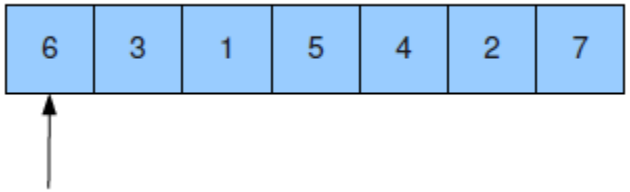
Busca Binária

Prof. Karl Apaza Agüero

Busca

- O objetivo de uma busca é recuperar um registro de um vetor a partir de uma chave (elemento buscado)
- Busca sequencial
  - É o tipo de busca mais simples
  - Não requer ordenação dos elementos
  - Pesquisa sequencialmente do primeiro até o último registro.
  - Quando encontrar a chave desejada, para.

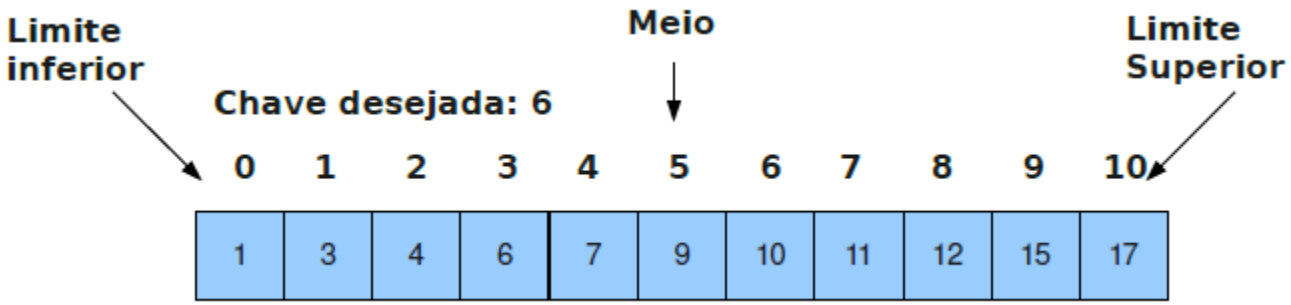


Busca Binária

- Busca mais eficiente que a busca sequencial
- Considera que os elementos estão ordenados no vetor
- Utiliza o limite superior (direita) e inferior (esquerda) para encontrar o meio do vetor

Meio = (limite inferior + limite superior)/2

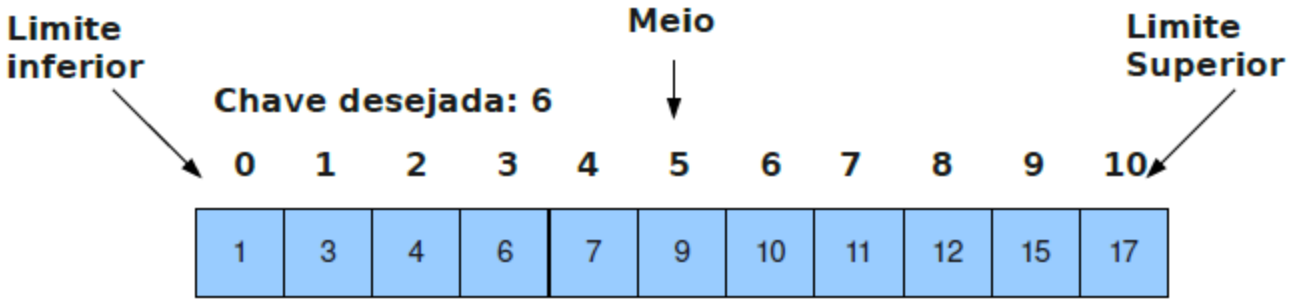
- Exemplo:



Busca Binária

- Compara a chave desejada ao elemento central na forma:

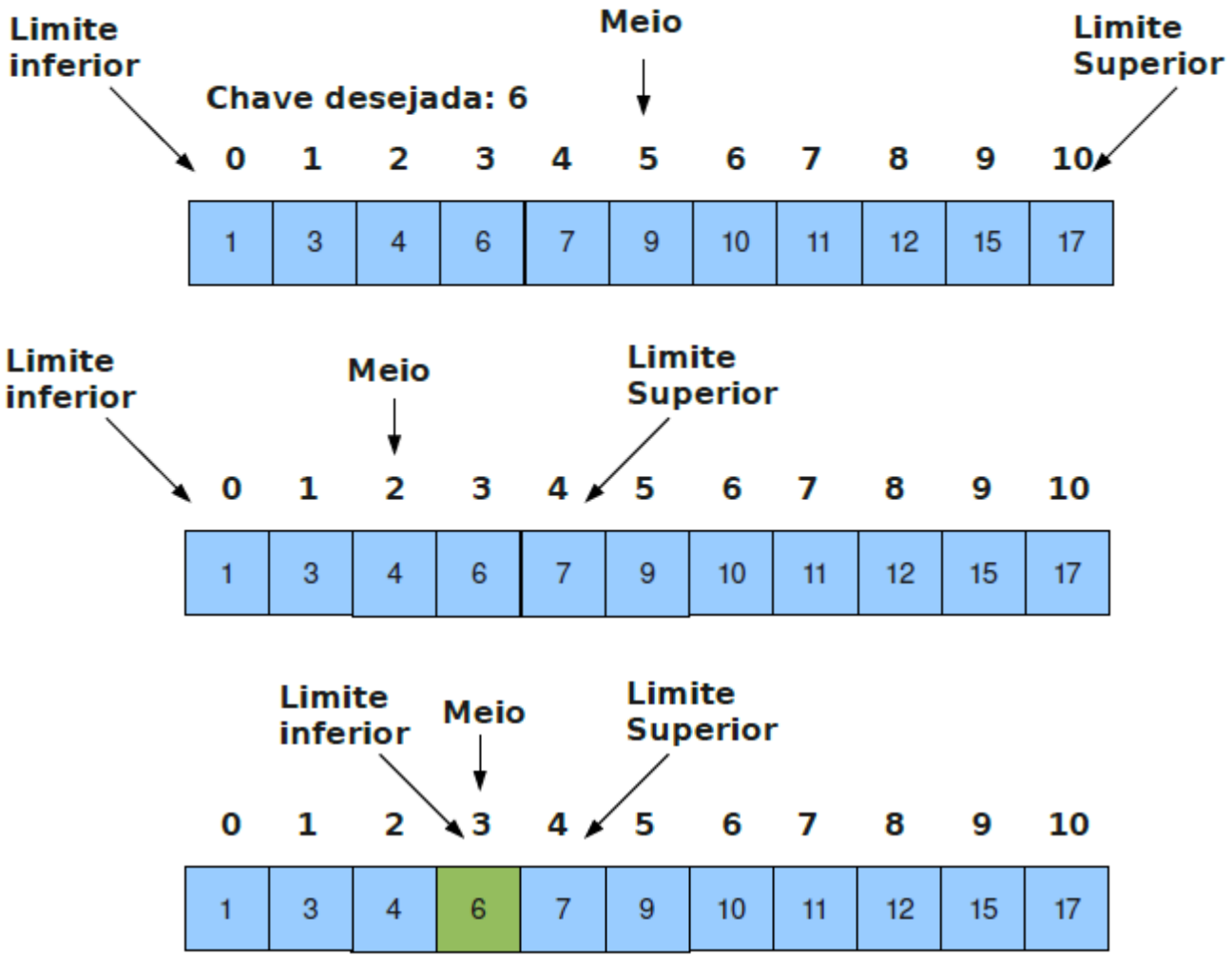
  if (chave<vetor[meio]) limite superior = meio-1
  else if (chave>vetor[meio]) limite inferior = meio+1
  else, se igual, encontrou a chave



- O processo é repetido até encontrar a chave ou enquanto o limite inferior<=limite superior

