Estruturas de Dados Avançadas

Noemi Rodriguez
sala 504RDC
noemi@inf.puc-rio.br

atendimento: 6as de 9 às 11

Informações do curso

- www.inf.puc-rio.br/~noemi/eda-19.2
  - só para 3wb!

Sobre o curso

- Estruturas de dados
  - Estratégias básicas para organização de dados na construção de algoritmos
- Objetivo:
  - Apresentar estruturas de dados de uso comum

Programa

- Parte 1
  - mapas de bits
  - árvores binárias de busca
  - árvores AVL
  - custo computacional

- Parte 2
  - árvores rubro-negras
  - árvores B
  - heaps
  - matrizes esparsas

- Parte 3:
  - tabelas de dispersão (hash)
  - estruturas de união e busca
  - grafos

Critério de Avaliação

Critério 4

NF = (G1 + G2 + G3) / 3

se G1, G2 e G3 >= 3,0 e NF >= 5,0
    então MÉDIA = NF
senão
    o aluno faz G4;

    se G4 >= 3,0
    então MÉDIA = (Gm + Gn + G4) / 3
    onde Gm e Gn são as maiores notas de G1, G2 e G3

    senão (i.e., G4 < 3,0)
    então MÉDIA = ((G1 + G2 + G3 + (G4*3)) / 6

Os graus G1, G2 e G3 serão compostos por nota de prova e laboratórios.

$Gn = (Pn^2.Ln)^{1/3}$

Bibliografia

Celes, W.; Cerqueira, R.; Rangel, J.L. (2016) Introdução a Estruturas de Dados – com técnicas de programação em C. Editora Campus.

Kruse, R.;Tondo, C.;Leung, B. Mogalla, S. (2007) Data Structures and Program Design in C. Pearson.

estruturas de dados e tipos abstratos de dados

programa
que usa esse
modulo

#include “modulo.h"

interface:

modulo.h

modulo.c
implementação

aqui aparecem as
estruturas de
dados variadas!

programa
que usa conjunto

```
#include “conjunto.h”

...
int main (void) {
  tconjunto C1, C2;
  ...
  C1 = criaConj();
  ...
```

interface: conjunto.h

```
typedef struct
      sconjunto *tconjunto;

tconjunto criaConj (void);
int insereConj (tconjunto c,
       int e);
...
int estaNoConj (tconjunto c,
        int e);
...
```

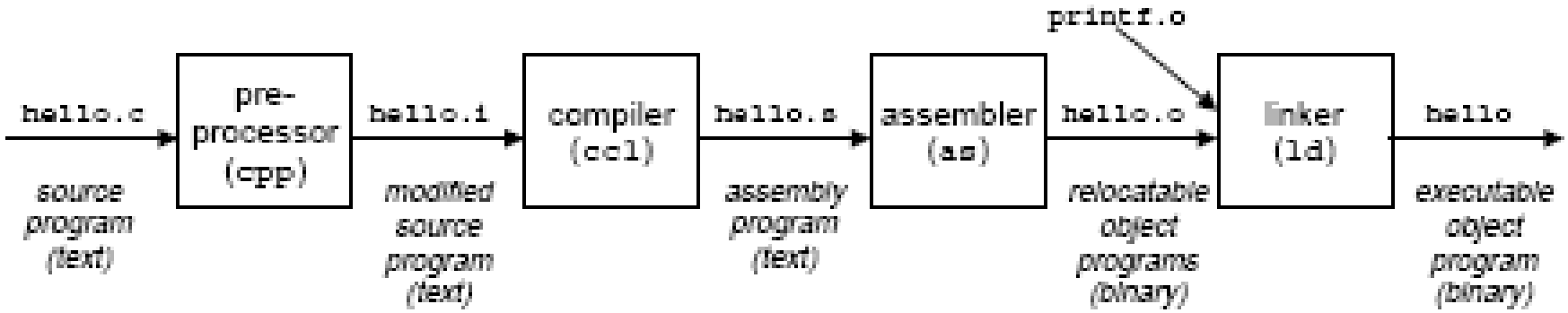
conjunto.c
implementação

lista
encadeada?
array?

