Banco de Memória

SCC-202 – Algoritmos e Estruturas de Dados I

Lucas Antiqueira

Alocação estática vs. dinâmica

- Alocação estática
  - Todo o espaço de memória a ser utilizado (para armazenar os elementos) é previsto na compilação do programa ou módulo (e não no decorrer da execução)
  - Esse espaço de memória permanece reservado durante toda a execução do programa, independente de estar sendo efetivamente utilizado ou não

Alocação estática vs. dinâmica

- Alocação dinâmica
  - O espaço de memória a ser utilizado (para armazenar os elementos) pode ser reservado (alocado) no decorrer da execução de um programa ou módulo, quando for efetivamente necessário (para armazenar vários elementos, é possível alocar espaço para um elemento de cada vez)
  - O espaço reservado pode ser liberado durante a execução do programa ou módulo, quando não for mais necessário (também é possível desalocar espaço de um elemento de cada vez)

Organização vs. alocação de memória

- O que aprendemos até agora?

Organização da memória / Alocação da memória	Seqüencial	Encadeada
	Estática	Dinâmica

- Seqüencial e estática **Sim**
- Encadeada e dinâmica **Sim**
- Seqüencial e dinâmica
- Encadeada e estática

Organização vs. alocação de memória

- O que aprendemos até agora?

Organização da memória / Alocação da memória	Seqüencial	Encadeada
	Estática	Dinâmica

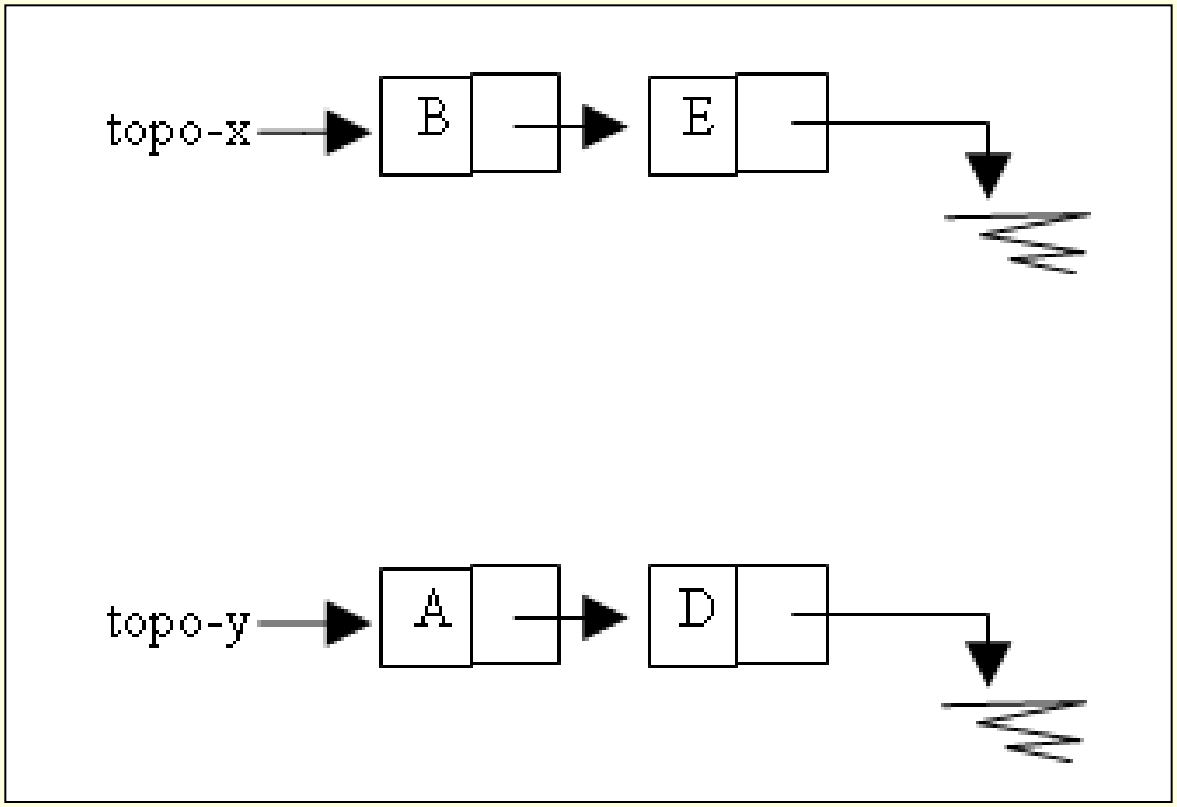
- Seqüencial e estática **Sim**
- Encadeada e dinâmica **Sim**
- Seqüencial e dinâmica
- Encadeada e estática

