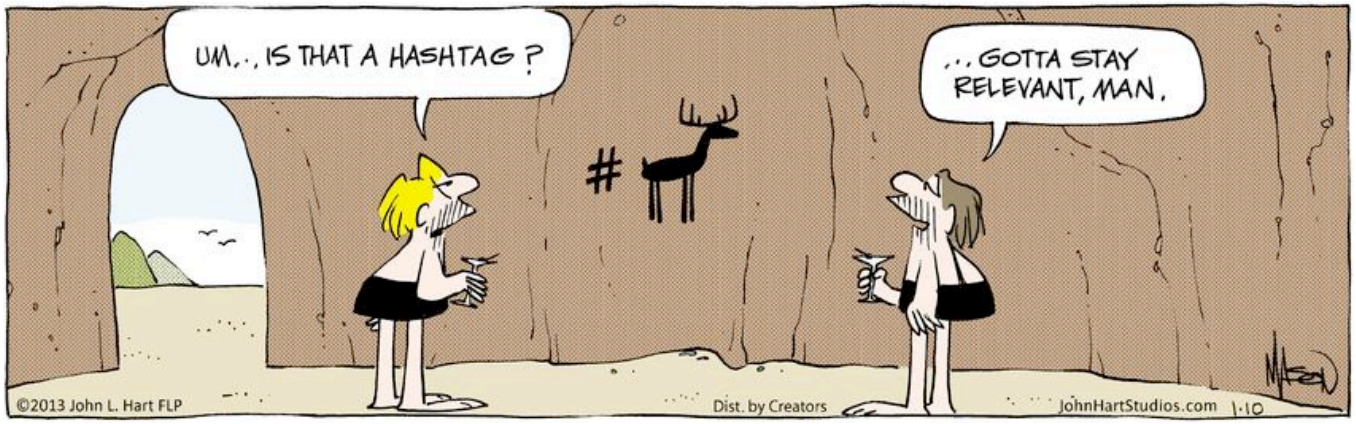
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Dicas para o projecto 2



IAED, 2015/2016

Processamento de mensagens e hashtags

- O programa irá permitir o registo e processamento de um número arbitrário *hashtags* retiradas de um número arbitrário de mensagens.

Comando	Descrição
`a <mensagem>`	Processa uma `mensagem` incrementando para cada *hashtag* encontrada o contador respectivo.
`s`	Mostra o número de *hashtags* distintas e o número total ocorrências de todas as *hashtags*.
`m`	Mostra a *hashtag* mais popular.
`l`	Lista todas as *hashtags* por ordem decrescente do número de ocorrências. Em caso de igualdade, as *hashtags* deverão ser ordenadas alfabeticamente.
`x`	Termina o programa

Exemplo *a* (adicionar, s/output)

- Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
```

```
void LeLinha(char*)
```

```
Ex:
char buffer[MAX+1];
LeLinha(buffer);
```

Exemplo *a* (adicionar, s/output)

- Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
```

```
char * fgets ( char * str, int num, FILE * stream );
```

```
Ex:
char message[MAX+1];
fgets(message, MAX+1,stdin);
```

Exemplo *a* (adicionar, s/output)

```
#define NUMSEP 11
static const char separators[] = {'
',',\t',',',';','.','?','!','"','\n',':','\0'};

void split(char *line)
{
  int i, j, k;
  char buffer[MAXLINESIZE];
  for(i = 0, k = 0; line[i] != '\0'; i++, k++) {
    buffer[k] = tolower(line[i]);
    for(j = 0; j < NUMSEP; j++) {
      if(line[i] == separators[j]) {
        if(k != 0) {
          buffer[k] = '\0';
          /* processar cada palavra guardada em buffer */
        }
        k = -1;
      }
    }
  }
}
```