Busca Binária

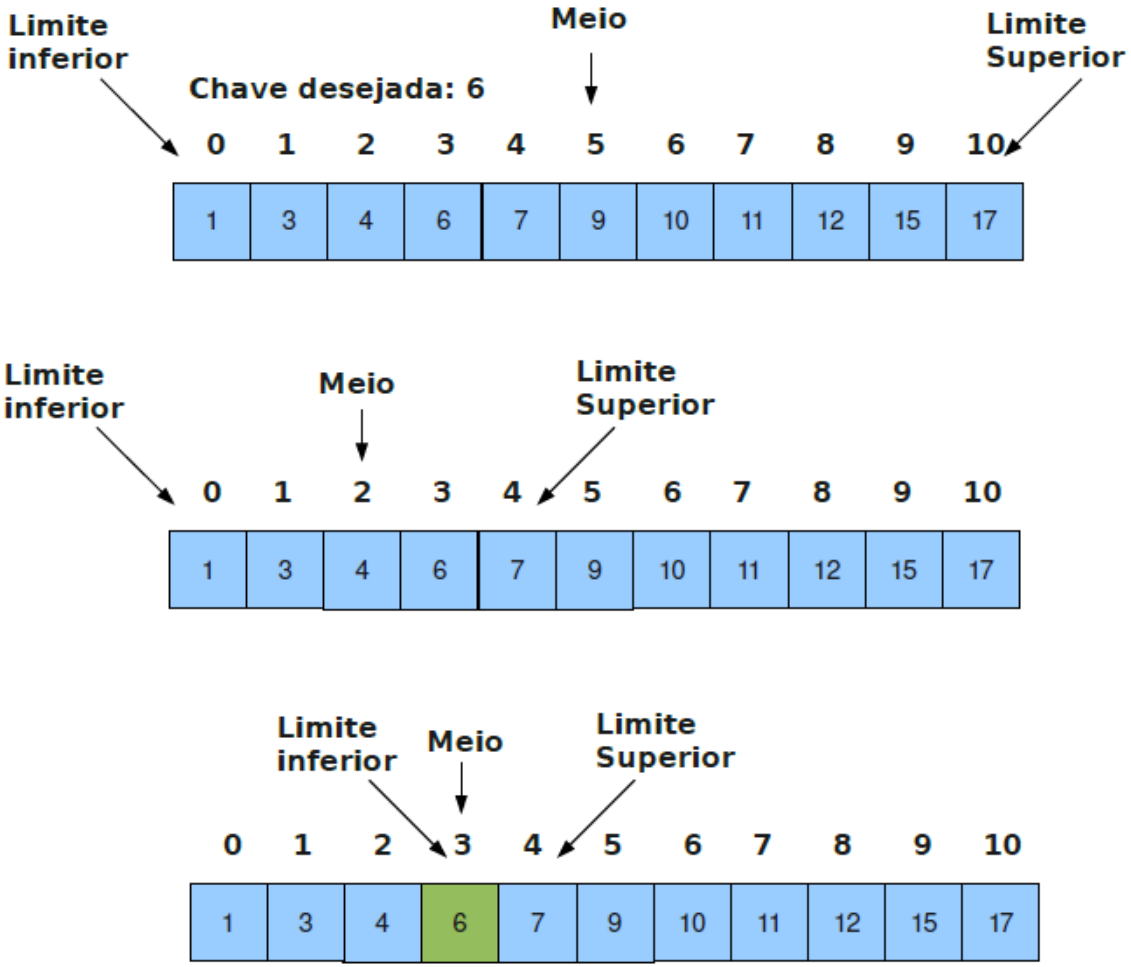
- Exemplo: busca da chave 6



Exemplo

- Descrição
  - Buscar se um número está no vetor
- Entrada
  - Um inteiro representando o número procurado, um inteiro N representando o tamanho do vetor e N elementos inteiros do vetor em ordem crescente
- Saída
  - “SIM” se o número está no vetor, “NAO” caso contrário.

Busca Binária




```
#include <iostream>
using namespace std;
int main(){
  int chave, n;
  cin>>chave>>n;
  int v[n];
  for(int i=0; i<n; i++)
    cin>>v[i];

  int li=0, ls=n-1, meio;
  while(li<=ls){
    meio = (li+ls)/2;
    if(chave<v[meio]) ls=meio-1;
    else if(chave>v[meio]) li=meio+1;
    else break;
  }

  if(chave==v[meio])
    cout<<"SIM"<<endl;
  else
    cout<<"NAO"<<endl;

  return 0;
}
```

Problema

- Descrição
  - Checar se um conjunto de números está no vetor.
- Entrada
  - Um inteiro N representando o tamanho do vetor, N inteiros do vetor em ordem crescente, um inteiro M indicando a quantidade de casos de teste, M inteiros como casos de teste.
- Saída
  - “SIM” quando um caso de teste está no vetor, “NAO” caso contrário.