Encadeada e estática

- Quando usar?
  - Não há recursos de alocação dinâmica
  - Quer se restringir a quantidade de memória a ser utilizada, mas manter a organização encadeada
  - Simulação da alocação dinâmica
- Representação encadeada e estática
  - Chamada de banco de memória

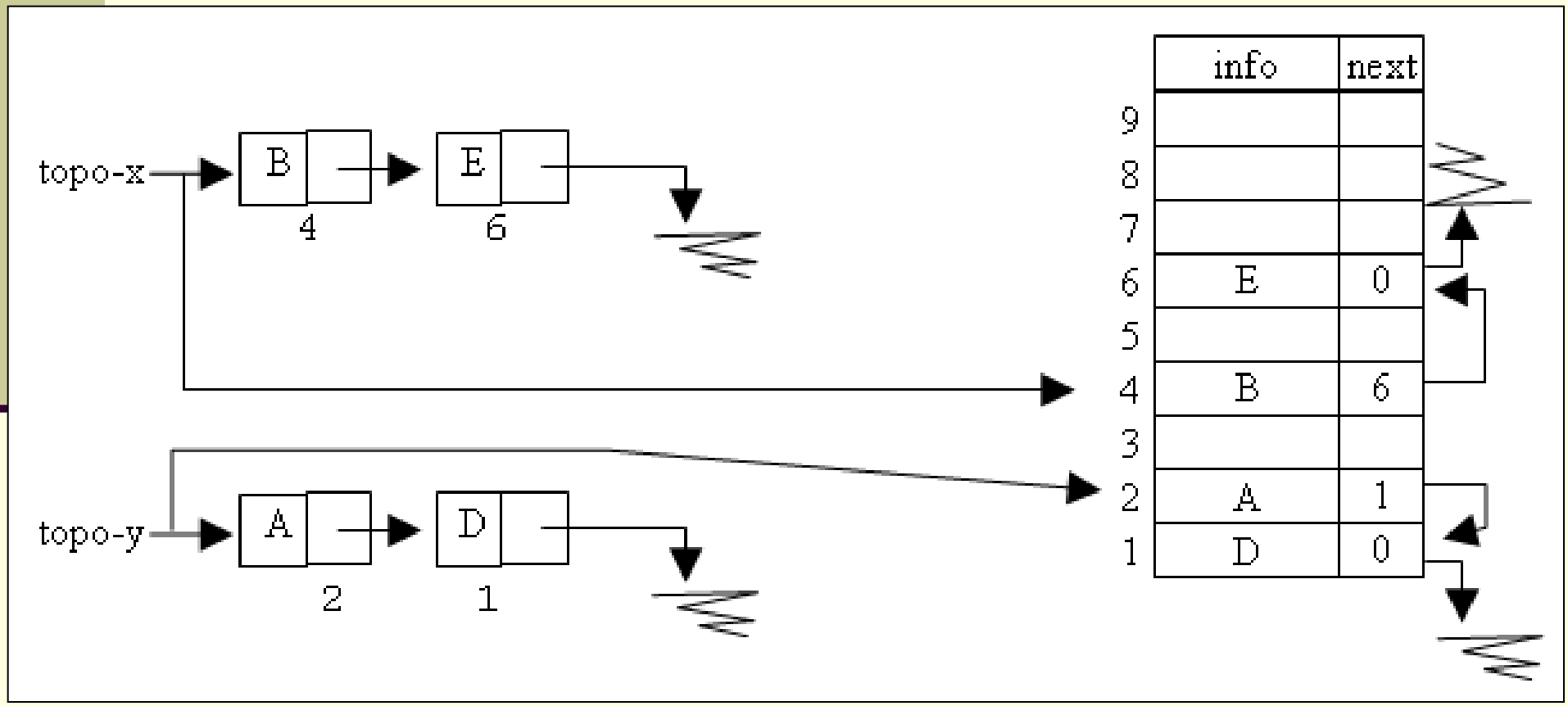
Representação

- Como simular estaticamente a alocação dinâmica?



