

Alocação Encadeada

Quais as desvantagens de usar o armazenamento seqüencial para representar listas?

Uma grande desvantagem é que uma quantidade fixa de armazenamento permanece alocada para a lista, mesmo quando a estrutura estiver de fato usando uma quantidade menor ou possivelmente nenhum armazenamento.

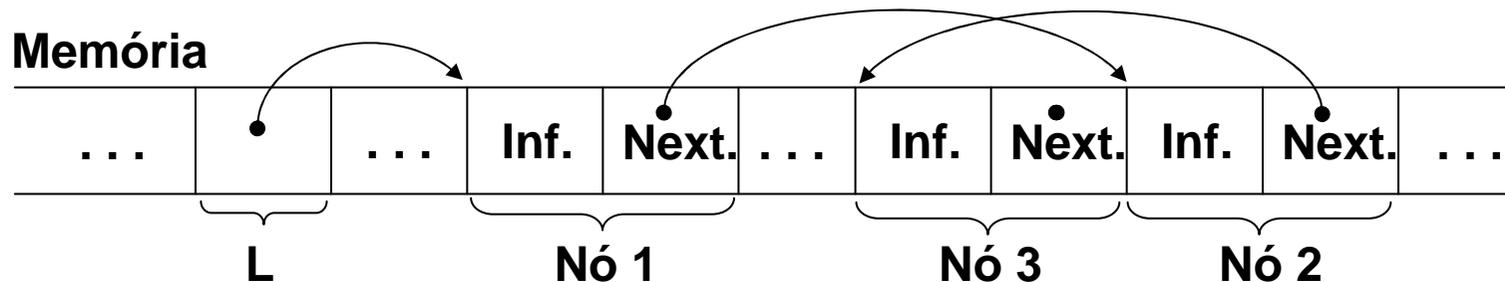
Além disso, não mais do que essa quantidade fixa de armazenamento poderá ser alocada, introduzindo, dessa forma, a possibilidade de estouro.

Para sanar estas desvantagens, foi proposta a técnica da alocação encadeada, na qual a posição relativa dos elementos da estrutura não têm que corresponder estritamente à sua posição lógica, facilitando operações como inserções e retiradas, que revelam-se muito dispendiosas, quando efetuadas sobre vetores. Tendo ainda a vantagem da alocação gradual da memória para armazenamento da estrutura, permitindo um crescimento indefinido da mesma (até os limites da memória disponíveis para o programa).

Alocação Encadeada

Na alocação encadeada os nós de uma lista encontram-se aleatoriamente dispostos na memória e são interligados por ponteiros, que indicam a posição do próximo elemento da lista. Logo, percebe-se que o custo agregado, para se obter flexibilidade, é a inclusão de um campo em cada nó (elemento), utilizado para armazenar o endereço de memória de seu sucessor, ou seja, um aumento no espaço de memória necessário para armazenar cada elemento, o esquema a seguir, ilustra este processo:

Alocação Encadeada



Obs.: Qualquer estrutura, inclusive listas, que seja armazenada em alocação encadeada requer o uso de um ponteiro que indique o endereço de seu primeiro nó.

Desta forma podemos definir o TAD `LISTA_ENC` como (considerando a informação um valor inteiro):

```
typedef struct nodo  
{  
    int inf;  
    struct nodo * next;  
}NODO;  
typedef NODO * LISTA_ENC;  
void cria_lista (LISTA_ENC *);  
int eh_vazia (LISTA_ENC);  
int tam (LISTA_ENC);  
void ins (LISTA_ENC *, int, int);  
int recup (LISTA_ENC, int);  
void ret (LISTA_ENC *, int);
```

Alocação Encadeada

Implementaremos agora as operações que compõem o TAD LISTA_ENC.

```
void cria_lista (LISTA_ENC *pl)
```

```
{
```

```
    *pl=NULL;
```

```
}
```

```
int eh_vazia (LISTA_ENC l)
```

```
{
```

```
    return (l == NULL);
```

```
}
```