Aula Inaugural

Ciência da Computação
Universidade Federal de São Carlos

Estruturas de Dados 1

Tiemi Christine Sakata (tiemi@ufscar.br)
https://sites.google.com/site/tisakata

Algoritmos e Estruturas de Dados

Para compreender uma estrutura de dados:

1. como os dados são armazenados
2. os algoritmos para manipular os dados
3. as características de desempenho das estrutura de dados para melhor escolha

Estruturas de dados e algoritmos estão intimamente ligados. Não se pode estudar estruturas de dados sem considerar os algoritmos associados a elas, assim como a escolha dos algoritmos em geral depende da representação e da estrutura dos dados.

Para compreender uma estrutura de dados é necessário: primeiro, saber como os dados são armazenados na memória do computador. Segundo, deve estar familiarizado com os algoritmos que manipulam a informação contida em uma estrutura de dados. Terceiro, deve conhecer as características de desempenho das estruturas de dados para, se necessário, selecionar dentre elas a que mais se adeque a uma determinada aplicação.

Algoritmos e Estruturas de Dados

"Um algoritmo é uma sequência não ambígua de instruções que é executada até que determinada condição se verifique." (Wikipédia)

Algoritmos manipulam dados e esses dados devem estar organizados de forma coerente (estrutura de dados).

Os algortimos fazem parte do dia-a-dia. O conceito de algoritmo é freqüentemente ilustrado pelo exemplo de uma receita, embora muitos algoritmos sejam mais complexos. Eles podem repetir passos (fazer iterações) ou necessitar de decisões (tais como comparações ou lógica) até que a tarefa seja completada. Um algoritmo corretamente executado não irá resolver um problema se estiver implementado incorretamente ou se não for apropriado ao problema.

Sabe-se que algoritmos manipulam dados. Quando estes dados estão organizados (dispostos) de forma coerente, caracterizam uma forma, uma estrutura de dados. São a organização e os métodos que manipulam esta determinada estrutura que lhes conferem singularidade. A escolha de uma estrutura de dados apropriada pode tornar um problema complicado em um de solução bastante trivial. O estudo das estruturas de dados está em constante desenvolvimento (assim como o de algoritmos), mas, apesar disso, existem certas estruturas clássicas que se comportam como padrões.

Objetivos

- definir e diferenciar as diversas estruturas de dados;
- manipular estruturas de dados utilizando algoritmos;
- selecionar e construir estruturas de dados adequadas para aplicações específicas;
- comparar estruturas de dados por meio de adequação ao problema.

Ementa - 60h

- recursão
- tipos abstratos de dados
- listas lineares: tipos de listas lineares, alocação seqüencial, alocação dinâmica, alocação encadeada, listas duplamente encadeadas, listas generalizadas, pilha e fila
- matrizes esparsas
- árvores: nomenclatura, representação, implementação, algoritmos de busca, inserção e remoção
- árvores binárias de busca
- árvores balanceadas AVL.
- aplicação das estruturas: pilha, fila, lista e árvore.

Aulas

- 2 teóricas e 2 práticas
- Atendimento aos alunos: sala CCGT 1102
  seg-sex 13h30 às 14h30 (enviar um e-mail)
- email: tiemi@ufscar.br

Avaliações

- T – média de trabalhos práticos:
  - Trabalhos individuais ou em grupo.
  - Trabalho integrado (ED + POO).
- A – média das avaliações $((A_1 + A_2)/2)$
  A_1 (28/04), A_2 (16/06)
  NÃO haverá prova substitutiva

Avaliações

Média final $MF = 0.4 * T + 0.6 * A$, se $A \geq 6,0$
ou
Média final $MF = 0.2 * T + 0.8 * A$, caso contrário

- Atenção! Cola (plágio) em qualquer uma das avaliações implicará em nota zero
- Frequência: F
- Aprovado se: $MF \geq 6.0$ e $F \geq 75\%$

- SAC - (MF + PC)/2
  PC – prova abrangendo todo o conteúdo visto

  A nova média final (NMF) será $NMF = 6,0$ se

  $(MF + PC)/2 >= 6,0$ ou $NMF = MF$ caso contrário.

Bibliografia

- **(7) – Ziviani, Nivio. Projeto de algoritmos com implementação em pascal e C. São Paulo, Ed. Pioneira, 2004.**
- **(6) – Tenenbaum, A.M.; Langsam, Y.; Augenstein, M.J. Estruturas de Dados Usando C. Makron Books, 1995.**
- Aho, A.V.; Ullman, J.D. *Foundations of Computer Science (C edition)*. Computer Science Press (W.H.Freeman), 1995. Versão gratuita em http://infolab.stanford.edu/ ullman/focs.html
- **(1) – Aho, A.V.; Hopcroft, J.E.; Ullman, J.D. Data Structures and Algorithms. Massachusetts, Addison-Wesley, 1983.**
- **(6) – Cormen, T.; Lieserson, C.; Rivest, R. Algoritmos: teoria e prática. RJ: Elsevier : Campus, 2012.**

Bibliografia

- **(1) – Sedgewick, R., Algorithms, Second Edition, Addison-Wesley, 1988.**
- **(2) – Feofiloff, P. Algoritmos em linguagem C. Campus/Elsevier. 2009.**
- **(1) – Bentley J. Programming Pearls. 2a edição. Addison-Wesley, 2000.**
- Drozdek A. *Estrtura de Dados e Algoritmos em C++.* Thompson, 2002.
- **(1) – Knuth, D., The Art of Computer Programming, Volume 1: Fundamental Algoritms, Addison-Wesley, Second Edition, 1973.**

Exercícios

Usando as funções anteriores, solucione os seguintes problemas:

1. Leia um vetor de 20 posições imprima o vetor inverso desconsiderando os valores menores que 10.

```
srand(time(NULL));
r = rand() % 100;
```

Exercícios

```
#define MAX 100
typedef struct {
  int item[MAX];
  int tam;
} vet;

// Inicializa o vetor
void inicia (vet *v);

// Insere um elemento no final do vetor.
//Retorna 1 se a inserçao ocorreu com sucesso ou 0 cc.
int insere (vet *v, int n);

// Remove um elemento do final do vetor.
//Retorna 1 se a remocao ocorreu com sucesso e 0 cc.
int retira (vet *v, int *n);
```

```
// Verifica o numero de vezes que o elemento n se repete no vetor
int verRepeticao (vet v, int n);

// Retorna 1 caso o vetor esteja cheio ou 0 caso contrario
int cheio (vet v);

// Retorna 1 caso o vetor esteja vazio ou 0 caso contrario
int vazio (vet v);

// Retorna o elemento na posicao ind
int acessa (vet v, int ind);

// Retorna o numero de elementos do vetor
int tamanho(vet v);
```

Exercícios

2. Tentando descobrir se um dado era viciado, um dono de cassino honesto (ha! ha! ha! ha!) o lançou n vezes. Dados os n resultados dos lançamentos, determinar o número de ocorrências de cada face.
3. Simule o sorteio de um bingo.

4. Neste exercício, você pode implementar as funções que achar mais adequado para simular o controle de trens fora da estação. O programa deve manter em memória apenas os dados (id do trem e hora de saída) dos trens que estão fora da estação, considerando que o primeiro trem que sai é o primeiro a retornar.