RICERCA DICOTOMICA

Cos'è la ricerca dicotomica

La **ricerca dicotomica** (o **ricerca binaria**) è un algoritmo che permette di cercare un valore all'interno di un array ordinato, riducendo progressivamente l'intervallo di ricerca. A differenza della ricerca sequenziale, che controlla un elemento alla volta, la ricerca dicotomica:

- confronta il valore cercato con l'elemento centrale dell'array;
- elimina metà degli elementi ad ogni passo;
- continua a dimezzare lo spazio di ricerca finché trova il valore o l'intervallo si esaurisce.

⚠ Condizione fondamentale: **l'array deve essere ordinato**.

Quando si usa

La ricerca dicotomica è particolarmente adatta quando:

- l'array è ordinato;
- il numero di elementi è elevato;
- si vuole un algoritmo molto efficiente.

Non è adatta quando:

- l'array non è ordinato;
- l'insieme dei dati è molto piccolo (dove la differenza di prestazioni è minima).

Funzionamento passo dopo passo

Dato:

- un array ordinato di `n` elementi;
- un valore `x` da cercare;

si procede così:

- si considera l'intervallo iniziale (inizio = 0, fine = n - 1);
- si calcola la posizione centrale: `medio = (inizio + fine) / 2`
- si confronta x con l'elemento centrale:
  - se sono uguali → trovato;
  - se `x` è minore → si continua nella metà sinistra;
  - se `x` è maggiore → si continua nella metà destra;
- si ripete finché:
  - si trova il valore, oppure
  - l'intervallo diventa vuoto.

Esempio

Array ordinato: `[2, 5, 8, 12, 16, 23, 38]`
Valore da cercare: `23`
Passaggi:

- medio = `12` → `23 > 12` → cerco a destra

- medio = 23 → trovato

La ricerca termina dopo pochissimi confronti.

Complessità

- **Caso migliore: O(1)** (il valore è al centro)
- **Caso peggiore: O(log n)**
- **Caso medio: O(log n)**

La ricerca dicotomica è molto più efficiente della ricerca sequenziale per grandi quantità di dati.

Ricerca dicotomica in C++

Versione senza funzione

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

int main() {
    int n = 7;
    int array[n] = {2, 5, 8, 12, 16, 23, 38};
    int x;
    bool trovato = false;

    cout << "Inserisci il valore da cercare: ";
    cin >> x;

    int inizio = 0;
    int fine = n - 1;

    while (inizio <= fine) {
        int medio = (inizio + fine) / 2;

        if (array[medio] == x) {
            cout << "Valore trovato in posizione " << medio << endl;
            trovato = true;
            break;
        }
        else if (x < array[medio]) {
            fine = medio - 1;
        }
        else {
            inizio = medio + 1;
        }
    }

    if (!trovato) {
        cout << "Valore non trovato" << endl;
    }

    return 0;
}
```

Commenti al codice

- `inizio` e `fine`: delimitano l'intervallo di ricerca;
- `medio`: posizione centrale dell'intervallo;
- `while (inizio <= fine)`: continua finché l'intervallo è valido;

- `array[medio] == x`: verifica se il valore è stato trovato;
- `fine = medio - 1`: restringe a sinistra;
- `inizio = medio + 1`: restringe a destra;
- `trovato`: indica se la ricerca ha avuto successo.

Versione con funzione

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

int ricercaDicotomica(int a[], int n, int x) {
    int inizio = 0;
    int fine = n - 1;

    while (inizio <= fine) {
        int medio = (inizio + fine) / 2;

        if (a[medio] == x) {
            return medio;  // posizione trovata
        }
        else if (x < a[medio]) {
            fine = medio - 1;
        }
        else {
            inizio = medio + 1;
        }
    }

    return -1; // valore non trovato
}

int main() {
    int array[] = {2, 5, 8, 12, 16, 23, 38};
    int n = 7;
    int x;

    cout << "Inserisci il valore da cercare: ";
    cin >> x;

    int pos = ricercaDicotomica(array, n, x);

    if (pos != -1) {
        cout << "Valore trovato in posizione " << pos << endl;
    } else {
        cout << "Valore non trovato" << endl;
    }

    return 0;
}
```

Vantaggi e svantaggi

Vantaggi

- molto efficiente per grandi quantità di dati;
- riduce drasticamente il numero di confronti;
- fondamentale nello studio degli algoritmi efficienti.

Svantaggi

- funziona solo con array ordinati;
- leggermente più complessa da comprendere rispetto alla ricerca sequenziale;
- se i dati non sono ordinati, serve prima un algoritmo di ordinamento.