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.

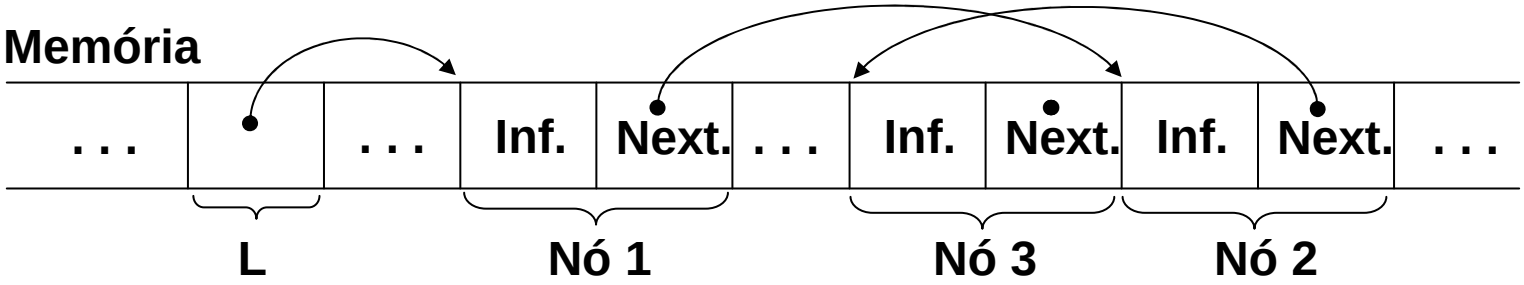
UNIVASF

Para sanar estas desvantagens, foi proposta a técnica da alocação encadeada, na qual a posição relativa dos elementos na memória não têm que corresponder estritamente à sua posição lógica na estrutura, 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, percebi-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 que a informação armazena 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.

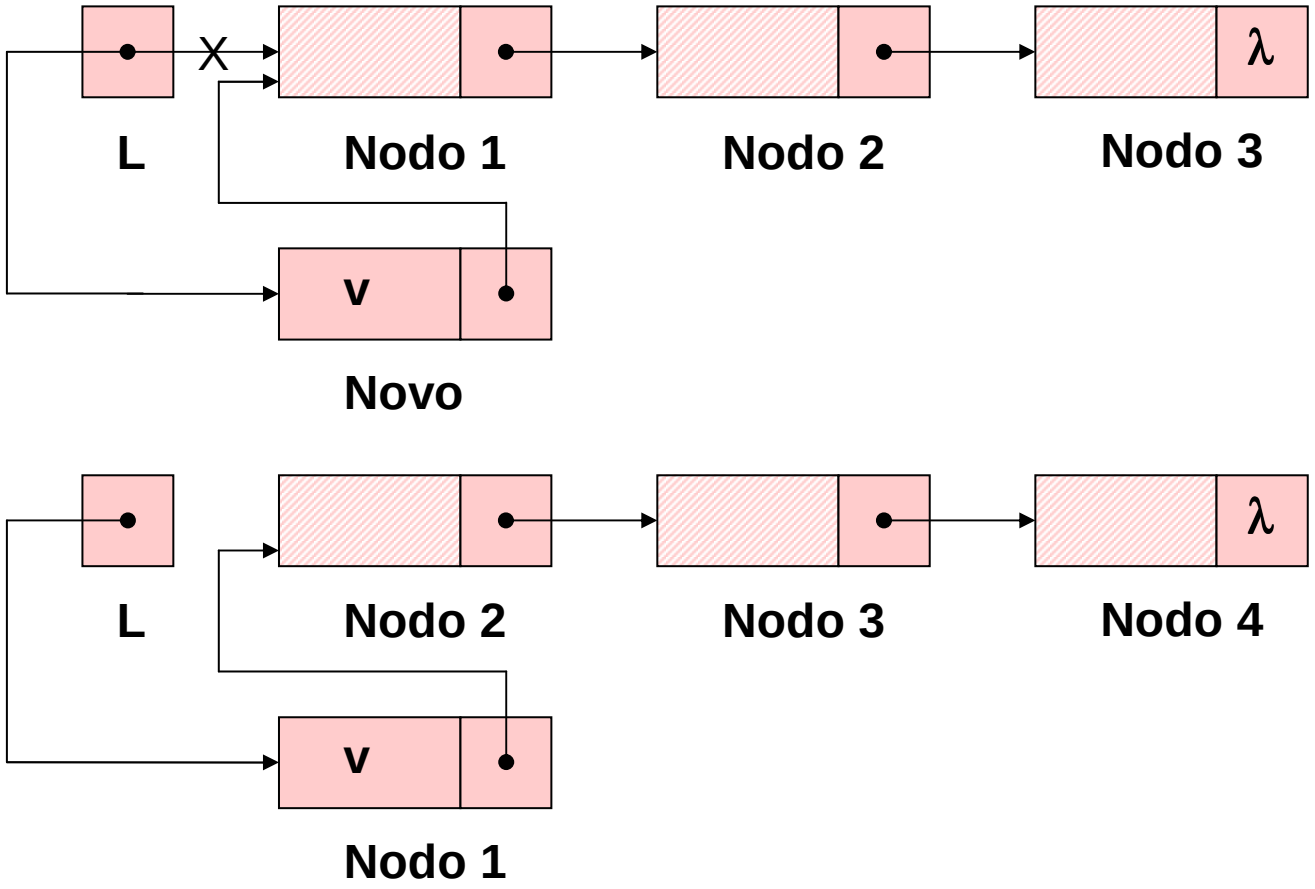
```
void cria_lista (LISTA_ENC *pl)
{
   *pl=NULL;
}

int eh_vazia (LISTA_ENC l)
{
   return (l == NULL);
}
```

```
int tam (LISTA_ENC l)
{
   int cont;
   for (cont=0; l!= NULL; cont++)
      l = l->next;
   return (cont);
}
```

