Curs Lot Național Juniori
Căutarea ternară

Ștefan Dăscălescu

29 Iulie 2023

1 Introducere

Un algoritm de căutare ternară este o tehnică în informatică pentru a găsi minimul sau maximul unei funcții unimodale (care are un singur punct/interval cu valoare maximă/minimă).

Spre deosebire de alte căutări similare precum căutarea binară, căutarea ternară este utilă pentru a afla dacă valoarea extremă nu se găsește în prima sau ultima treime a spațiului de căutare, algoritmul repetându-se pentru celelalte două treimi ale intervalului căutat.

Căutarea ternară este un exemplu de algoritm de tipul divide et impera, la fel ca și căutarea binară sau alți algoritmi similari.

2 Funcția în sine

Pentru a putea aplica căutarea ternară, trebuie ca funcția să fie (des)crescătoare (de regulă, strict) până la un punct x unde găsim cea mai mică poziție care ne dă valoarea maximă/minimă a funcției, urmând că funcția să fie mai apoi constantă până la un punct y, iar apoi funcția va avea monotonia opusă față de cea până la punctul x.

Cu alte cuvinte, funcția crește până la punctul x, apoi e constantă în intervalul $[x, y]$, iar apoi scade de la punctul y. Similar, putem spune și pentru cazul opus al unei funcții unimodale.

3 Algoritmul standard

Să presupunem că avem o funcție f care este definită pe intervalul $[a, b]$. Asemănător căutării binare, vom începe prin a căuta acel punct extrem pe întreg intervalul. La fiecare pas, vom lua punctele $m1$ și $m2$, care vor fi situate la $\frac{1}{3}$ respectiv $\frac{2}{3}$ de capătul din stânga al intervalului, iar în funcție de valorile $f(m1)$ și $f(m2)$, avem următoarele cazuri, acestea fiind similare și pentru o funcție mai întâi descrescătoare.

- Dacă $f(m1) < f(m2)$, maximul nu poate fi în intervalul $[a, m1]$ deoarece $f(m2)$ este mai mare decât $f(m1)$.
- Dacă $f(m1) > f(m2)$, maximul nu poate fi în intervalul $[m2, b]$ deoarece $f(m1)$ este mai mare decât $f(m2)$.
- Dacă $f(m1) = f(m2)$, maximul nu poate fi decât în intervalul $[m1, m2]$ deoarece $f(m1)$ și $f(m2)$ sunt de părți opuse ale punctului sau punctelor maxime.

După ce am redus căutarea la o lungime suficient de mică pentru a preveni erori de precizie, putem trata intervalul rămas folosind brute force pentru a ajunge la răspunsul dorit. Alternativ, putem apela algoritmul de un numar finit de ori, similar cu modul în care am rula căutarea binară pe numere reale.

Complexitatea algoritmului este $O(\log n)$ unde n este dimensiunea intervalului de căutare. Se poate remarca faptul că spre deosebire de căutarea binară, constanta este una mai mare deoarece în medie reducem intervalul de căutare cu $\frac{1}{3}$ la un pas.

```cpp
// f(i) este o func ie oarecare
long long ternara(int precizie)
{
    int st = 0;
    int dr = 1000000000;
    long long ans = -(1LL<<60);
    while(st <= dr)
    {
        int mid1 = st + (dr - st) / 3;
        int mid2 = dr - (dr - st) / 3;
        if(dr - st + 1 <= precizie)
        {
            for(int i = st; i <= dr; ++i)
                ans = max(ans, f(i));
            break;
        }
        ll xa = f(mid1);
        ll xb = f(mid2);
        if(xa == xb)
            st = mid1, dr = mid2;
        else
            if(xa < xb)
                st = mid1;
            else
                dr = mid2;
        ans = max(ans, xa);
        ans = max(ans, xb);
    }
    return ans;
}
```

4 Golden Section Search

Pe lângă căutarea ternară, putem folosi pentru a optimiza procesul de căutare și căutarea Golden Section, care spre deosebire de căutarea ternară, imparte intervalul astfel încât cele două valori de mijloc să fie puse la distanta $r = \frac{3-\sqrt{5}}{2}$, distantă care este egală cu $\frac{1}{\phi}$, unde ϕ este numărul de aur, egal cu aproximativ 1.618.

```python
import math

gr = (math.sqrt(5) + 1) / 2


def gss(f, a, b, tol=1e-5):
    """Golden-section search
    to find the minimum of f on [a,b]
    f: a strictly unimodal function on [a,b]

    Example:
    >>> f = lambda x: (x-2)**2
    >>> x = gss(f, 1, 5)
    >>> print("%.15f" % x)
    2.000009644875678

    """
    c = b - (b - a) / gr
    d = a + (b - a) / gr
    while abs(b - a) > tol:
        if f(c) < f(d):  # f(c) > f(d) to find the maximum
            b = d
        else:
            a = c

        # We recompute both c and d here to avoid loss of precision which may lead to incorrect results or infinite loop
        c = b - (b - a) / gr
        d = a + (b - a) / gr

    return (b + a) / 2
```

5 Problema exemplu - Devu and his Brother

Devu and his Brother

În această problemă avem doi vectori a și b și putem crește/scădea o valoare dintr-unul din cei doi vectori cu costul 1. Vrem să aflăm costul minim pentru ca minimul din vectorul a să fie cel puțin egal cu maximul din vectorul b.

Se poate observa că e clar că vrem să creștem valorile din a și să scădem valorile din b. De asemenea, se poate observa că pe măsură ce vrem să aducem răspunsul la o oarecare "mediană", costul va fi tot mai mic. Aceste lucruri ne duc spre o abordare bazată pe o căutare ternară a răspunsului.

Astfel, vom căuta ternar răspunsul în intervalul $[1, 10^9]$ răspunsul aplicând algoritmul descris mai sus.

```
// f(i) este o func ie oarecare
long long ternara(int precizie)
{
    int st = 0;
    int dr = 1000000000;
    long long ans = -(1LL<<60);
    while(st <= dr)
    {
        int mid1 = st + (dr - st) / 3;
        int mid2 = dr - (dr - st) / 3;
        if(dr - st + 1 <= precizie)
        {
            for(int i = st; i <= dr; ++i)
                ans = max(ans, f(i));
            break;
        }
        ll xa = f(mid1);
        ll xb = f(mid2);
        if(xa == xb)
            st = mid1, dr = mid2;
        else
            if(xa < xb)
                st = mid1;
            else
                dr = mid2;
        ans = max(ans, xa);
        ans = max(ans, xb);
    }
    return ans;
}
```

6 Probleme suplimentare

1. CF 439D
2. copii3 infoarena
3. CEOI 2017 - Sure Bet
4. Solutia de la copii3 - infoarena
5. CF 1355 E
6. CF 1848 D
7. CCO 18-Gradient Descent

7 Bibliografie și lectură suplimentară

1. Ternary Search - CP Algorithms
2. Ternary Search on Integers - Codeforces
3. Tutorial On Tof (Ternary Search) - Codeforces
4. Solutia de la copii3 - infoarena
5. Ternary Search - Wikipedia
6. Numerical Methods with Programming - Golden Section Search
7. Golden Section Search - Wikipedia

8 Cuvânt de încheiere

Dacă aveți sugestii, păreri, întrebări sau altceva ce ați vrea să împărtășiți cu privire la articol sau la alte aspecte legate de conținuturile prezentate, mă puteți contacta pe Discord (stefdasca).