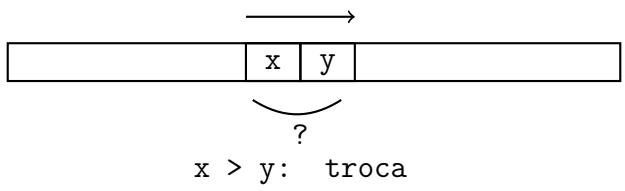
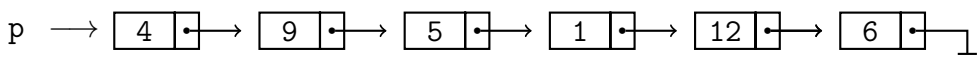
Lembre que a operação de varredura percorre um vetor comparando elementos consecutivos, e fazendo a troca quando o elemento da esquerda é maior do que o elemento da direita



Agora, nós queremos implementar essa operação na lista encadeada.

Para isso, nós imaginamos que o ponteiro p aponta para uma lista encadeada



E agora?

bom, agora nós aplicamos a nossa estratégia padrão.

Quer dizer, nós vamos remover os elementos da lista p um a um.

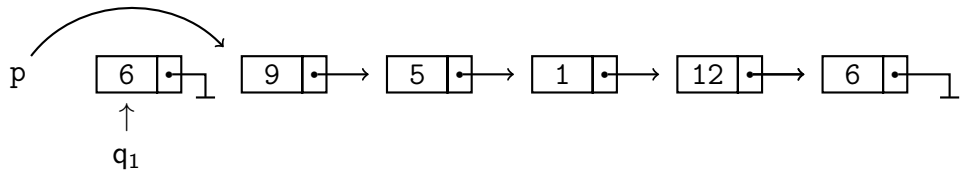
E vamos reconstruir a lista no ponteiro aux.

Só que dessa vez, nós vamos comparar o primeiro e o segundo elementos.

E daí

- se o primeiro for menor, é ele quem vai para a lista aux
- senão, é o segundo que vai para a lista aux

Para fazer isso, nós vamos manter o primeiro elemento de p separado



A coisa fica assim

```
func Varredura ( p )

      Se ( p == Nulo  OU  p->prox == Nulo )   Retorna (p)

      aux = Nulo;   ult = Nulo

      q1 = p;   q1->prox = Nulo                          // separa o primeiro elemento
      p = p->prox

      Enquanto (p != Nulo )
            Se (q1->num < p->num)                        // o 1o elemento é maior
                  Se (aux == Nulo)
                        aux = q1;   ult = q1
                  Senão
                        ult->prox = q1;   ult = q1
                  q1 = p;   q1->prox = Nulo
                  p = p->prox
            Senão                                        // o 2o elemento é maior
                  q2 = p;   q2->prox = Nulo
                  p = p->prox
                  Se (aux == Nulo)
                        aux = q2;   ult = q2
                  Senão
                        ult->prox = q2;   ult = q2

      Retorna (aux)
```

O algoritmo da bolha

Agora que nós já temos a operação de varredura, é a coisa mais fácil do mundo ordenar a lista encadeada

```
Repita  n  vezes
    p = Varredura(p)
```

◇