Entrada
5
1 3 6 7 8
3
2 6 8

Saída
NAO
SIM
SIM

Busca Binária

```
#include <iostream>
using namespace std;
int main(){
  int n;
  cin>>n;
  int v[n];
  for(int i=0; i<n; i++)
    cin>>v[i];
  int m, chave;
  cin>>m;
  for(int i=0; i<m; i++){
     cin>>chave;

     int li=0, ls=n-1, meio;
     while(li<=ls){
       meio = (li+ls)/2;
       if(chave<v[meio]) ls=meio-1;
       else if(chave>v[meio]) li=meio+1;
       else break;

     }

     if(chave==v[meio])
       cout<<"SIM"<<endl;
     else
       cout<<"NAO"<<endl;
  }

  return 0;
}
```

Problema

Os colegas de laboratório de João decidiram que dariam a ele um presente. Porém, como João não gosta de quase nada, eles resolveram dar a ele um vetor. Cada um dos N colegas de João vai contribuir com um número. Porém, João gosta de vetores que contenham a maior quantidade de números possíveis com uma determinada propriedade: se dado um número X do vetor, o número 2X ou 3X estiverem no vetor, o número X aumentará a satisfação de João. Dito isso, cada um dos N colegas contribuíram com um número e formaram um vetor. Eles agora querem saber quantos números desse vetor atendem a propriedade para descobrir o quão feliz João ficará.

Entrada: A primeira linha da entrada possui um inteiro N , indicando o número de elementos do vetor. Na próxima linha haverão N inteiros, em ordem crescente e separados por um espaço em branco: os números dados por cada um dos colegas de João.

Saída: A saída deve conter um único inteiro numa única linha: a quantidade de números que atendem a propriedade.

Entrada
5
1 2 5 8 10

Saída
2

Entrada
8
1 2 4 12 36 71 500 1500

Saída
5

Busca Binária

```
#include <iostream>
using namespace std;

int main() {
  int n; cin>>n;
  int v[n], cont=0;

  for(int i=0; i<n; i++) cin>>v[i];

  for(int i=0; i<n-1; i++){
    int dobro=2*v[i], triplo=3*v[i];
    int li=i+1, ls=n-1, m;
    while(li<=ls){
      m=(li+ls)/2;
      if(dobro<v[m]) ls=m-1;
      else if(dobro>v[m]) li=m+1;
      else break;
    }
    if(dobro==v[m]) cont++;
    else{
      li=i+1, ls=n-1;
      while(li<=ls){
        m=(li+ls)/2;
        if(triplo<v[m]) ls=m-1;
        else if(triplo>v[m]) li=m+1;
        else break;
      }
      if(triplo==v[m]) cont++;
    }
  }
  cout<<cont<<endl;

  return 0;
}
```

Problema

Recentemente Juquinha ganhou de aniversário um joguinho bem clássico: Tiro ao Alvo. Segundo as regras, o alvo do jogo é composto por *C* círculos, todos centrados na origem (0,0). Juquinha atira *T* vezes e após cada tiro informa suas coordenadas. A pontuação de cada tiro é feita da seguinte forma: para cada círculo em que o tiro estiver contido Juquinha recebe um ponto. Considere por exemplo as coordenadas dos 10 tiros na figura mostrada, a pontuação total de Juquinha é de 13 pontos = 3p(0 0) + 2p(-2 0) + 2p(0 -2) + 1p(3 -4) + 1p(-4 -3) + 1p(3 1) + 0p(6 2) + 1p(-1 2) + 0p(-5 -2) + 2p(1 -1)

Entrada

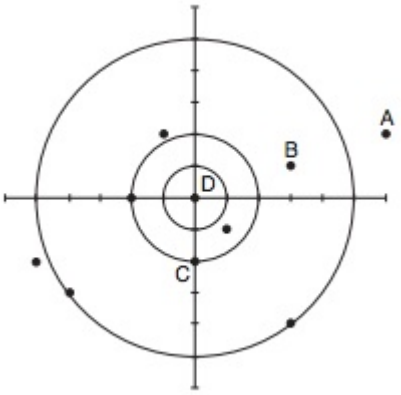
A primeira linha da entrada contém dois inteiros positivos, *C* e *T*, que representam, respectivamente, o número de círculos do alvo e o número de tiros.

Cada uma das *C* linhas seguintes contém um inteiro positivo. O *i*-ésimo inteiro R_i representa o raio do *i*-ésimo círculo. Os raios R_i são fornecidos em ordem crescente.

Cada uma das *T* linhas seguintes contém um par *X*, *Y* de inteiros, que representam as coordenadas de cada tiro.

Saída

Seu programa deve imprimir uma única linha, contendo apenas um inteiro, o total de pontos obtidos por Juquinha.



Entrada

3 10
1
2
5
0 0
-2 0
0 -2
3 -4
-4 -3
3 1
6 2
-1 2
-5 -2
1 -1

Saída

13