MC-202 — Aula 27
Generalizações: árvore geral e lista generalizada

Lehilton Pedrosa

Instituto de Computação – Unicamp

Segundo Semestre de 2015

Roteiro

1. Introdução
2. Árvore geral
3. Lista generalizada

Introdução

Um livro de estruturas de dados tem diversos capítulos (conceitos, estruturas), seções (algoritmos, técnicas, listas, árvores etc), subseções (filas, pilhas, árvore binária).

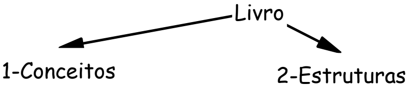
Como representar sua estrutura?

Criando uma hierarquia

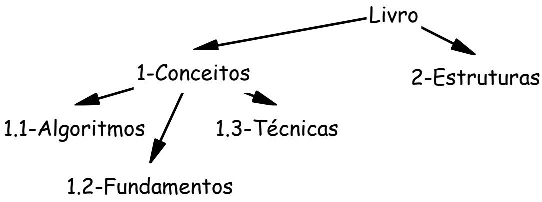
Criando uma hierarquia

Livro

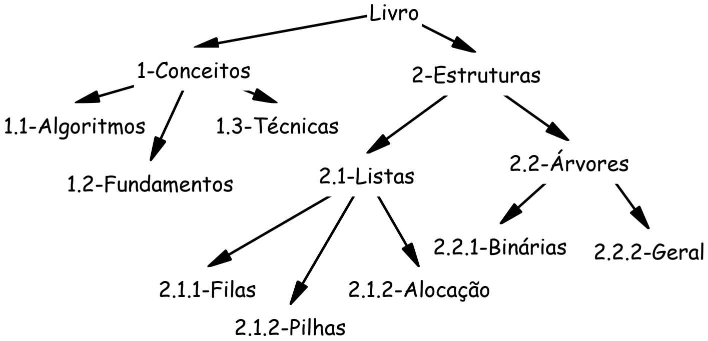
Criando uma hierarquia



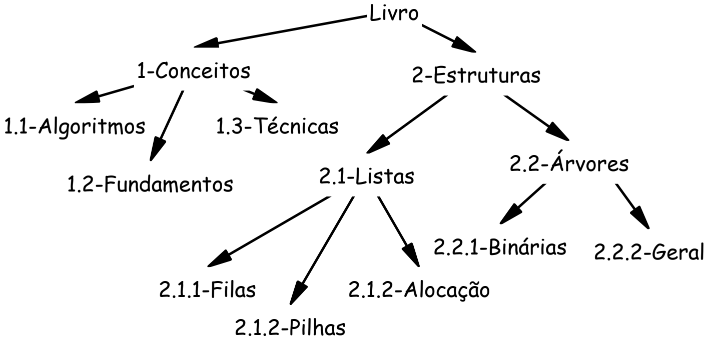
Criando uma hierarquia



Criando uma hierarquia



Criando uma hierarquia



Criamos uma árvore ordenada **não** binária!

Árvore geral

Definição

Uma árvore (ou árvore geral) é um conjunto de nós T tal que

- T contém um elemento especial r, denominado **raiz**;
- os elementos $T \setminus \{r\}$ podem ser particionados em m ($m \geq 0$) árvores disjuntas $T_1, \ldots, T_m$, denominados filhos de T.

Árvore geral

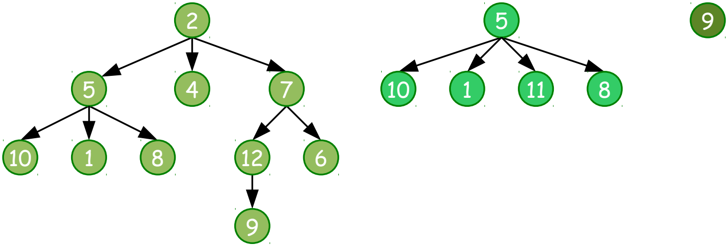
Definição

Uma árvore (ou árvore geral) é um conjunto de nós T tal que

- T contém um elemento especial r, denominado **raiz**;
- os elementos $T \setminus \{r\}$ podem ser particionados em m ($m \geq 0$) árvores disjuntas $T_1, ..., T_m$, denominados filhos de T.

