DICHOTOMOUS SEARCH

B PRAJEETH | 205002062

What is Dichotomous search?

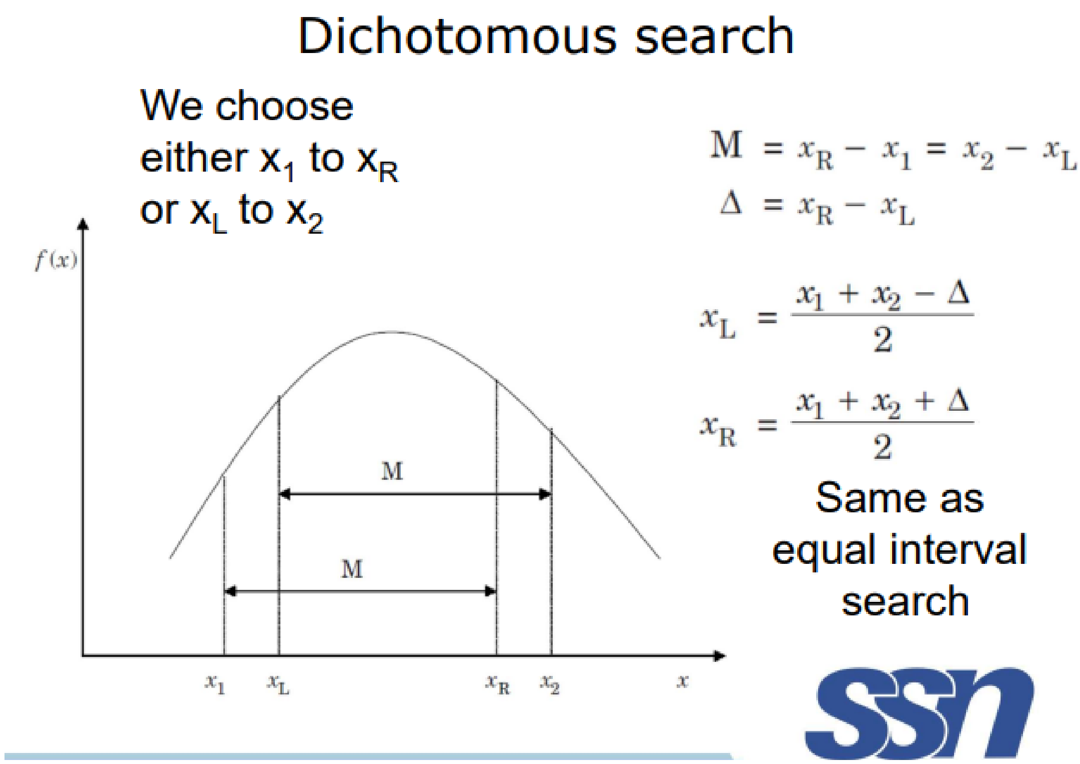
As the name indicates, Dichotomous search refers to algorithmic procedures that search for a target in an unknown location within an interval (the interval of uncertainty, or the search interval) by repeatedly dividing the interval into two parts.

At each iteration, the searcher selects a point in the search interval and places there a *query*, determining at which side of the chosen point the target is located.

This approach is ubiquitous and it is applied naturally not just by sophisticated scientists but also in everyday intuitive trial-and-error experimentation.

In the simplest form of dichotomous search, the searcher has no prior information on where the target is located (or assumes it is uniform over the interval of uncertainty), and the goal is to minimize the worst-case or expected cost of the search.

Dichotomous search is one among many popular search techniques to find the optimal solution.



Here we are considering a unimodal function.

Here X_R and X_L are the left-most and right-most range values for the given function.Every time this range reduces and we move a bit closer to the objective or aimed result, i.e minimization or maximization.

Δ also known as delta is helpful in checking which way the range should converge in each iteration.

Using the above formula we can converge the range at each point.

