A opção mais simples para construção do índice é a seguinte:

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>]

Neste exemplo, para os documentos d_1 e d_2, o índice foi armazenado em um dicionário onde a chave é o termo e o valor é uma lista ocorrência representado pelas tuplas <id_doc, freq> onde id_doc é o id do documento e freq é a frequência do termo no documento.

Outra opção é a seguinte, para o mesmo exemplo de documentos:

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>

<1,d_1,1>	<1,d_2,1>	<2,d_1,1>	<2,d_2,1>	<3,d_1,1>	<3,d_2,1>	<4,d_1,1>	<5,d_2,1>	<6,d_2,1>
0	1	2	3	4	5	6	7	8

Desta vez, há um dicionário onde a chave é o termo e o valor é uma tripla <termo_id,pos_inicial,num_oc>. Onde termo_id é o id do termo, pos_inicial e num_oc são valores que possibilitam saber as posições do vetor de ocorrencias que correspondem à este termo: pos_inicial é a posição inicial e num_oc é o número de ocorrências deste termo (ou seja, o número de documentos que o termo ocorre). A lista de ocorrências é representada por <termo_id, doc_id, freq>. Por exemplo, para o termo “a” o valor da tripla é <1,0,2>, isso significa que, no vetor de ocorrências, existe duas ocorrências do termo “a” sendo que a primeira ocorrencia está na posição 0 e a segunda na posição 1 indicando que o termo “a” ocorre uma vez no documento d_1 e uma vez no

d_2.

Note que, para construir o índice na memória primaria dessa forma precisamos de:
1- Criar o array de ocorrencias <termo_id, id_doc, freq>. Para cada palavra nova, adicionar no dicionario o termo junto com a tripla <termo_id, null, null>. Pois ainda não sabemos a posição inicial e o número de documentos que possuem esse termo.
2- Ordenar o array de ocorrências por termo_id;
3- Utilizando o array de ocorrencias, atualizar o dicionário, para cada termo, sua tripla <id_termo, pos_inicial, num_oc> com a posição inicial e número de documentos que o termo ocorre.

Com o objetivo de diminuir mais ainda o consumo de memória, o array de ocorrencias pode ser dividido em 3 arrays de inteiro (tipo primitivo). O primeiro array é o id do termo, o segundo é o id do documento e o terceiro é a frequência do termo no documento. Assim, o consumo de memória diminuirá significativamente. Mas, lembre-se de usar apenas vetores e tipos primitivos. Ver em: http://java-performance.info/overview-of-memory-saving-techniques-java/
(O overview é sobre java, mas se aplica a outras linguagem de programação). A forma de criação indice é parecida com o anterior, a única diferença é a necessidade de ordenar os 3 arrays ao mesmo tempo. Veja o na figura seguinte:

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>

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

Caso prefira, é possível armazenar as ocorrencias em arquivo. Dessa forma, o dicionário terá uma tupla <termo_id,offset,num_oc> onde offset é o deslocamento no arquivo que é necessário para buscar a primeira ocorrencia da palavra. Lembrar que, como a ordenação é por termo, o armazenamento do vetor de ocorrências em memória secundária implica na necessidade de ordenação externa (o que dificulta um pouco mais o trabalho):

Chave	Valor
a	<1,0,2>
casa	<2,6,2>
é	<3,12,2>
verde	<4,18,1>
não	<5,21,1>
vermelha	<6,24,1>

offset	Arquivo
0	<1,d_1,1>
3	<1,d_2,1>
6	<2,d_1,1>
9	<2,d_2, 1>
12	<3,d_1, 1>
15	<3,d_2, 1>
18	<4,d_1, 1>
21	<5,d_2, 1>
24	<6,d_2, 1>