Observação: quando a ordem dos filhos de T é relevante, dizemos que T é uma árvore ordenada.

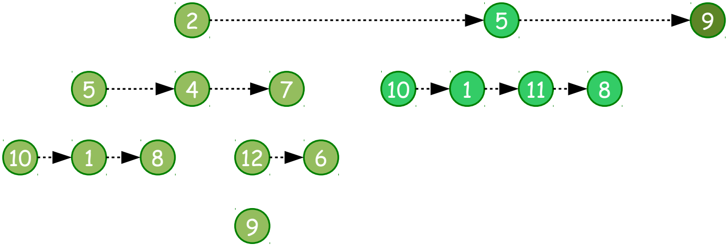
Representação de árvore geral como árvore binária



Representação

Podemos representar uma árvore geral como árvore binária:

Representação de árvore geral como árvore binária

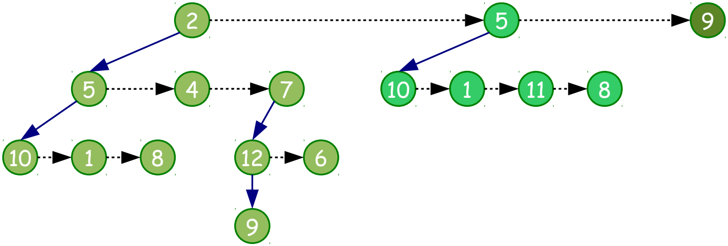


Representação

Podemos representar uma árvore geral como árvore binária:

- **filho direito**: representa o próximo irmão na árvore/floresta

Representação de árvore geral como árvore binária



Representação

Podemos representar uma árvore geral como árvore binária:

- **filho direito**: representa o próximo irmão na árvore/floresta
- **filho esquerdo**: representa o primeiro filho na árvore

Outro problema

Como representar a primeira raiz da equação quadrática?

Outro problema

Como representar a primeira raiz da equação quadrática?

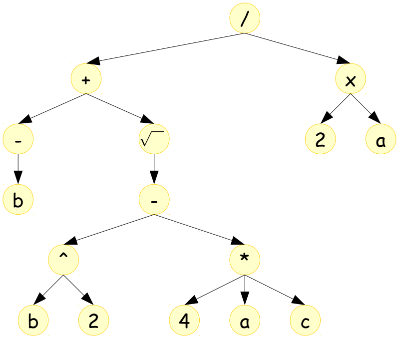
$$\Delta = b^2 - 4ac$$

$$x = \frac{-b + \sqrt{\Delta}}{2a}$$

Fórmula de Bhaskara

Raízes da equação de segundo grau

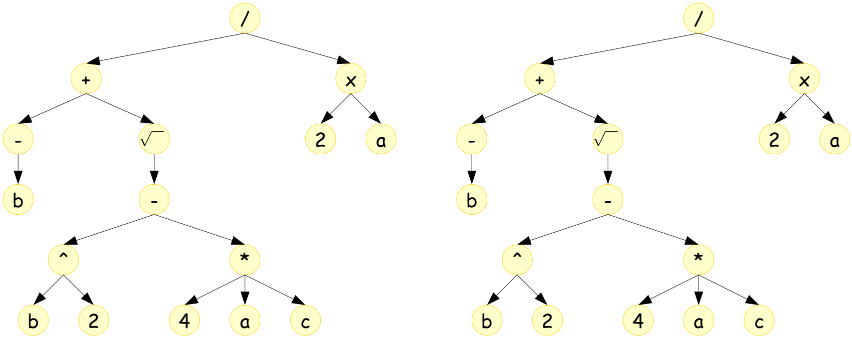
Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz

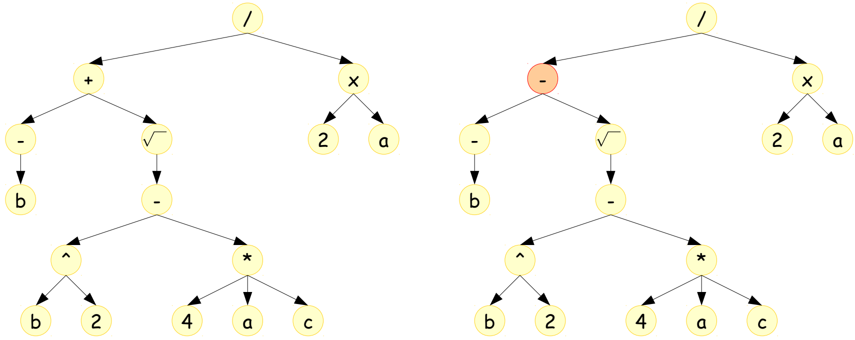
Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz e a segunda

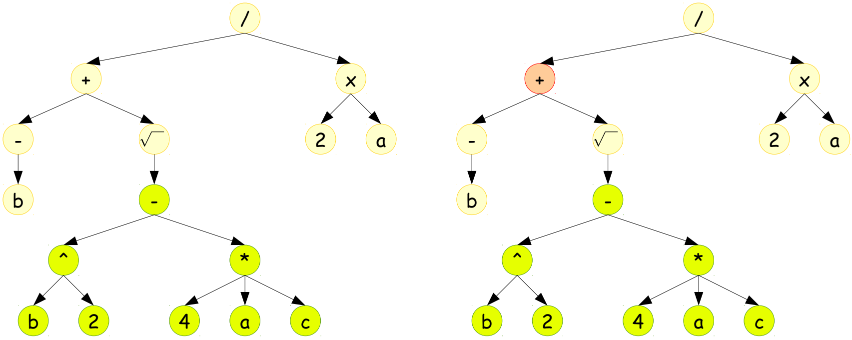
Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz e a segunda (quase idênticas)

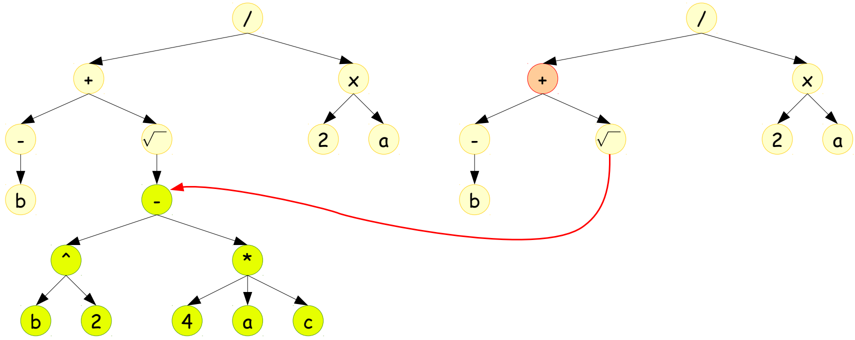
Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz e a segunda (quase idênticas)
- o valor do discriminante é o **mesmo** nas duas!

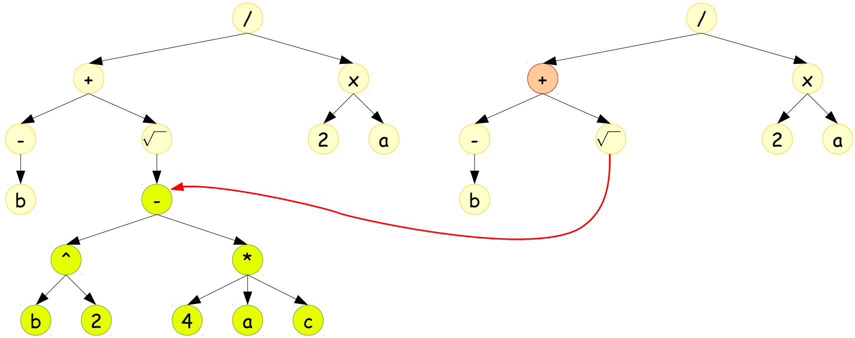
Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz e a segunda (quase idênticas)
- o valor do discriminante é o **mesmo** nas duas!

Fórmula de Bhaskara



Raízes da equação de segundo grau

- obtemos a primeira raiz e a segunda (quase idênticas)
- o valor do discriminante é o **mesmo** nas duas!