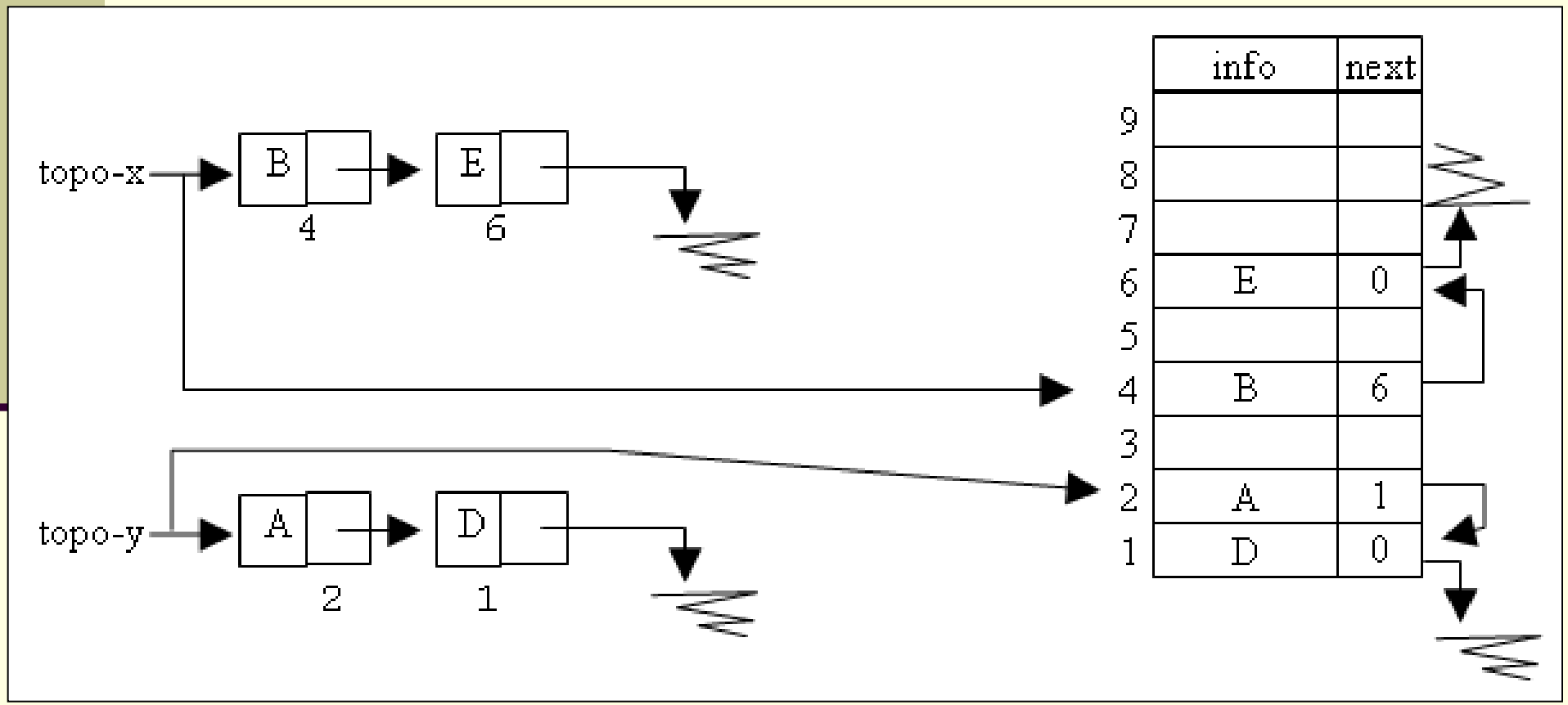
Representação

- Uso de vetor (de estruturas, possivelmente)



Representação

- Como saber quais espaços estão vazios? Como gerenciá-los?