After certain iterations, we can conclude the minimum value to be in = (X1 + X2 / 2)

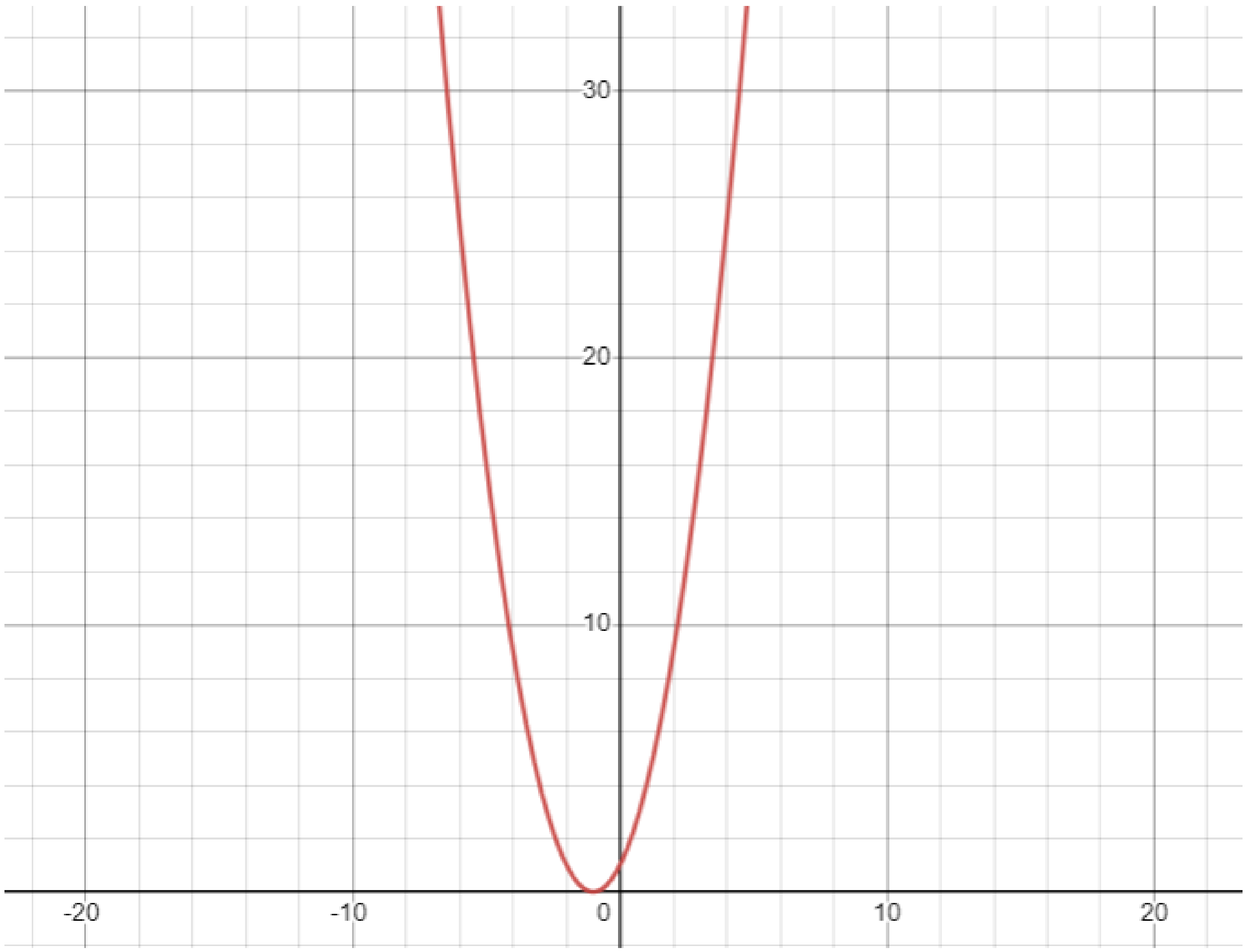
Python Program for the dichotomous search

```
#DICHOTOMOUS SEARCH
#B PRAJEETH
#205002062
import matplotlib.pyplot as plt
import random
import numpy as np


def plot(coeff,xl,xr,type):
    x = np.linspace(-20, 20, 200)
    plt.plot(x,objective(coeff,x))  # the graph
    plt.plot(xl,objective(coeff,xl),"hm",ms="15") #initial left
point
    plt.plot(xr,objective(coeff,xr),"hm",ms="15") #initial right
point
    iter=20
    f=open("dichotomous.txt","w")

    for i in range(iter):  #total of 15 iteraitions
        delta=(xr-xl)/iter
        mid=(xl+xr)/2
        x2=mid+delta/2  #right side
        x1=mid-delta/2 #left side
        y1=objective(coeff,x1)
        y2=objective(coeff,x2)
        txt=str((i+1))+"th iteration"
        f.write(txt)
        txt="\nXl = "+str(xl)+"\nXr = "+str(xr)+""
        f.write(txt)
        txt="\nx1 = "+str(x1)+"\nx2 = "+str(x2)
        f.write(txt)
        txt="\nf(x1) = "+str(y1)+"\nf(x2) = "+str(y2)+"\n"
        f.write(txt)
        plt.plot(x2,y2,"ob",ms="8")  #the right point
        plt.plot(x1,y1,"*b",ms="8")  #the left point
        if(type == 1):
            if(y1>y2):
                xl=x1
                f.write("f(x1) > f(x2)\n\n")
            else:
```

```
                xr=x2
                f.write("f(x1) < f(x2)\n\n")
        elif(type == 2):
            if(y1>y2):
                xr=x2
                f.write("f(x1) > f(x2)\n\n")
            else:
                xl=x1
                f.write("f(x1) < f(x2)\n\n")

    finalx=(xl+xr)/2
    finaly=objective(coeff,finalx)
    plt.plot(finalx,finaly,"sy",ms="15")
    f.close()
    plt.show()


def objective(coeff,val):
    #start=float(input("enter start "))
    sum=0
    for i in range(len(coeff)):
        sum=sum+(coeff[i]*(val**i))
    return sum


n=int(input("enter the degree of function = "))
coeff=[]
for i in range(n+1):
    print("Enter coeff of x^",i ," = ",end="")
    c=float(input())
    coeff.append(float(c))

type=int(input("Enter 1. minimisation 2. maximisation (1/2))=
"))
xl=random.randint(-20,20)  #random xl left
xr=random.randint(xl,20)  #random xr right
print(" xl initial = ",xl)
print(" xr initial = ",xr)
#delta=float(input("Enter the delta value = "))
plot(coeff,xl,xr,type) #have set delta to be xl-xr / total
iterations
```

