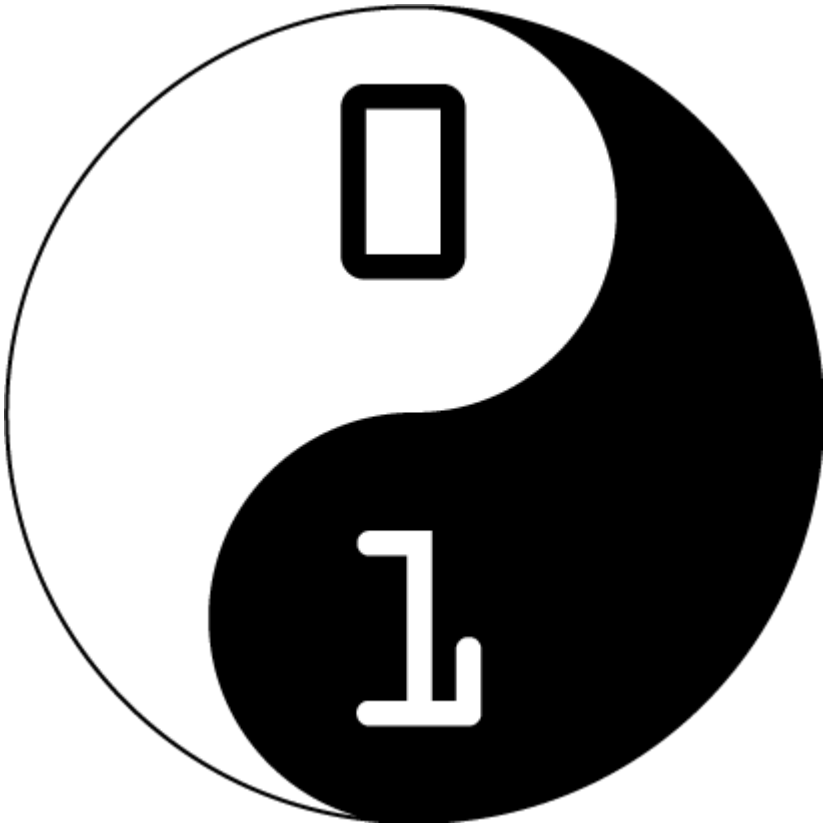
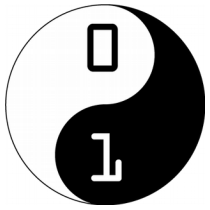
Coding Dojo