Exemplo *a* (adicionar, s/output)

```
#define NUMSEP 11
static const char separators[] = {'
',' ','\t',',',';','.','?','!','"','\n',':','\0'};

void split(char *line){
    char *token = strtok(line, separators);
    while(token!=NULL) {
          /* processar aqui a palavra guardada em token */
        token = strtok(NULL, separators);
    }
}
```

Exemplo *s* (stats)

- Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
s
x
```

- Output:

```
5 7
```

Exemplo *m* (máximo)

- Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
m
x
```

- Output:

```
#iaed 3
```

Exemplo 1 (listar)

- Input:

```
a Four more years
a The North Pole is moving - and we're to blam  #science #climatechange
a If only Bradley's arm was longer. Best photo ever. #oscars
a pick the right data structure #iaed
a suggestion #iaed : try this test #project2 #iaed
l
x
```

- Output:

```
#iaed 3
#climatechange 1
#oscars 1
#project2 1
#science 1
```

Como organizar os dados?

- Não existe limite para o número de hashtags
- Usar estruturas de dados dinâmicas!
  - Listas (simplesmente ligadas?)
  - Árvores binárias (de pesquisa? equilibradas?)
  - Tabelas de dispersão?
  - Priority queues / binary heap?
  - Outra?
- Não se conhece a frequência de utilização de cada comando.

  ➡ Procurar soluções que permitam lidar com todos os comandos eficientemente.

Como organizar os dados?

- Estrutura de dados para as hashtags?

Esta estrutura de dados deverá permitir

- Introduzir uma nova hashtag na colecção
- Procurar um elemento já existente
- Obter a hashtag mais popular
- Listar todos os elementos de forma ordenada

Como organizar os dados?

- Estrutura de dados para as hashtags?

Esta estrutura de dados deverá permitir

- Introduzir uma nova hashtag na colecção **(eficientemente!)**
- Procurar um elemento já existente **(eficientemente!)**
- Obter a hashtag mais popular **(eficientemente!)**
- Listar todos os elementos de forma ordenada **(eficientemente!)**

Pensar na eficiência da solução escolhida

- A eficiência vai ser avaliada (componente avaliação automática & discussão).
- A escolha das estruturas de dados vai ter impacto na eficiência que se pode atingir.

- Complexidade da inserção de hashtags?
- Complexidade da pesquisa de hashtags?
- Complexidade na procura do elemento com maior número de ocorrências?
- Complexidade na impressão ordenada (segundo o número de ocorrências) de todos os elementos?

Abstracção de dados - para o 20! ☺

- O uso de ADTs vai ser avaliado (componente qualidade de código).

Separação entre a implementação do ADT e o cliente

- Definição do interface no `ADT.h`, implementação no `ADT.c`
- Cliente apenas usa funções definidas no `ADT.h`

➡ Verificação: posso substituir a implementação `ADT1.c` pela `ADT2.c` (ambas compatíveis com `ADT.h`), e o programa continua a funcionar?

Separação entre o tipo de objectos quardados e a definição e implementação do ADT

- Definição do interface no `Item.h`, implementação no `Item.c`

➡ Verificação: posso substituir o par `Item1.h`/`Item1.c` pelo par `Item2.h`/ `Item2.c` (ambos compatíveis com `ADT.h`), e o programa continua a funcionar?

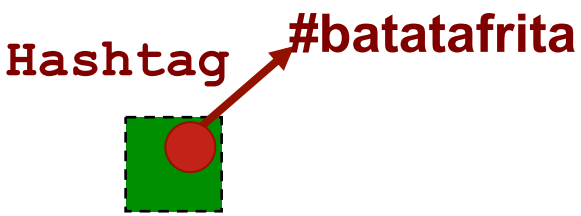
Estruturas úteis

```
typedef struct hashtag{
    char text[MAX];
    int count;
}Hashtag;
```