Módulo Conjunto - interface

```
typedef struct
      sconjunto *tconjunto;

tconjunto criaConj (void);

int eConjValido (tconjunto c);

tconjunto insereConj (tconjunto c, int e);

tconjunto retiraConj (tconjunto c, int e);

int estaNoConj (tconjunto c, int e);

/* ... outras */
```

revisão: geração de executável



Módulo Conjunto - 1

```
#include "conjunto.h"
#include <stdlib.h>
#include <stdio.h>

struct sconjunto {
  int info;
  struct sconjunto *prox;
};

tconjunto criaConj (void) {
  return NULL;
}

...
```

Módulo Conjunto - 1

```
#include "conjunto.h"  /* fundamental */
#include <stdlib.h>
#include <stdio.h>

struct sconjunto {
  int info;
  struct sconjunto *prox;
};

tconjunto criaConj (void) {
  return NULL;
}

...
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {
  struct sconjunto *novo =
    (struct sconjunto *) malloc (sizeof(struct sconjunto));
  novo->info = e;
  novo->prox = c;
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {

  struct sconjunto *novo =
    (struct sconjunto *) malloc (sizeof(struct sconjunto));
  novo->info = e;
/* e se não conseguiu alocar??? seg fault! */
  novo->prox = c;
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {

  struct sconjunto *novo =
    (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo;
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {
struct sconjunto *novo =
  (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo; /* se não conseguiu inserir vai retornar NULL */
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Módulo Conjunto – 2

```
...
tconjunto insereConj (tconjunto c, int e) {
struct sconjunto *novo =
  (struct sconjunto *) malloc (sizeof(struct sconjunto));
  if (novo!=NULL) {
    novo->info = e;
    novo->prox = c;
  }
  return novo; /* se não conseguiu inserir vai retornar NULL */
}

int estaNoConj (tconjunto c, int e) {
   while (c!=NULL)
     if (c->info == e) return 1;
     else c = c->prox;
   return 0;
}
```

Problemas dessa implementação

- para eliminar...

Problemas dessa implementação

- para saber que está no conjunto...

podemos usar outras estruturas de dados...

- como escolher?
  - entender tradeoffs
    - custos: memória e processamento
    - operações mais frequentes
      - custo computacional

... e se não é um conjunto, mas sim um MAPA?

conjunto é caso particular, mapeando para verdadeiro/falso

Módulo Mapa- interface

```
typedef struct smapa *tmapa;

tmapa criaMapa (void);

tmapa insereMapa (tmapa c, int chave, int valor);

tmapa retiraMapa (tmapa c, int chave);

int estaNoMapa (tmapa c, int chave);

/* ... outras */
```

... e se o elemento é mais complexo que um inteiro?

Módulo Mapa de objetos - interface

```
#include "meuobjeto.h"

typedef struct smapa *tmapa;

tmapa criaMapa (void);

tmapa insereMapa (tmapa c, int chave, tobjeto *);

tmapa retiraMapa (tmapa c, int chave);

int estaNoMapa (tmapa c, int chave);

/* ... outras */
```

Módulo Mapa – Implementação

```
#include "mapaObj.h"
#include <stdlib.h>
#include <stdio.h>

struct smapa {
  int chave;
  tobjeto* obj;
  struct smapa *prox;
};

tmapa criaMapa (void) {
  return NULL;
}

…
```

laboratórios às 4as

- sala 546L
- ambiente linux
  - igual ao de inf1018
  - mas não usaremos características específicas
- entrega via ead
- conferir conta labgrad, espaço em disco e acesso ead (senha puc online)

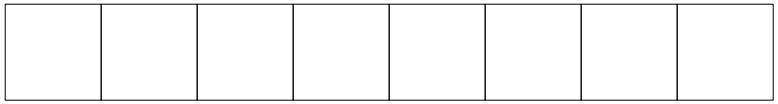
lab1: representação de conjuntos por mapas de bits