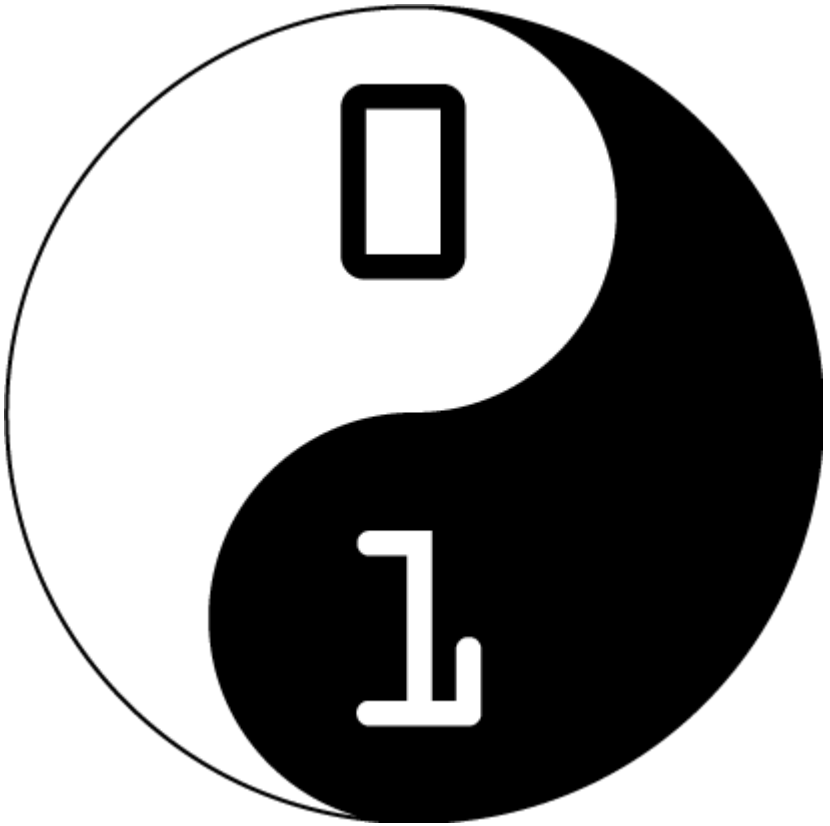




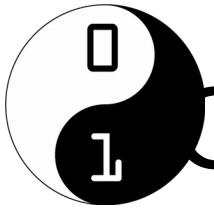
Coding Dojo - Regras

- Ponto de participação, poderá perdido se:
  - Indisciplina
  - Aluno atrasado (-0,5 após 30 min. -1 ponto após 1h)
  - Ao fazer o exercício, tentar se adiantar
  - Recusar participação como piloto/copiloto
    - Ou não querer sair do computador, quando solicitado :-)
    - Demorar para sair quando solicitado
  - Parar de participar por:
    - Uso de outro computador
    - Uso de celular
    - Usar a internet
    - Conversa

Coding Dojo



Indice Simples

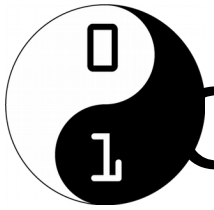


Coding Dojo – Indice Simples

d_1 = “a casa é verde”
d_2 = “a casa não é vermelha”

Chave	Valor
a	[<d_1,1>,<d_2,1>]
casa	[<d_1,1>,<d_2,1>]
é	[<d_1,1>,<d_2,1>]
verde	[<d_1,1>]
não	[<d_2,1>]
vermelha	[<d_2,1>]

Para cada termo, um documento podera aparecer apenas uma vez na lista de ocorrencias.



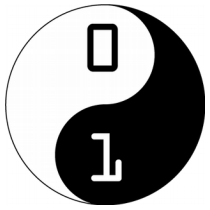
Coding Dojo – Indice Simples

O termo “a” ocorreu
Uma vez no documento d_1

d_1 = “a casa é verde”
d_2 = “a casa não é vermelh

Chave	Valor
a	[<d_1,1>,<d_2,1>]
casa	[<d_1,1>,<d_2,1>]
é	[<d_1,1>,<d_2,1>]
verde	[<d_1,1>]
não	[<d_2,1>]
vermelha	[<d_2,1>]

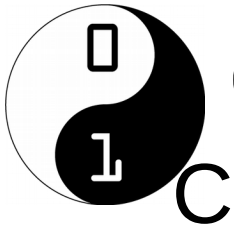
Para cada termo, um documento podera aparecer apenas uma vez na lista de ocorrencias.



Coding Dojo – Escalonador
Classe IndiceSimples – metodo index

- Crie o metodo index(String termo,int docId,int freqTermo) em que:
  - termo: termo a ser indexado
    - Um termo
  - DocId: id do documento
  - FreqTermo: numero de vezes que este termo ocorreu no documento em questao

- Este metodo deve Indexar um terminado termo que ocorreu freqTermo vezes em um determinado documento docId.
  - Para isso, ele devera adicionar uma ocorrencia (instancia da classe Ocorrencia) deste termo com uma determinada frequencia no mapa mapIndice.
    - Nao esqueca de criar uma nova lista quando o termo aparecer pela primeira vez no mapa!



Coding Dojo – Escalonador
Classe IndiceSimples – demais metodos

- getNumDocPerTerm:
  - Retorna um Map em que a chave é o termo e o valor é a quantidade de documentos em que este termo ocorreu
  - idx.getNumDocPerTerm() retorna:

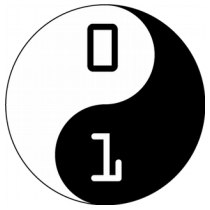
  [<“a”,2>, <“casa”,2>, <“é”,2>, <“verde”,1>, <“não”, 1>, <“vermelha”,1>]

- getNumDocumentos
  - Retorna o numero de documentos indexados
  - Idx.getNumDocumentos() retorna: 2
- getListTermos
  - Retorna a lista de termos indexados
  - idx.getListTermos() retorna:

    [“a”, “casa”, “é”, “verde”, “não”, “vermelha”]

- getListOccur
  - Retorna a lista de ocorrencias de um termo
  - idx.getListOccur(“casa”) retornara:
  - [<d_1,1>,<d_2,1>]

Chave	Valor
a	[<d_1,1>,<d_2,1>]
casa	[<d_1,1>,<d_2,1>]
é	[<d_1,1>,<d_2,1>]
verde	[<d_1,1>]
não	[<d_2,1>]
vermelha	[<d_2,1>]