Hashtag

Estruturas úteis

Assim posso reservar
memória
para o nº de chars
que efectivamente preciso

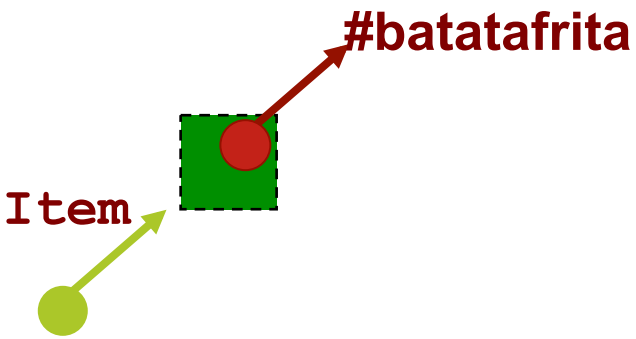



```
typedef struct hashtag{
    char* text;
    int count;
}Hashtag;
```

Estruturas úteis

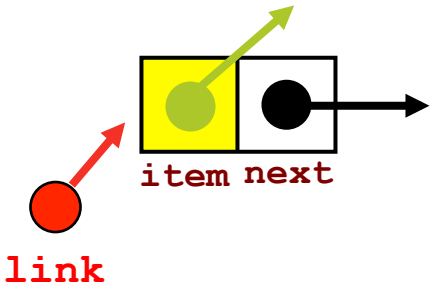
O Item passa a ser um pointer para esta estrutura permitindo uma abstracção conveniente

```
typedef struct hashtag{
    char *text;
    int count;
}*Item;
```

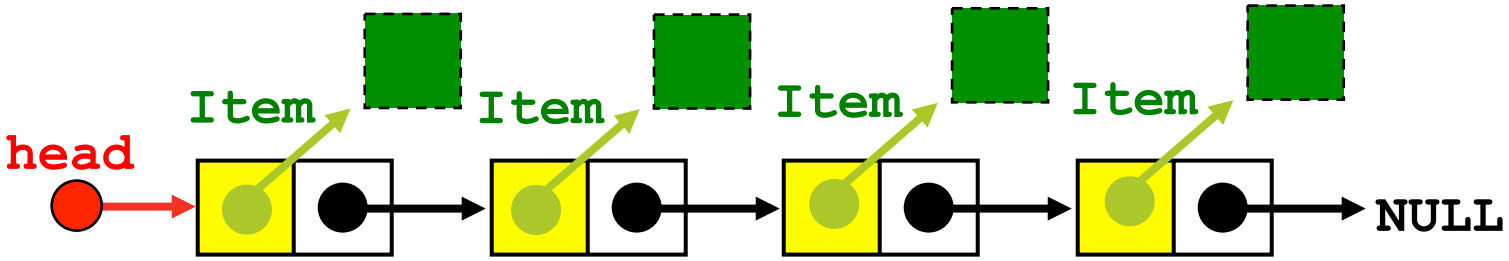


ADT: Estruturas possíveis (ver aulas!)

```
typedef struct node{
   Item item;
   struct node*next;
}*link;
```

