia e eficiência da solução (4 pontos).
- Documentação interna (1 ponto).
- Relatório (1 ponto).

Exemplo de programa principal e interface (cliente/main.c e set.h)

- Consultem o Tidia: Repositório/Códigos/

Arquivo Exemplo_projeto2.zip

Exemplos de entradas e saídas:

Todos os arquivos de entrada e saída são arquivos texto. Os valores são separados po um espaço. Ao fim de cada linha exitem um <enter>. Lembrem-se que sistemas Linux representam <enter> de modo diferente de sistemas Windows.

Entrada 1: (1.in)

```
0
5
3
1 2 3 4 5
7 3 8
1
2
```

Saída 1: (1.out)

```
Pertence.
```

Obs.: o pertencimento é verificado sempre no SET A.

Obs2.: caso não pertencesse, a saída seria: Nao pertence.

Entrada 2: (2.in)

```
1
5
3
```

1 2 3 4 5
7 2 8
2

Saída 2: (2.out)

1, 2, 3, 4, 5, 7, 8,

Obs: a saída não precisa estar ordenada.

Bom Trabalho!