Aula 2

Dados ordenados

- Os dados ordenados são importantes para recuperar a informação de forma eficiente

E se os dados não estiverem ordenados?

Considere:

5 7 2 4 6 9 3 1

0 1 2 3 4 5 6 7 (posição dos dados)

Um algoritmo de busca simples:

boolean achei = false;

for(int i = 0; i < 8 ; i++)

if(v[i] == x)

achei = true;

Melhorando:

int i = 0;

boolean achei = false

while (i < 8 && !achei)

if(v[i] == x)

achei = true;

else

i++;

Pensando em custo

Pior caso: elemento procurado não está.

tamanho da lista número de interações

10

100

10000

...

n

10

100

10000

...

n

Dizemos que o tempo de execução cresce de forma linear com relação ao tamanho da lista

Caso construa um método:

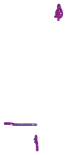
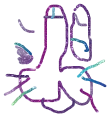
- Entradas : um vetor um valor



- Saída: a posição da primeira ocorrência de x no vetor ou -1 caso , ele não seja.



Static int buscaSimples (int[]v , int x){

 for(int i = 0 ; i < v.legth ; i++)

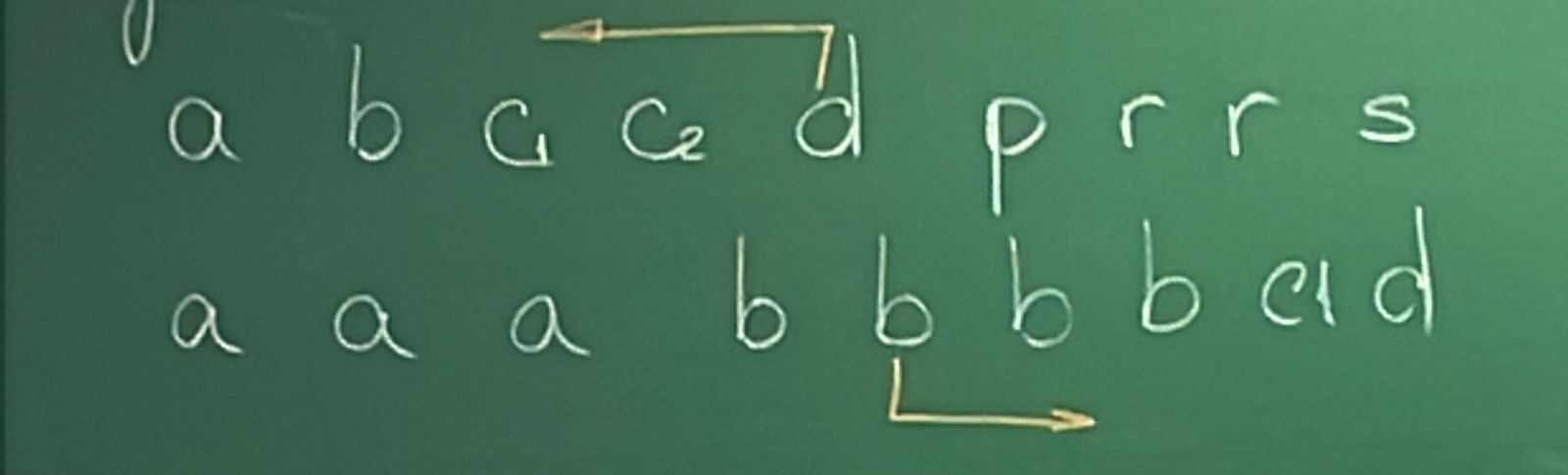
if (v[i] == x)

 return i;