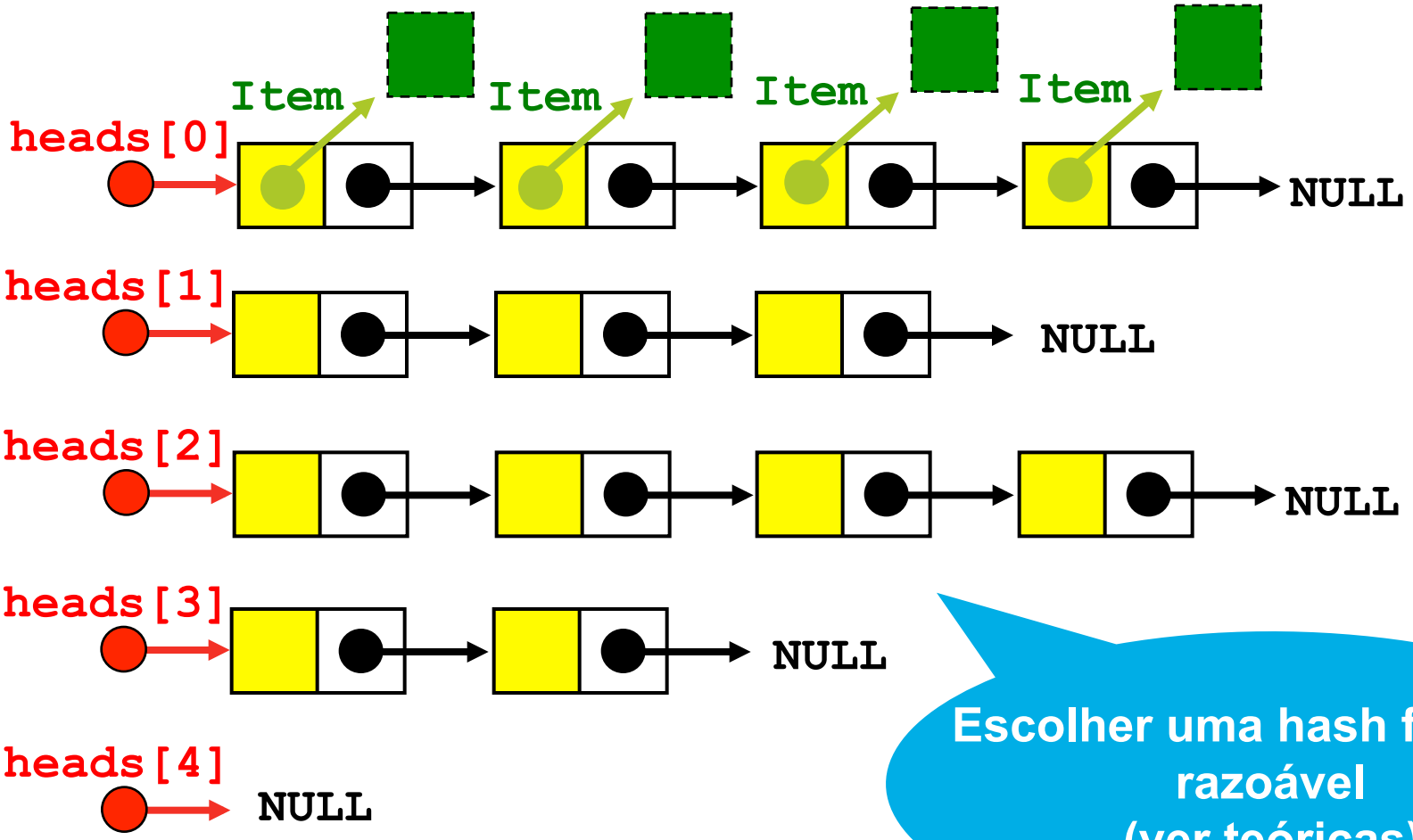


Criamos uma lista com toda a informação



ADT: Estruturas possíveis (ver aulas!)