Let us consider the example: $f(x) = X^2+2X+1$ which we want to minimize.

$f(x) = X^2+2X+1$ curve plotted below



The initial range is always random and iteration further is based upon the formulae mentioned above.

I have considered $\Delta=(X_R-X_L$ / total iterations) and have considered 20 iterations. Always the more iterations the better!

Having the initial X_L = -4 and X_R=7 which was randomly chosen we have collected the given data for 20 iterations for the given function, f(x) = X^2+2X+1

At each iteration, each of these ones is done $X_L$to X_2 or X_1 to X_R

Here our aim is to find the minimum value for the provided objective function.

DATA COLLECTED AT EACH ITERATION

1st iteration
Xl = -4
Xr = 7
x1 = 1.225
x2 = 1.775
f(x1) = 4.950625000000005
f(x2) = 7.700625
f(x1) < f(x2)

2nd iteration
Xl = -4
Xr = 1.775
x1 = -1.256875
x2 = -0.968125
f(x1) = 0.06598476562499989
f(x2) = 0.0010160156249999774
f(x1) > f(x2)

3rd iteration
Xl = -1.256875
Xr = 1.775
x1 = 0.183265625

x2 = 0.33485937499999996
f(x1) = 1.4001175393066405
f(x2) = 1.7818495510253904
f(x1) < f(x2)

4th iteration
Xl = -1.256875
Xr = 0.33485937499999996
x1 = -0.500801171875
x2 = -0.421214453125
f(x1) = 0.24919947000137332
f(x2) = 0.3349927092713928
f(x1) < f(x2)