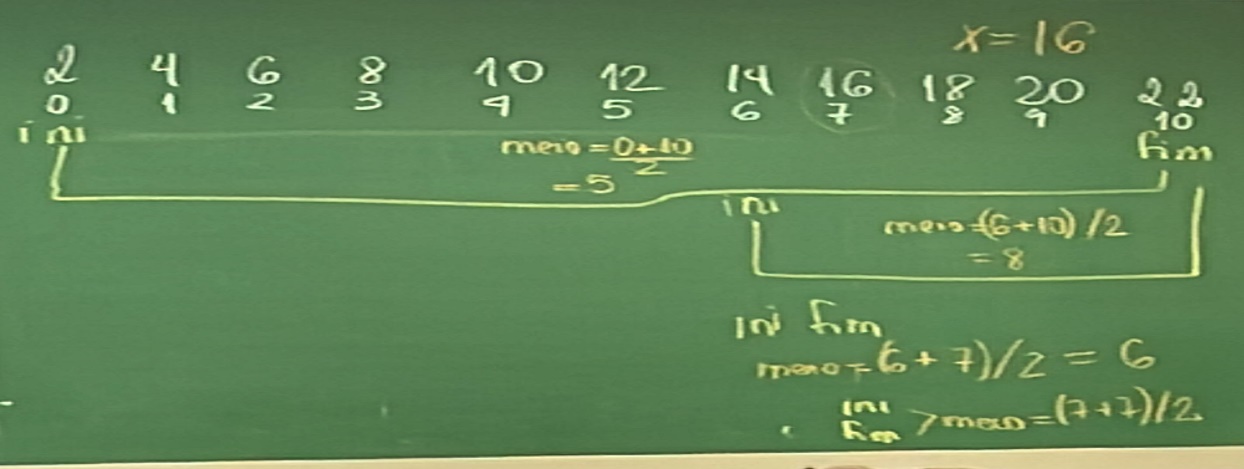
 return -1;

}

Se os dados estiverem ordenados ganhamos na busca?



Não conhecemos os dados!

Dada uma lista de valores ordenados , o melhor chute para reduzir o “erro” é a metade , além disso , em cada interação , reduzimos o espaço de busca à metade do espaço atual: 

Como foi a redução do tamanho da lista?

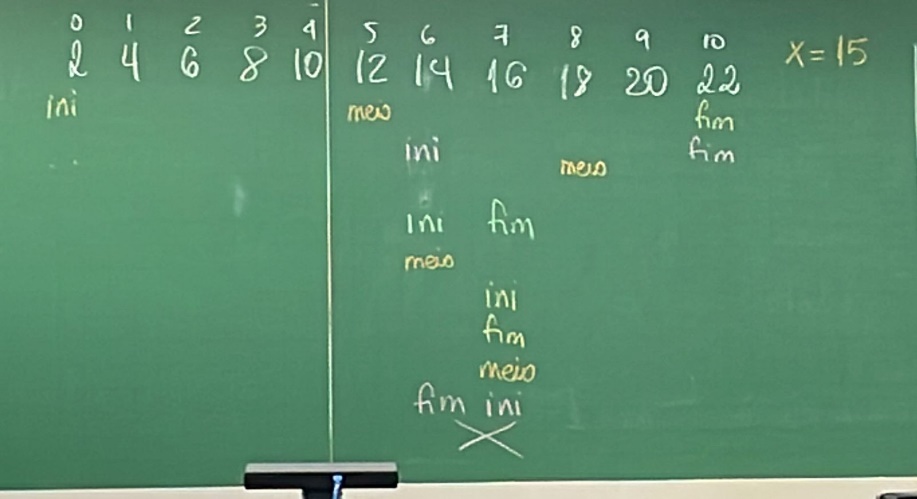
0 – 10

6 – 10

6 – 7

7 – 7 achamos aqui

Oque acontece quando o elemento procurado não está na lista?



Como foi a redução do tamanho da luta?

0 – 10

6 – 10

6 – 7

7 – 7

7 – 6 – não encontrado!