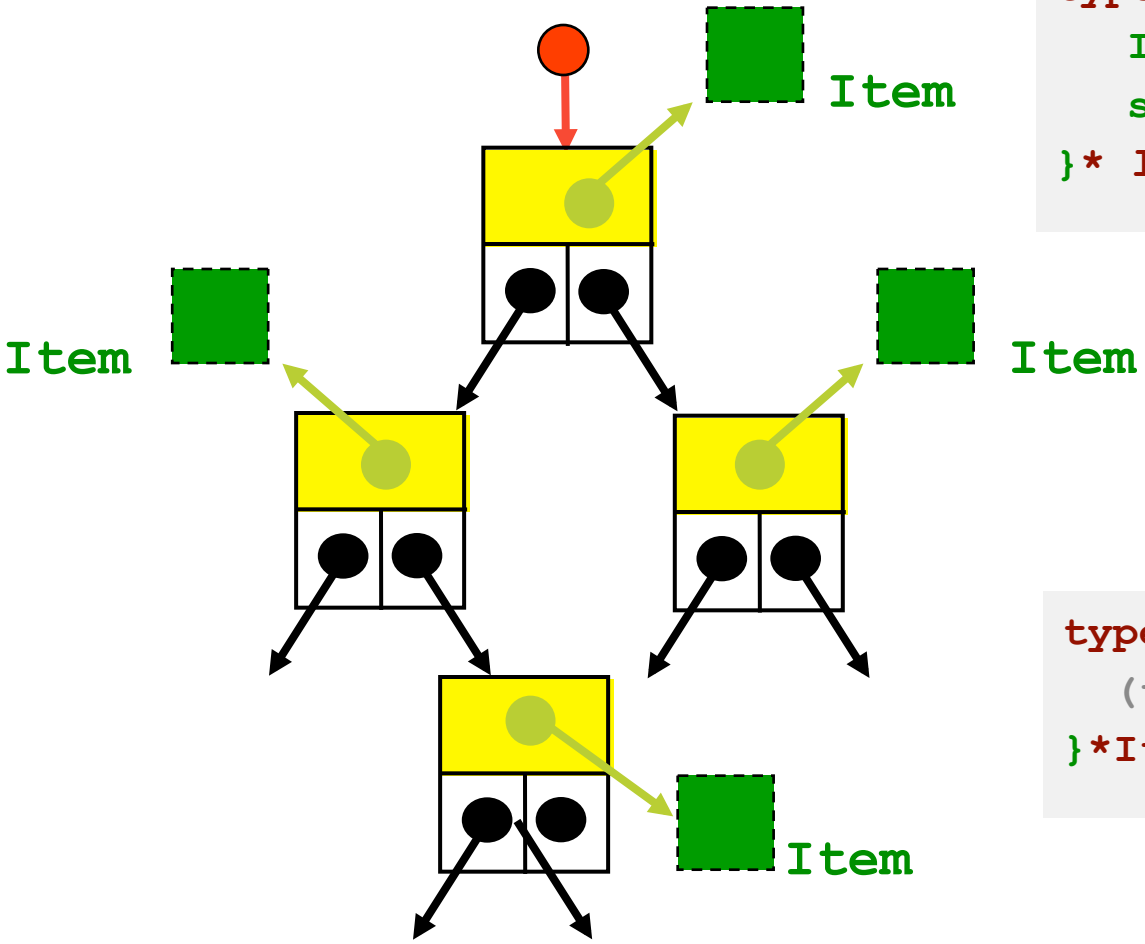
Hash table, por exemplo, com encadeamento externo...



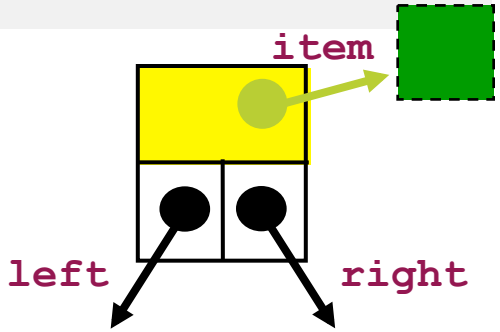
Escolher uma hash function razoável
(ver teóricas)

ADT: Estruturas possíveis (ver aulas!)

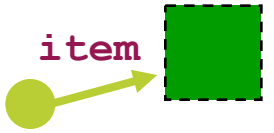
Árvores (equilibradas?) de Items... ETC!




```
typedef struct node{
   Item item;
   struct node *left, *right;
}* link;
```

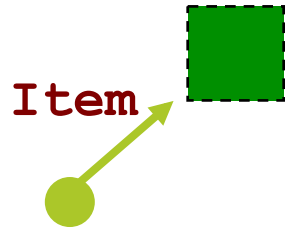



```
typedef struct hashtag{
   (texto, contador, ...)
}*Item;
```