  $\Rightarrow$ Não é mais floresta!

Lista generalizada

Definição

Uma lista (ou lista generalizada) é uma sequência finita de zero ou mais **átomos** ou **listas**.

Lista generalizada

Definição

Uma lista (ou lista generalizada) é uma sequência finita de zero ou mais **átomos** ou **listas**.

Observação: um átomo é qualquer elemento que pode ser distinguido de uma lista.

Lista generalizada

Definição

Uma lista (ou lista generalizada) é uma sequência finita de zero ou mais **átomos** ou **listas**.

Observação: um átomo é qualquer elemento que pode ser distinguido de uma lista.

Exemplos

- $P = (2\ 3\ 5\ 7\ 11\ 13\ 17)$ é uma lista de alguns primos
- $C = ((2\ 2)\ (2\ 3)\ ((22)\ (2\ 2)))$ é uma lista de alguns compostos
- $N = (P\ \ C\ \ P)$ é uma lista com três listas, duas iguais
- $R = (R)$ é uma lista recursiva
- $V = ()$ é uma lista vazia

Especificando em C

Um exemplo de especificação para representar funções

```
typedef struct NoLista {
    struct NoLista *prox;
    enum {
      OPERADOR,
      NUMERO,
      LISTA
    } tipo;
    union {
      char valOp;
      double valNum;
      struct NoLista *valLis;
    };
} NoLista;
```

Especificando em C

Um exemplo de especificação para representar funções

```
typedef struct NoLista {
    struct NoLista *prox;
    enum {
      OPERADOR,
      NUMERO,
      LISTA
    } tipo;
    union {
      char valOp;
      double valNum;
      struct NoLista *valLis;
    };
} NoLista;
```

- **union** significa que apenas uma das variáveis é válida

Especificando em C

Um exemplo de especificação para representar funções

```
typedef struct NoLista {
    struct NoLista *prox;
    enum {
      OPERADOR,
      NUMERO,
      LISTA
    } tipo;
    union {
      char valOp;
      double valNum;
      struct NoLista *valLis;
    };
} NoLista;
```

- **`union`** significa que apenas uma das variáveis é válida
- o **`tipo`** indica qual valor é guardado no `union` do nó (lista ou átomo)

Representação de lista generalizada em memória

P = (* 4 a (c))

Representação

Cada nó da lista contém

Representação de lista generalizada em memória

P = (* 4 a (c))

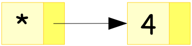
Representação

Cada nó da lista contém

- um átomo

Representação de lista generalizada em memória

P = (* 4 a (c))



Representação

Cada nó da lista contém

- um átomo

Representação de lista generalizada em memória

P = (* 4 a (c))

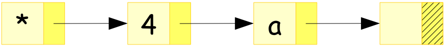
Representação

Cada nó da lista contém

- um átomo

Representação de lista generalizada em memória

P = (* 4 a (c))



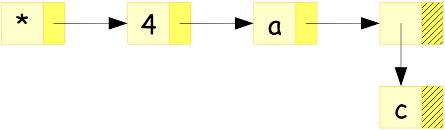
Representação

Cada nó da lista contém

- um átomo

Representação de lista generalizada em memória

P = (* 4 a (c))



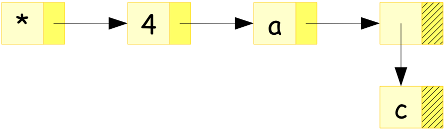
Representação

Cada nó da lista contém

- um átomo ou ou uma lista

Representação de lista generalizada em memória

P = (* 4 a (c))



D = (- (^ b 2) P)

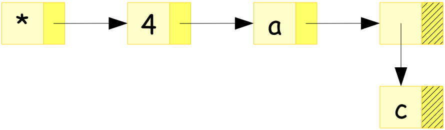
Representação

Cada nó da lista contém

- um átomo ou ou uma lista

Representação de lista generalizada em memória

P = (* 4 a (c))



D = (- (^ b 2) P)