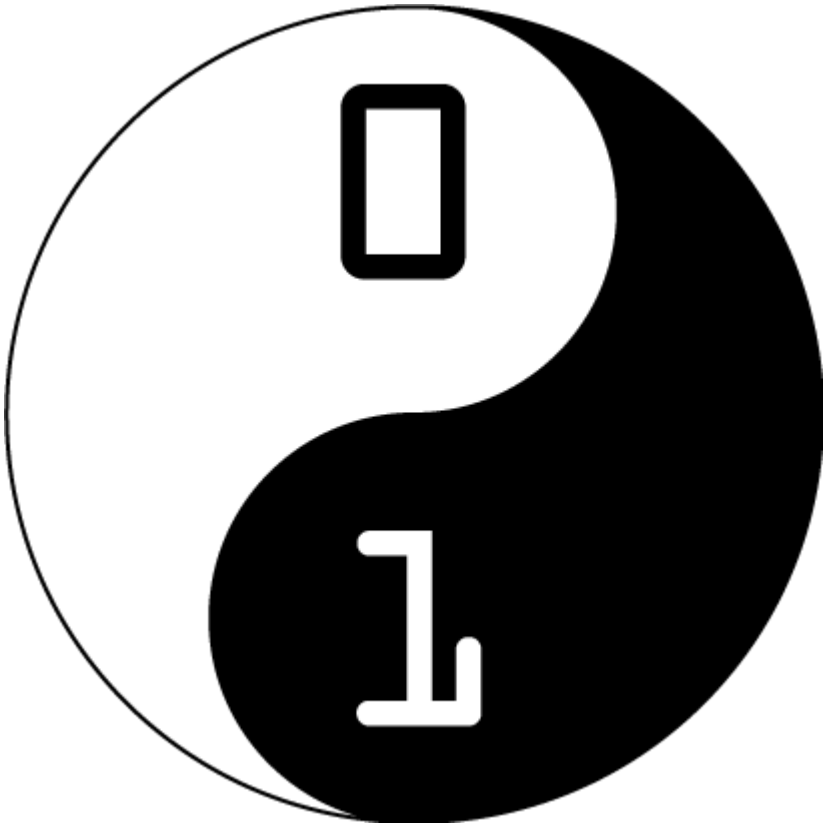
Coding Dojo – Escalonador

Classe IndiceSimples – teste

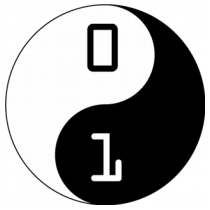
- Rode o teste TesteEstruturaIndice. Veja se o teste esta usando uma instancia do IndiceSimples. Para isso, veja se as duas primeiras linhas do metodo iniciaIndice() esta assim:

```
indiceTeste = new IndiceSimples();
//indiceTeste = new IndiceLight(15000);
```

Coding Dojo



Indice “Light”



Coding Dojo – Indice “Light”

d_1 = “a casa é verde”
d_2 = “a casa não é vermelha”

Chave	Valor
a	<1,0,2>
casa	<2,2,2>
é	<3,4,2>
verde	<4,6,1>
não	<5,7,1>
vermelha	<6,8,1>

arrTermId

1	1	2	2	3	3	4	5	6
0	1	2	3	4	5	6	7	8

arrDocId

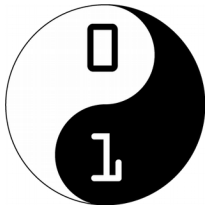
d_1	d_2	d_1	d_2	d_1	d_2	d_1	d_2	d_2
0	1	2	3	4	5	6	7	8

arrFreqTermo

1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8

- Utiliza-se vetores com tipos primitivos
  - mais leve em termos de uso de memoria
- Deve-se realizar a ordenaçao dos elementos no final
- Para cada termo, um documento podera aparecer apenas uma vez

na lista de ocorrencias.



Coding Dojo – Indice “Light”

d_1 = “a casa é verde”
d_2 = “a casa não é vermelha”

O termo id 1 (“a”) ocorreu uma vez no documento d_1

Chave	Valor
a	<1,0,2>
casa	<2,2,2>
é	<3,4,2>
verde	<4,6,1>
não	<5,7,1>
vermelha	<6,8,1>

arrTermId

1	1	2	2	3	3	4	5	6
0	1	2	3	4	5	6	7	8

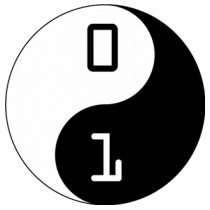
arrDocId

d_1	d_2	d_1	d_2	d_1	d_2	d_1	d_2	d_2
0	1	2	3	4	5	6	7	8

arrFreqTermo

1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8

- Utiliza-se vetores com tipos primitivos
  - mais leve em termos de uso de memoria
- Deve-se realizar a ordenaçao dos elementos
- Para cada termo, um documento podera aparecer apenas uma vez na lista de ocorrencias.