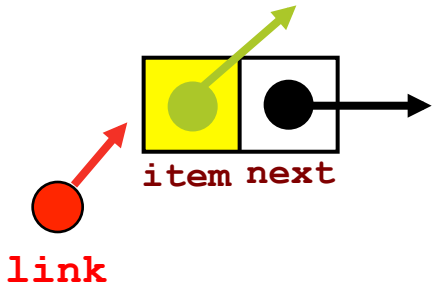


Mais abstracção

```
typedef struct hastag{
(…)
}*Item;
```



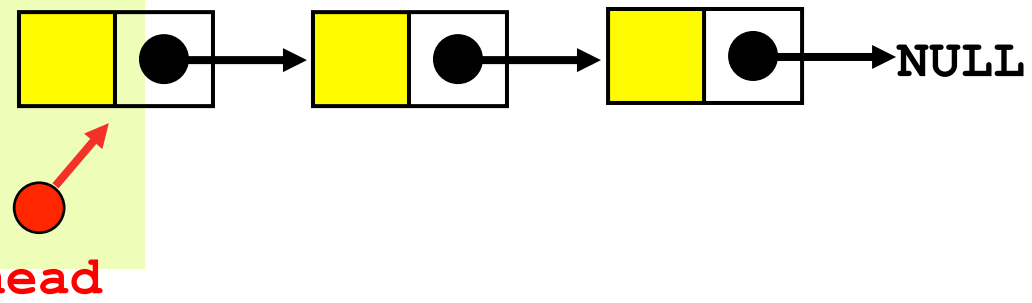

```
typedef struct node{
   void* item;
   struct node*next;
}*link;
```



Mais abstracção (exemplo para uma lista)

Esta também pode dar jeito…

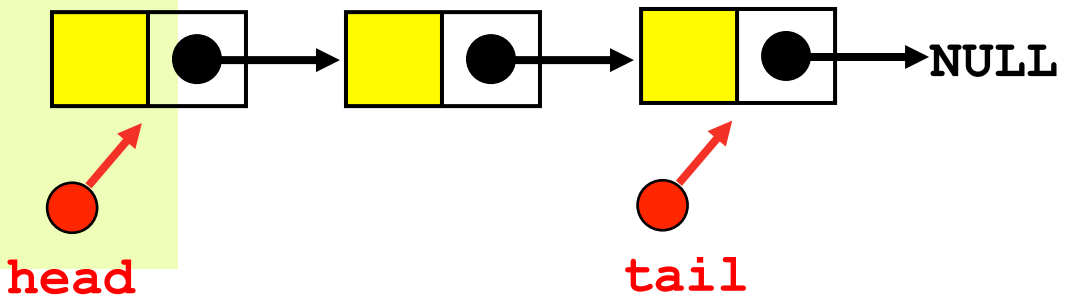
```
typedef struct list{
   link head;
   int size;
}*List;
```



Mais abstracção (exemplo para uma queue)

Esta também pode dar jeito…

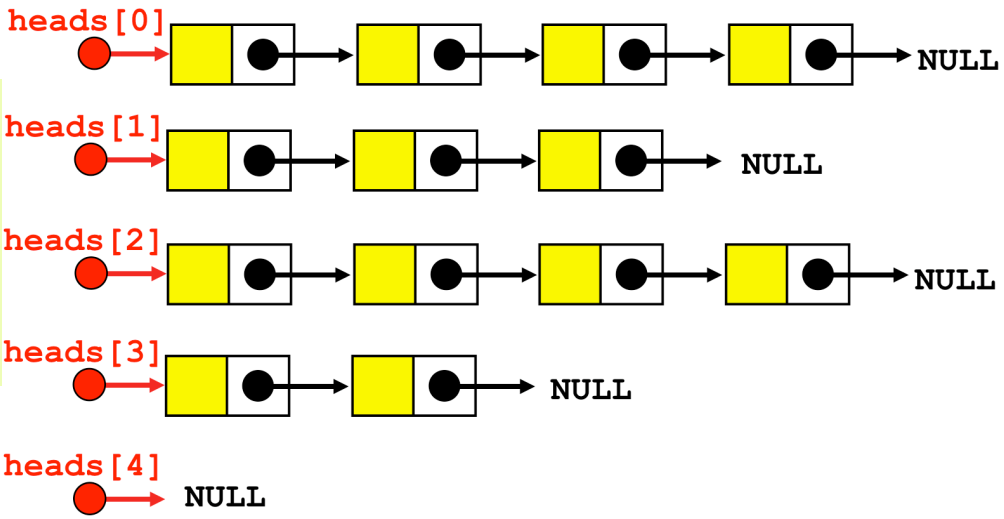
```
typedef struct queue{
   link head, tail;
   int size;
}*Queue;
```