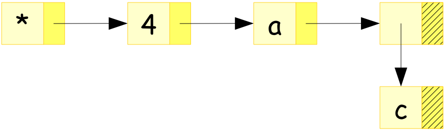
Representação

Cada nó da lista contém

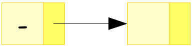
- um átomo ou ou uma lista

Representação de lista generalizada em memória

P = (* 4 a (c))



D = (- (^ b 2) P)



Representação

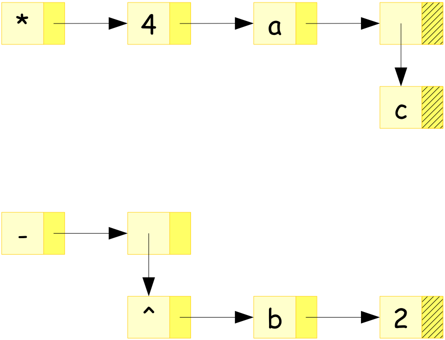
Cada nó da lista contém

- um átomo ou ou uma lista

Representação de lista generalizada em memória

P = (* 4 a (c))

D = (- (^ b 2) P)



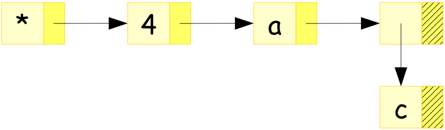
Representação

Cada nó da lista contém

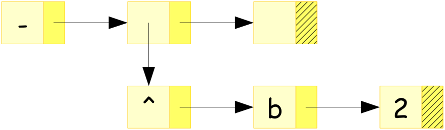
- um átomo ou ou uma lista

Representação de lista generalizada em memória

P = (* 4 a (c))



D = (- (^ b 2) P)

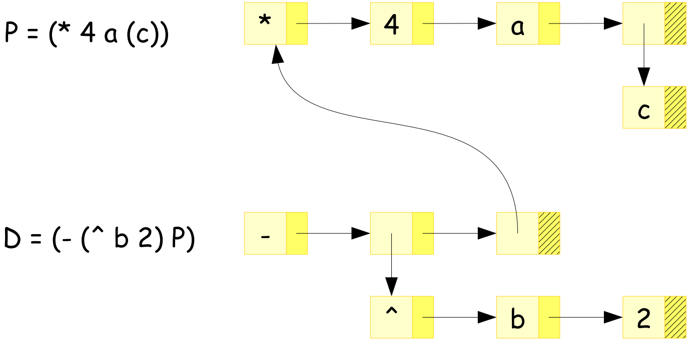


Representação

Cada nó da lista contém

- um átomo ou ou uma lista

Representação de lista generalizada em memória

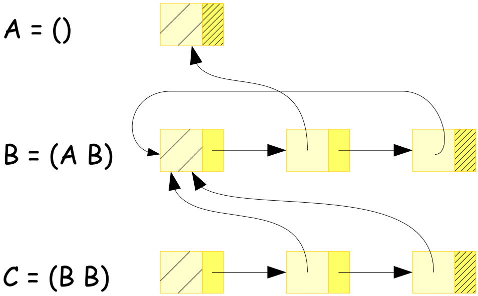


Representação

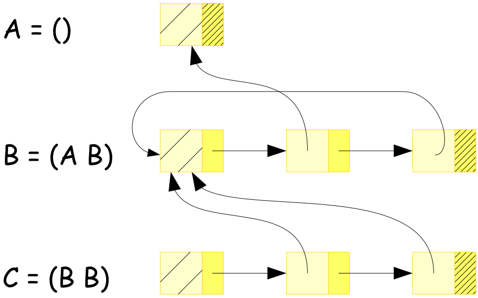
Cada nó da lista contém

- um átomo ou ou uma lista

Usando nó cabeça e referências múltiplas



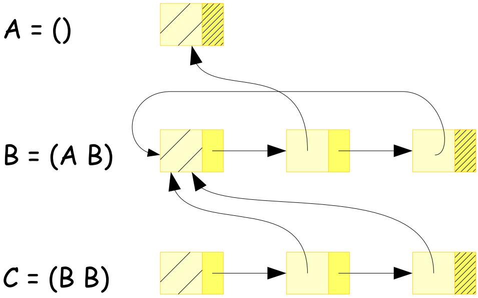
Usando nó cabeça e referências múltiplas



Perguntas:

- como seria se não quiséssemos usar nó cabeça?