A busca binária

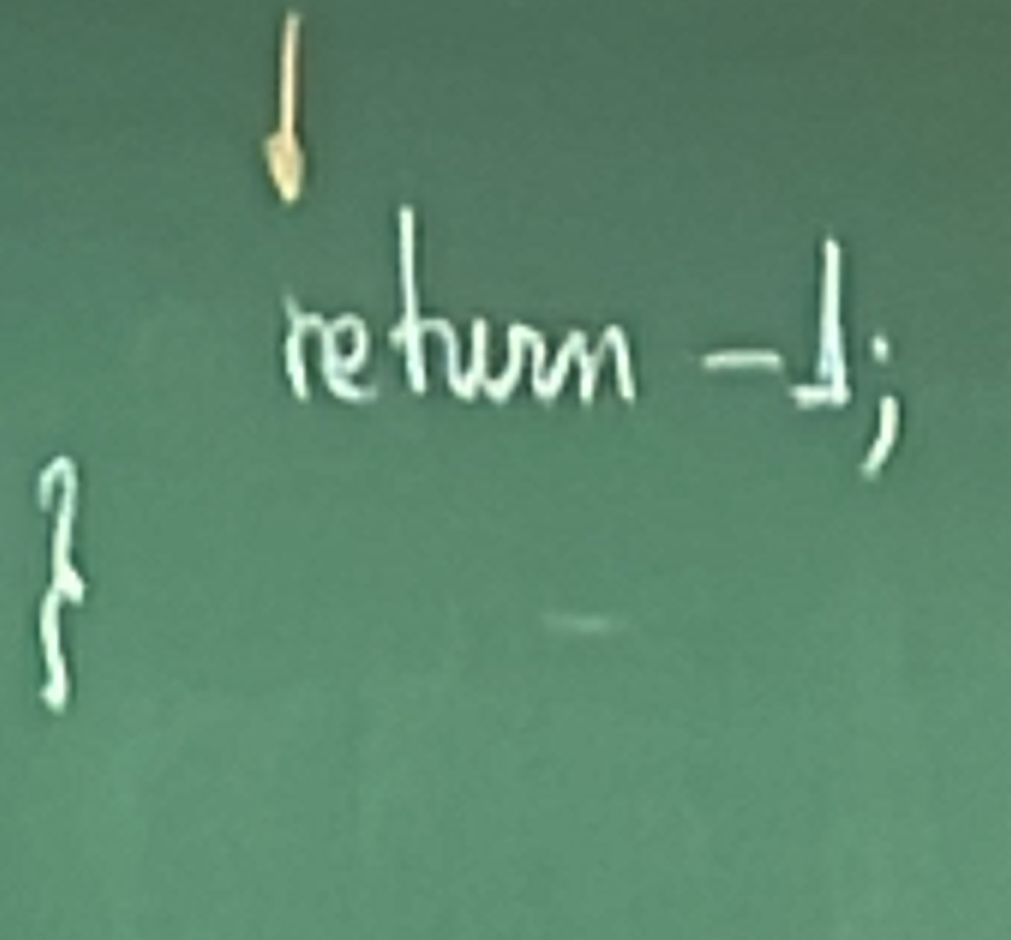
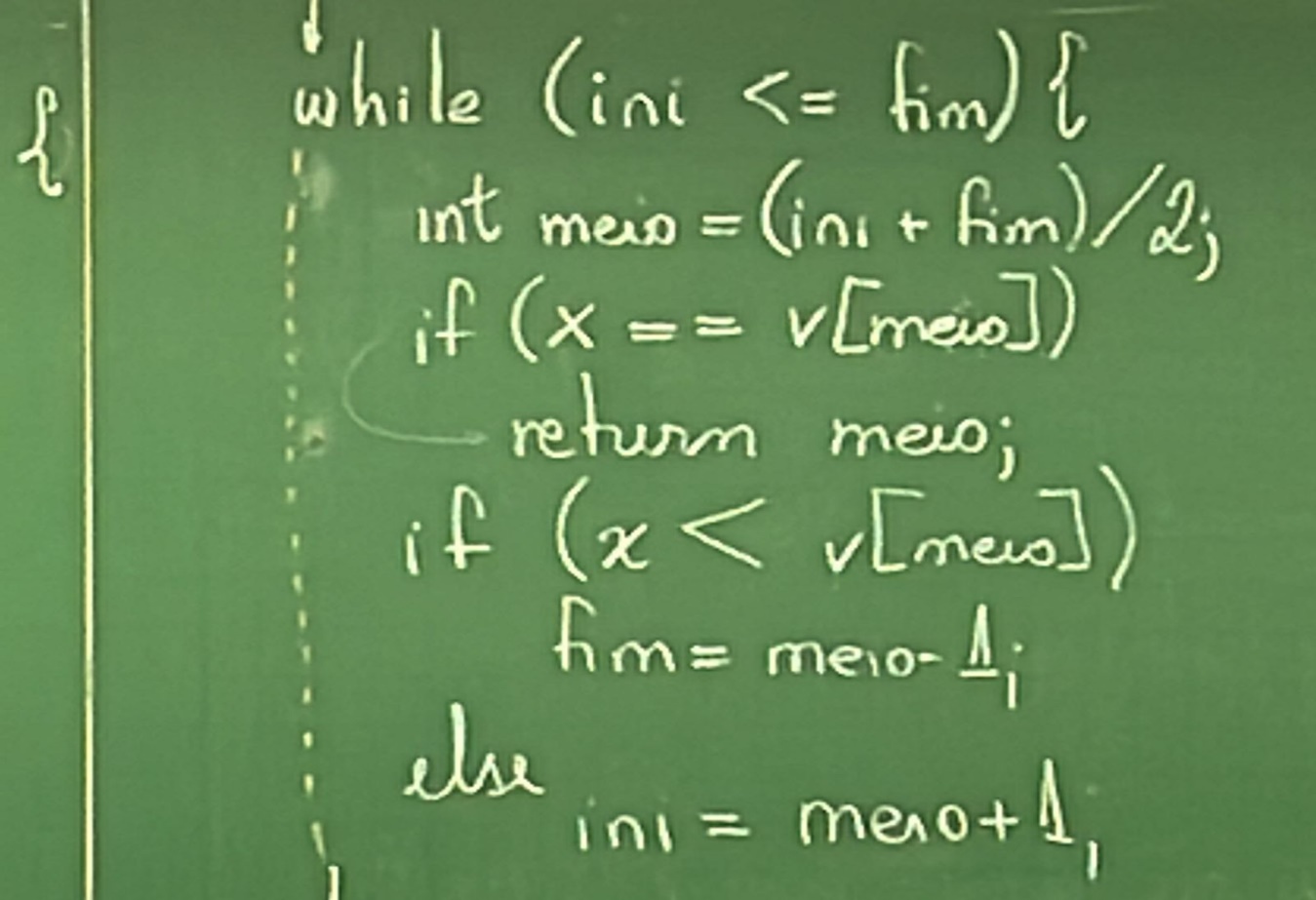
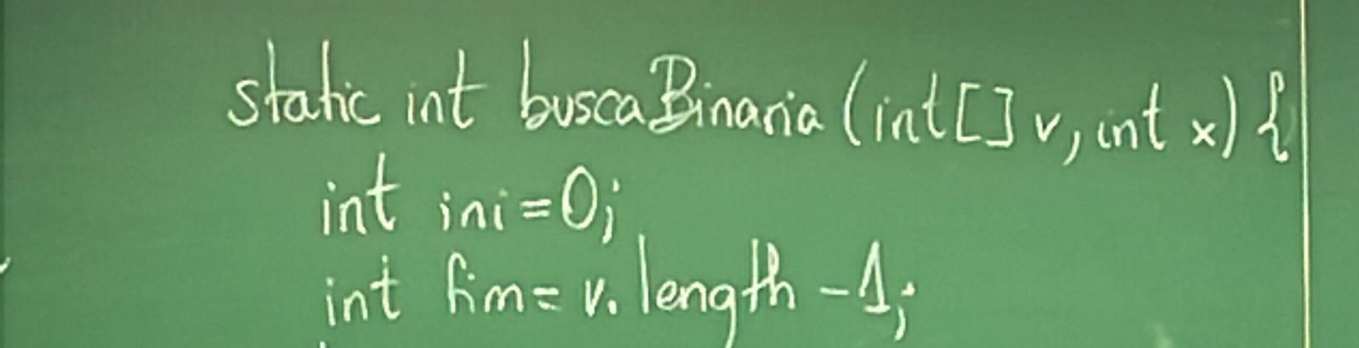
Entrada: um vetor



um valor



Sáida: uma posição em que x aparece ou -1 caso encontrado.



A variação do tempo da execução da busca binária é logarítimica em relação ao tamanho do vetor