```
/* cria um conjunto com n elementos */
Set setCreate(void);
/* destroi (desaloca) o conjunto */
Set setDestroy(Set set);
/* mostra os elementos do conjunto */
void setShow(char* title, Set set);
/* insere o elemento i no conjunto */
void setInsert(Set set, int i);
/* remove o elemento i do conjunto */
void setRemove(Set set, int i);
/* cria uma copia do conjunto */
Set setCopy(Set set);
/* cria a uniao de dois conjunto  */
Set setUnion( Set set1,  Set set2);
...
```

lab1: representação de conjuntos por mapas de bits

```
typedef unsigned int *Set;

/* cria um conjunto com n elementos */
Set setCreate(void);
/* destroi (desaloca) o conjunto */
Set setDestroy(Set set);
/* mostra os elementos do conjunto */
void setShow(char* title, Set set);
/* insere o elemento i no conjunto */
void setInsert(Set set, int i);
/* remove o elemento i do conjunto */
void setRemove(Set set, int i);
/* cria uma copia do conjunto */
Set setCopy(Set set);
/* cria a uniao de dois conjunto  */
Set setUnion( Set set1,  Set set2);
...
```

lab1: Mapas de bits



- bits podem ser usados para representar presença ou ausência de elemento em conjunto

lab1: Mapas de bits

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0

- bits podem ser usados para representar presença ou ausência de elemento em conjunto

lab1: Mapas de bits

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	0

- como manipular representação?
- operadores *bit a bit* de C

operadores bit a bit

- operam sobre inteiros (sem sinal)
- o valor resultante é construído por operações sobre cada bit
- operadores &, | e ~
  - 0 falso, 1 verdadeiro

Operações Lógicas em C

- Operadores **lógicos**: or (||), and (&&) e not (!)
  - tratam qualquer argumento diferente de zero como **true** e igual a zero como **false**
  - operação resulta em 1 (true) ou 0 (false)

Expressão	**Resultado**
`!0x41`	`0x00`
`!0x00`	`0x01`
`0xaa && 0x55`	`0x01`
`0xaa || 0x55`	`0x01`

Operadores *bit a bit* em C

- Podem ser aplicados a qualquer dos tipos **inteiros**
- Baseados em álgebra booleana (**bit a bit**)

A & B

&	0	1
0	0	0
1	0	1

A | B

|	0	1
0	0	1
1	1	1

~A

~	
0	1
1	0

A ^ B

^	0	1
0	0	1
1	1	0

Exemplos

```
unsigned short a,b,c;

a = 0xFF00;          /* a = 1111 1111 0000 0000
      */
b = 0xA5A5;          /* b = 1010 0101 1010 0101
      */

c = a | b;           /* c = 0xFFA5 */

c = a & b;           /* c = 0xA500 */

c = ~a;              /* c = 0x00FF */

c = a^b;             /* c = 0x5AA5 */
```

Exemplo - máscaras

```
unsigned short a,b,c;

a = ?;                  /* a = ???? ???? ???? ????    */
b = 1;                  /* b = 0000 0000 0000 0001    */

c = a | b;

c = a & b;
```

Exemplo - máscaras

```
unsigned short a,b,c;

a = ?;              /* a = ???? ???? ???? ????    */
b = 1;              /* b = 0000 0000 0000 0001    */

c = a | b;          /* c = ???? ???? ???? ???1 */

c = a & b;          /* c = 0000 0000 0000 000? */
```

Deslocamento de bits

- **x << n**

  deslocamento (shift) para a esquerda de **n** bits

  - bits à direita são preenchidos com 0

- **x >> n**

  deslocamento (shift) para a direita de **n** bits

  - unsigned: bits à esquerda são preenchidos com 0

Exemplos

Operação	unsigned char x = 99; (0x63) **x = [01100011]**	unsigned char x = 149; (0x95) **x = [10010101]**
`x << 4`	0 0 1 1 0 0 0 0	0 1 0 1 0 0 0 0
`x >> 4` (lógico)	0 0 0 0 0 1 1 0	0 0 0 0 1 0 0 1