Usando nó cabeça e referências múltiplas



Perguntas:

- como seria se não quiséssemos usar nó cabeça?
- o que aconteceria ao remover o primeiro elemento de B?

Representando outras estruturas com lista

Aplicações

Representando outras estruturas com lista

Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)

Representando outras estruturas com lista

Aplicações

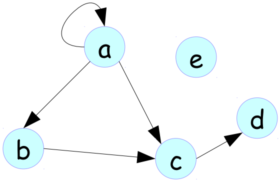
- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Representando outras estruturas com lista

Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Exemplo: podemos representar um grafo



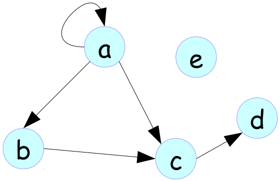
Representando outras estruturas com lista

Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Exemplo: podemos representar um grafo

G = (a b c d e)



Representando outras estruturas com lista

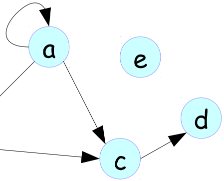
Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Exemplo: podemos representar um grafo

G = (a b c d e)

a = (a b c)



Representando outras estruturas com lista

Aplicações

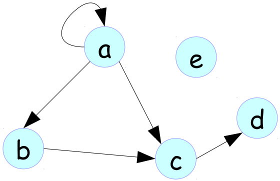
- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Exemplo: podemos representar um grafo

G = (a b c d e)

a = (a b c)

b = (c)



Representando outras estruturas com lista

Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

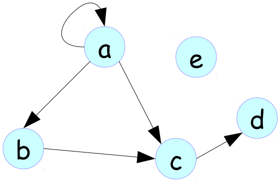
Exemplo: podemos representar um grafo

G = (a b c d e)

a = (a b c)

b = (c)

c = (d)



Representando outras estruturas com lista

Aplicações

- algumas linguagens de programação são baseadas em lista (e.g., LISP)
- nessas linguagens, a lista é a principal estrutura de dados

Exemplo: podemos representar um grafo

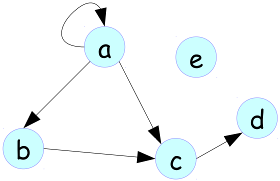
G = (a b c d e)

a = (a b c)

b = (c)

c = (d)

e = ()



Exercício - Percorrimentos

Assim como em árvores binárias, podemos percorrer árvores (gerais) em largura ou profundidade:

- **profundidade:** filhos são percorridos recursivamente;
- **largura:** os nós são percorridos em ordem de altura.

1. Defina uma política de percorrimento em profundidade de árvores gerais (com número arbitrário de filhos) para cada situação a seguir. Elas são análogas aos percorrimentos pré-ordem, pós-ordem e inordem.
   - se quisermos obter a notação polonesa a partir da árvore de operações;
   - se quisermos obter a notação polonesa reversa;
   - se quisermos obter a notação infixa.
2. Escreva um algoritmo para percorrimentos em profundidade definidos acima utilizando a representação em árvore binária. Dica: experimente os algoritmos para árvore binária.
3. Escreva um algoritmo para percorrimento em largura de árvore geral.

Exercício - Memory leak

Um *memory leak* ou vazamento de memória acontece quando alocamos vários nós dinamicamente, mas não conseguimos acessar alguns deles. Isso acontece porque não temos o endereço de memória do bloco em alguma variável local e, além disso, todas os nós alocados dinamicamente e acessíveis direta ou indiretamente a partir de uma variável local não guardam esse endereço. Quando as únicas estruturas de dados alocadas dinamicamente são listas generalizadas, é relativamente fácil decidir se há vazamento de memória. Planeje e descreva uma estratégia para verificar se há memory leak nessa situação. Que dados adicionais e algoritmos devem ser utilizados? Dica: os nós alocados dinamicamente formam um grafo! um nó é acessível a partir de outro se existe um caminho entre eles no grafo.