Remoção na lista com ponteiros aleatórios

Sim, mas aqui nós temos um problema.

Quer dizer, nós não podemos remover os elementos da lista com ponteiros aleatórios.

Porque pode haver um elemento (ou mais de um) em algum lugar da lista que aponta para esse elemento — (*com o seu ponteiro aleatório*)

E daí, se o elemento é removido (e o seu registro liberado da memória), nós vamos obter um erro de execução quando estivermos utilizando um ponteiro aleatório que aponta para ele.

E agora?

Bom, agora a gente adiciona dois novos campos ao nosso registro

- `apagado`: indica que esse elemento já foi removido da lista — (*mas o seu registro continua lá*)
- `cont`: indica quantos ponteiros aleatórios estão apontando para esse elemento

A ideia é que se `cont = 0`, então o registro pode ser retirado da lista e liberado da memória — (*no procedimento de remoção*)

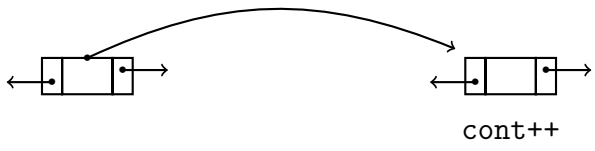
Mas se `cont > 0`, então o registro continua lá e é apenas marcado como apagado — (*utilizando o campo* `apagado`)

Os registros apagados não são levados em conta no procedimento de busca — (*mas eles podem servir de passagem para os ponteiros* q_i *e* q_j)

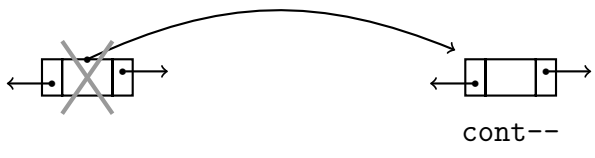
O problema com essa solução é que o número de registros apagados na lista pode ficar muito grande — (*afetando o tempo de execução da busca*)

Mas, esse problema pode ser controlado.

No momento em que o ponteiro aleatório de um novo elemento aponta para um registro, o campo `cont` desse registro é incrementado de 1



Mas no momento em que esse elemento é removido da lista (ou marcado como apagado), nós podemos decrementar o contador de 1



Dessa maneira, o campo `cont` de um elemento que já foi apagado pode ir diminuindo.

E quando ele chega em zero, o seu registro pode ser removido da lista.

Abaixo nós temos o código do procedimento de remoção

```
func remocao-PA (k,p,u)

      q = busca-PA (k,p,u)
      Se ( q == Nulo )   Retorna (Nulo)

      Se ( q->cont == 0 )
      m1 = q->ant;   m2 = q->prox
            Se ( m1 != Nulo )   m1->prox = m2
            Senão               p = m2
            Se ( m2 != Nulo )   m2->ant = m1
            Senão               u = m1
            (q->ale)->cont --
            free (q)
      Senão
            q->apagado = 1
            (q->ale)->cont --

      Retorna (p,u)
```