Coding Dojo – Indice “Light”

d_1 = “a casa é verde”
d_2 = “a casa não é vermelha”

Chave	Valor
a	<1,0,2>
casa	<2,2,2>
é	<3,4,2>
verde	<4,6,1>
não	<5,7,1>
vermelha	<6,8,1>

arrTermId

1	1	2	2	3	3	4	5	6
0	1	2	3	4	5	6	7	8

arrDocId

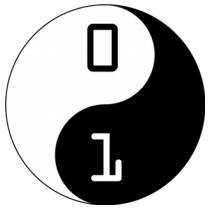
d_1	d_2	d_1	d_2	d_1	d_2	d_1	d_2	d_2
0	1	2	3	4	5	6	7	8

arrFreqTermo

1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8

- Utiliza-se vetores com tipos primitivos
  - mais leve em termos de uso de memoria
- Deve-se realizar a ordenaçao dos elementos
- Para cada termo, um documento podera aparecer apenas uma vez

na lista de ocorrencias.



Coding Dojo – Indice “Light”

d_1 = “a casa é verde”
d_2 = “a casa não é vermelha”

Chave	Valor
a	<1,0,2>
casa	<2,2,2>
é	<3,4,2>
verde	<4,6,1>
não	<5,7,1>
vermelha	<6,8,1>

arrTermId

1	1	2	2	3	3	4	5	6
0	1	2	3	4	5	6	7	8

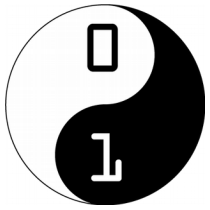
arrDocId

d_1	d_2	d_1	d_2	d_1	d_2	d_1	d_2	d_2
0	1	2	3	4	5	6	7	8

arrFreqTermo

1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8

- Utiliza-se vetores com tipos primitivos
  - mais leve em termos de uso de memoria
- Deve-se realizar a ordenaçao dos elementos
- Para cada termo, um documento podera aparecer apenas uma vez na lista de ocorrencias.



Coding Dojo – Indice “Light”

d_1 = “a casa é verde”

d_2 = “a casa não é vermelha”

Chave	Valor
a	<1,0,2>
casa	<2,2,2>
é	<3,4,2>
verde	<4,6,1>
não	<5,7,1>
vermelha	<6,8,1>

2

arrTermId

1	1	2	2	3	3	4	5	6
0	1	2	3	4	5	6	7	8

arrDocId

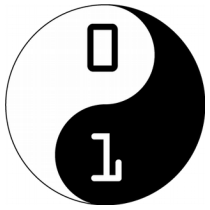
d_1	d_2	d_1	d_2	d_1	d_2	d_1	d_2	d_2
0	1	2	3	4	5	6	7	8

arrFreqTermo

1	1	1	1	1	1	1	1	1
0	1	2	3	4	5	6	7	8

- Utiliza-se vetores com tipos primitivos
  - mais leve em termos de uso de memoria
- Deve-se realizar a ordenaçao dos elementos
- Para cada termo, um documento podera aparecer apenas uma vez

na lista de ocorrencias.

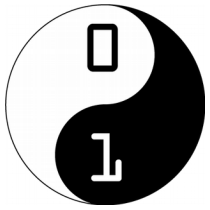


Coding Dojo – Escalonador

Classe IndiceLight – metodo index

- Crie o metodo int[] aumentaCapacidadeVetor(int[] vetor,double d) em que:
  - Retorna um novo vetor:

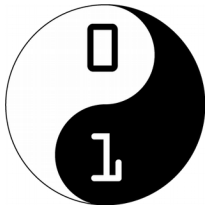
    - (100+d)% maior do que o vetor passado como parametro
    - Com todos os elementos do vetor **vetor**



Coding Dojo – Escalonador
Classe IndiceLight – metodo index

- Crie o metodo index(String termo,int docId,int freqTermo) em que:
  - termo: termo a ser indexado
    - Um termo
  - DocId: id do documento
  - FreqTermo: numero de vezes que este termo ocorreu no documento em questao