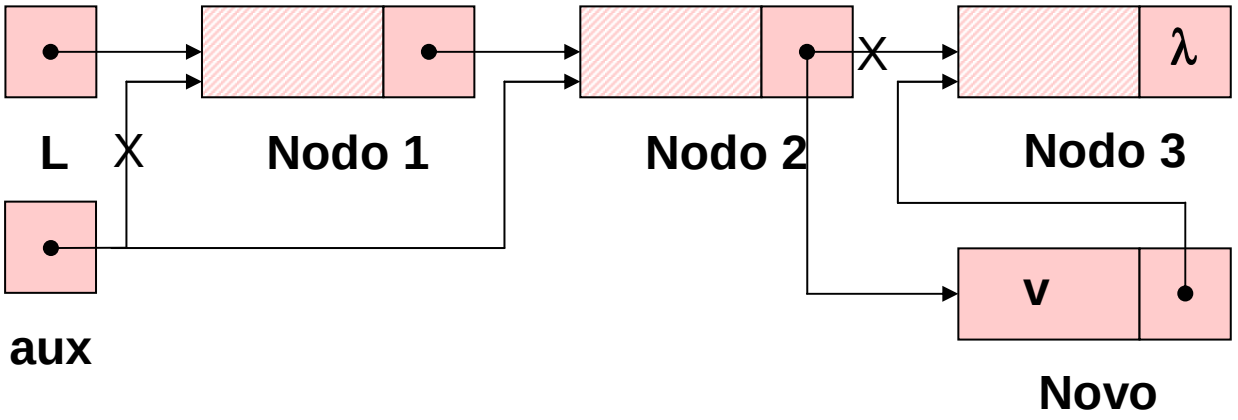
Alocação Encadeada

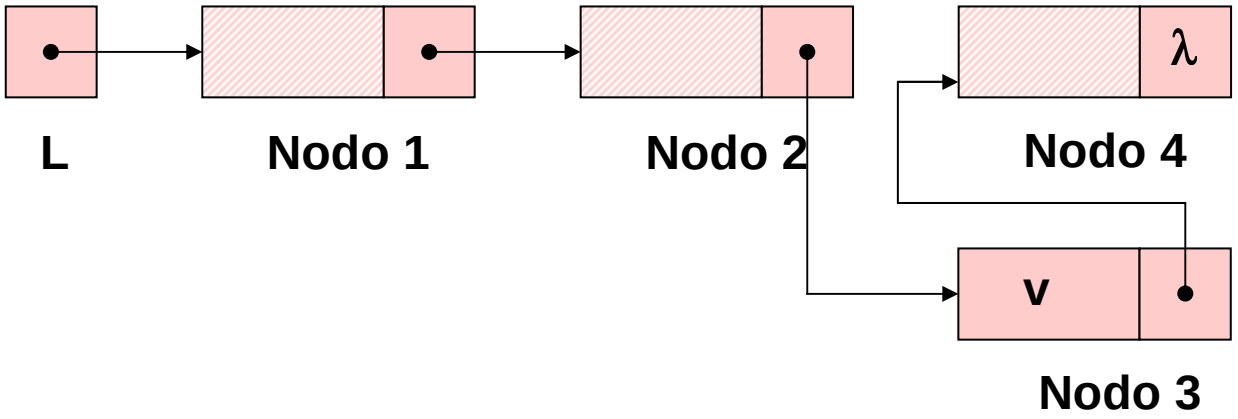
Esquema do processo da inserção de um novo nó na lista. (situação um)



Alocação Encadeada

Esquema do processo da inserção de um novo nó na lista. (situação dois)





```
void ins (LISTA_ENC *pl, int v, int k)
{
   NODO *novo;
   if (k < 1 || k > tam(*pl)+1)
   {
      printf ("\nERRO! Posição invalida para
    insercao.\n");
      exit (1);
   }
   novo = (NODO *) malloc (sizeof(NODO));
   if (!novo) { printf ("\nERRO! Memoria insuficiente!\n");
                  exit (2); }
```

```
    novo->inf = v;
    if (k==1){
       novo->next = *pl;
       *pl = novo;
    }
    else {
       LISTA_ENC aux;
       for (aux=*pl; k>2; aux=aux->next, k--);
       novo->next = aux->next;
       aux->next = novo;
    }
}
```