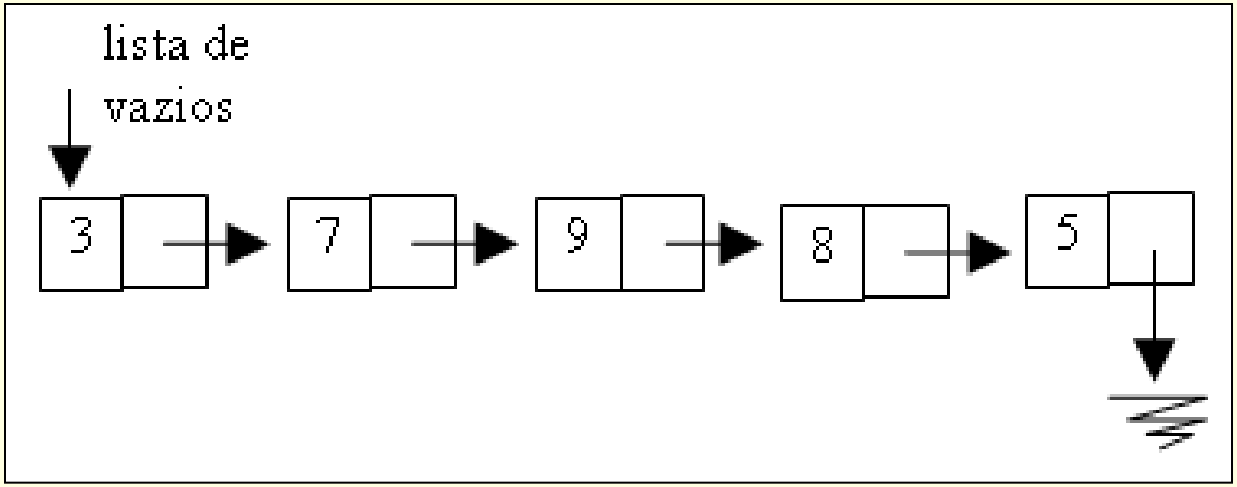
Representação

- Como saber quais espaços estão vazios? Como gerenciá-los?

Banco de memória

	info	next
9		
8		
7		
6	E	0
5		
4	B	6
3		
2	A	1
1	D	0

Possível solução: armazenamento dos vazios em uma lista encadeada, em um outro vetor, ou no próprio banco



Banco de memória

- Mantém características do encadeamento
  - Seqüência lógica dos elementos
  - Compartilhamento de memória entre estruturas similares/diferentes

- Pode representar pilha, fila ou qualquer estrutura de dados que se queira

  - Capaz de realizar as mesmas operações?
    - Cria, Entra, Sai, Empty, IsEmpty, IsFull

Banco de memória

- Suponha que você tem um banco de memória de 10 elementos que funciona como uma fila
  - Realize as operações abaixo em seqüência, mostrando, em diagramas, o estado do banco em cada passo
    - Inserir elemento A
    - Inserir elemento B
    - Retirar elemento A
    - Inserir elemento C
    - Retirar elemento B
    - Inserir elemento D

Banco de memória: implementação

Declare a estrutura de dados

Banco de memória: implementação

```
#define TAM 100

typedef int elem;

typedef struct {
   elem info;
   int prox;
} no;

typedef struct {
   int ini, fim, pvazio;
   no v[TAM];
} Banco;
```

Banco de memória: implementação

```
void create(Banco *B) {
   int i;
   for (i = 0; i < TAM - 1; i++)
       B->v[i].prox = i + 1;
   B->v[TAM-1].prox = -1;
   B->ini = -1;
   B->fim = -1;
   B->pvazio = 0;
}
```

Banco de memória: implementação

- Implemente funções nativas que simulem o `malloc` e o `free`
  - `getnode`
  - `freenode`

Banco de memória: implementação

- Implementar as operações
  - `isempty`
  - `isfull`
  - `Inserir no começo`
    - `insert_begin`
  - `Retirar do começo`
    - `remove_begin`
  - `Inserir no fim`
    - `insert_end`

Créditos

- *Material gentilmente cedido pelo Prof. Thiago A. S. Pardo*