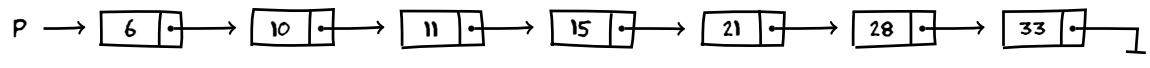
Par com soma k

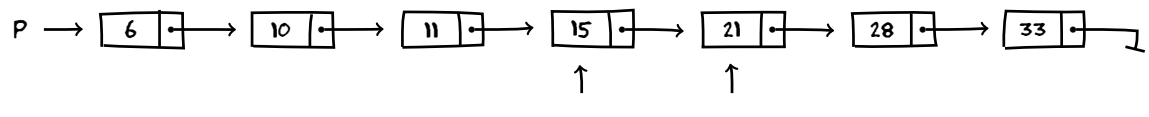
Imagine que o ponteiro p aponta para uma lista encadeada ordenada



A tarefa é

→ *Verificar se existem dois elementos cuja soma é igual a* k

No exemplo acima, para k = 36, isso nos daria



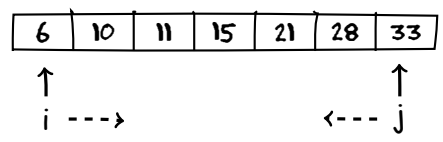
=> Sim

Bom, é fácil resolver esse problema em tempo $O(n^2)$.

Mas nós queremos fazer em tempo $O(n)$.

E queremos fazer isso de maneira recursiva.

O problema é que a estratégia que resolve o problema em tempo $O(n)$ (no vetor) trabalha nas duas pontas da lista



Só que a nossa lista não é duplamente encadeada.

E agora?

Bom, agora é a hora para uma pequena esperteza.

Quer dizer, a ideia é fazer a coisa na volta.

E daí, a gente passa um ponteiro para o primeiro elemento como o parâmetro pri para a função.

Dessa maneira, a chamada que vê o último elemento também vê o primeiro elemento — (*por meio do parâmetro*)

E dessa maneira, ela pode implementar a lógica eficiente de solução do problema — (*trabalhando nas duas pontas*)

```
Se (pi->num + pj->num == k)  Achei
Se (pi->num + pj->num < k)   avança pi
Senão                        retrocede pj
```

Na prática, pi vai ser um valor de retorno.

E pj vai ser o elemento que a chamada vê.

A coisa fica assim

```
func PSK-Rec (q)                                                  // Par com Soma K
      Se (q->prox == Nulo)                                        // a coisa começa aqui
            pi = pri;   pj = q
            Enquanto (pi->num < pj->num)                          // pi está antes de pj
                  Se (pi->num + pj->num == k)  Retorna (Nulo,1)   // Achei
                  Sse (pi->num + pj->num < k)  pi = pi->prox      // avança pi
                  Senão                        Retorna (pi,0)     // retrocede pj
            Retorna (Nulo,0)
      Senão
            pi,resp = PSK-Rec (q->prox,k,pri)
            Se (pi == Nulo)  Retorna (pi,resp)
            pj = q
            Enquanto (pi->num < pj->num)                          // pi está antes de pj
                  Se (pi->num + pj->num == k)  Retorna (Nulo,1)   // Achei
                  Sse (pi->num + pj->num < k)  pi = pi->prox      // avança pi
                  Senão                        Retorna (pi,0)     // retrocede pj
            Retorna (Nulo,0)
```