Mais abstracção (exemplo para uma hashtable)

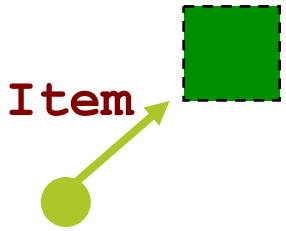
Esta também pode dar jeito…

```
typedef struct hashtable{
   link* heads;
   int M, size;
}*hashtable;
```



Funções úteis – Item's

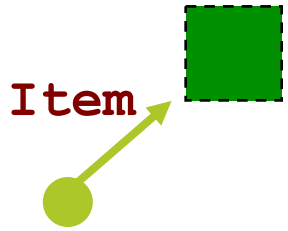
```
typedef struct hashtag{
   (…)
}*Item;
```




```
Item NewItem (char*text)
{
   /* cria um novo Item */
}
```

Funções úteis – Item's

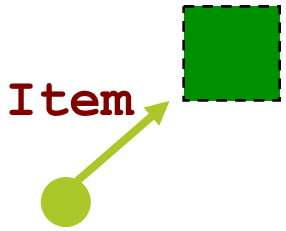
```
typedef struct hashtag{
   (…)
}*Item;
```




```
void showItem(Item x)
{
  /* mostra o conteúdo de um item – ver enunciado */
}
```

Funções úteis – Item's

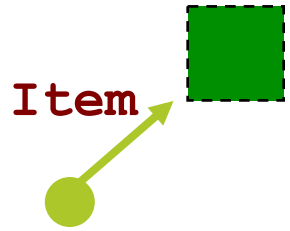
```
typedef struct hashtag{
   (…)
}*Item;
```




```
int cmpItem(Item a, Item b)
{
   /* retorna um valor < 0 se a<b, 0 se forem iguais e um valor
   >0 se b>a */
}
```

Funções úteis – Item's

```
typedef struct hashtag{
   (…)
}*Item;
```

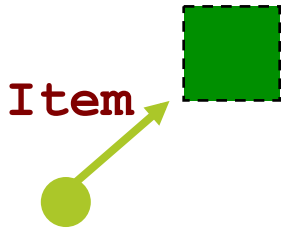



```
int cmpItem(const void *a, const void* b)
{
   /* retorna um valor < 0 se a<b, 0 se forem iguais e um valor
   >0 se b>a */
}
```

Se definir de uma forma geral (com void*) pode tentar usar a função **qsort** da **stdlib.h**....

Funções úteis – Item's

```
typedef struct hashtag{
   (…)
}*Item;
```




```
Item newItem(char*text);
void showItem(Item a);
int cmpItem(Item a, Item b);
void sort(Item a[], int l, int r);
```

Funções úteis – Estrutura de dados

```
Init?
Insert?
Search?
PrintSorted?
Free?

GetMax?
Count?
Sum?
```

Para o 20! ☺

- Procure usar abstracções
- Idente e documente convenientemente o seu código.
- Organize o código da melhor forma, se possível em vários ficheiros.
- Elimine eventuais fugas de memória (valgrind).
- Teste o seu código à medida que o escreve.
- Submeta com antecedência.

Bons códigos!