- Este metodo deve Indexar um terminado termo que ocorreu freqTermo vezes em um determinado documento docId. Para isso:
  - Utilize o posicaoIndice para resgatar o id do termo.
  - Caso este id não exista, crie-o utilizando a variável lastTermId.
    - Atualize o mapa posicaoIndice com o id do termo
  - Adicione o id do termo, id do documento e frequencia nos vetores correspondentes
    - Caso o vetor já esteja no seu limite, você deve criar um vetor 10% maior e realocar todos os elementos.
    - Apos criar os novos vetores, execute o System.gc() para requisitar que seja eliminado os vetores antigos da memoria
  - Não se preocupe em atualizar o posInicial e numOcorrencias agora (você ainda ira ordenar este vetor).



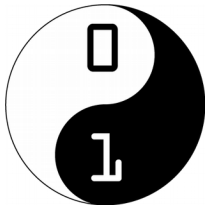
Coding Dojo – Escalonador

Classe IndiceLight – metodo concluiIndexacao

- Neste metodo deve-se:
  - ordenar o indice de acordo com o id do termo.
    - Use o metodo ordenaIndice()
  - atualize a posicaoInicial e numOcorrencia de cada termo no Map posicaoIndice.

- Para saber qual instancia **PosicaoVetor** um id de termo se refere:
  - Crie um vetor que relaciona o id do termo (como indice) e a instancia PosicaoVetor que esta no mapa posicaoIndice.
    - Percorra o mapa posicaoIndice para obter essa relação.
    - Ou seja, cosidere que o arrTermoPorId é o vetor criado. Este vetor possuirá o tamanho lastTermId+1 (pois o id do termo é incremental) você povoará o este vetor da seguinte forma:

```
para cada termo ∈ posicaoIndice:
    pos = posicaoIndice.get(termo)
    arrTermoPorId[pos.getIdTermo()] = pos;
```



Coding Dojo – Escalonador
Classe IndiceLight – demais metodos

- getNumDocPerTerm:
  - Retorna um Map em que a chave é o termo e o valor é a quantidade de documentos em que este termo ocorreu
  - idx.getNumDocPerTerm() retorna:

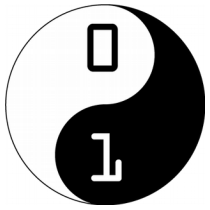
  [<“a”,2>, <“casa”,2>, <“é”,2>, <“verde”,1>, <“não”, 1>, <“vermelha”,1>]

- getNumDocumentos
  - Retorna o numero de documentos indexados
  - Idx.getNumDocumentos() retorna: 2
- getListTermos
  - Retorna a lista de termos indexados
  - idx.getListTermos() retorna:

    [“a”, “casa”, “é”, “verde”, “não”, “vermelha”]

- getListOccur
  - Retorna a lista de ocorrencias de um termo
  - idx.getListOccur(“casa”) retornara:
  - [<d_1,1>,<d_2,1>]

Chave	Valor
a	[<d_1,1>,<d_2,1>]
casa	[<d_1,1>,<d_2,1>]
é	[<d_1,1>,<d_2,1>]
verde	[<d_1,1>]
não	[<d_2,1>]
vermelha	[<d_2,1>]

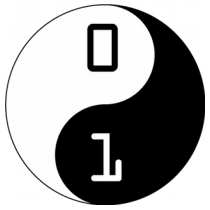


Coding Dojo – Escalonador

Classe IndiceLight – teste

- Rode o teste TesteEstruturaIndice. Veja se o teste esta usando uma instancia do IndiceLight. Para isso, veja se as duas primeiras linhas do metodo iniciaIndice() esta assim:

```
//indiceTeste = new IndiceSimples();
indiceTeste = new IndiceLight(1000);
```



Coding Dojo – Escalonador
Classe IndiceLight e IndiceSimples – teste de performance

- Você devera executar o teste TestePerformance tanto para a classe IndiceSimples e IndiceLight para verificar tanto o tempo de execuçao e uso de memoria de ambos.

- Primeiramente, você veja se o teste esta usando uma instancia do IndiceLight. Para isso, veja se as duas primeiras linhas do metodo iniciaIndice() esta assim:

```
//indiceTeste = new IndiceSimples();
indiceTeste = new IndiceLight(15000);
```

- Anote o tempo de execucao e o uso de memoria (foram impressos no console).

- Faça o mesmo para o indiceSimples:

```
indiceTeste = new IndiceSimples();
//indiceTeste = new IndiceLight(15000);
```