5th iteration
Xl = -1.256875
Xr = -0.421214453125
x1 = -0.859936240234375
x2 = -0.8181532128906249
f(x1) = 0.019617856799682754
f(x2) = 0.033068253982002416
f(x1) < f(x2)

6th iteration
Xl = -1.256875
Xr = -0.8181532128906249
x1 = -1.0484821511230467
x2 = -1.0265460617675781
f(x1) = 0.0023505189775179236
f(x2) = 0.0007046933953680501
f(x1) > f(x2)

7th iteration
Xl = -1.0484821511230467
Xr = -0.8181532128906249
x1 = -0.9390759054626464
x2 = -0.9275594585510253
f(x1) = 0.003711745295196356
f(x2) = 0.005247632045420669
f(x1) < f(x2)

8th iteration
Xl = -1.0484821511230467
Xr = -0.9275594585510253
x1 = -0.9910438721513366
x2 = -0.9849977375227355
f(x1) = 8.021222604159828e-05
f(x2) = 0.00022506787943676887
f(x1) < f(x2)

9th iteration
Xl = -1.0484821511230467
Xr = -0.9849977375227355
x1 = -1.0183270546628989
x2 = -1.0151528339828833
f(x1) = 0.0003358809326168277
f(x2) = 0.00022960837771290876
f(x1) > f(x2)

10th iteration
Xl = -1.0183270546628989
Xr = -0.9849977375227355
x1 = -1.0024956290213214
x2 = -1.0008291631643131
f(x1) = 6.2281642121408964e-06

f(x2) = 6.875115530213805e-07
f(x1) > f(x2)

11th iteration
Xl = -1.0024956290213214
Xr = -0.9849977375227355
x1 = -0.9941841305594931
x2 = -0.9933092359845638
f(x1) = 3.382433734899 77e-05
f(x2) = 4.476632311023465e-05
f(x1) < f(x2)

12th iteration
Xl = -1.0024956290213214
Xr = -0.9933092359845638
x1 = -0.9981320923288616
x2 = -0.9976727726770237
f(x1) = 3.4890790678865358e-06
f(x2) = 5.415987012757917e-06
f(x1) < f(x2)

13th iteration
Xl = -1.0024956290213214
Xr = -0.9976727726770237
x1 = -1.00020477225778
x2 = -0.9999636294405652
f(x1) = 4.1931677463580286e-08
f(x2) = 1.3228176332091834e-09
f(x1) > f(x2)

14th iteration
Xl = -1.00020477225778
Xr = -0.9976727726770237

x1 = -0.9990020724569209
x2 = -0.998875472477883
f(x1) = 9.958593811809635e-07
f(x2) = 1.2645621479956404e-06
f(x1) < f(x2)

15th iteration
Xl = -1.00020477225778
Xr = -0.998875472477883
x1 = -0.9995733548623289
x2 = -0.9995068898733341
f(x1) = 1.820260735474477e-07
f(x2) = 2.4315759705739737e-07
f(x1) < f(x2)

16th iteration
Xl = -1.00020477225778
Xr = -0.9995068898733341
x1 = -0.9998732781251682
x2 = -0.9998383840059459
f(x1) = 1.6058433582877285e-08
f(x2) = 2.6119729490403643e-08
f(x1) < f(x2)

17th iteration
Xl = -1.00020477225778
Xr = -0.9998383840059459
x1 = -1.000030737838159
x2 = -1.0000124184255672
f(x1) = 9.448146709445382e-10
f(x2) = 1.5421730559239677e-10
f(x1) > f(x2)

18th iteration
Xl = -1.000030737838159
Xr = -0.9998383840059459
x1 = -0.9999393697678578
x2 = -0.9999297520762471
f(x1) = 3.6760250399225924e-09
f(x2) = 4.934770814202238e-09
f(x1) < f(x2)

19th iteration
Xl = -1.000030737838159
Xr = -0.9999297520762471
x1 = -0.9999827696012509
x2 = -0.9999777203131552
f(x1) = 2.9688662639415497e-10
f(x2) = 4.963844890681912e-10
f(x1) < f(x2)

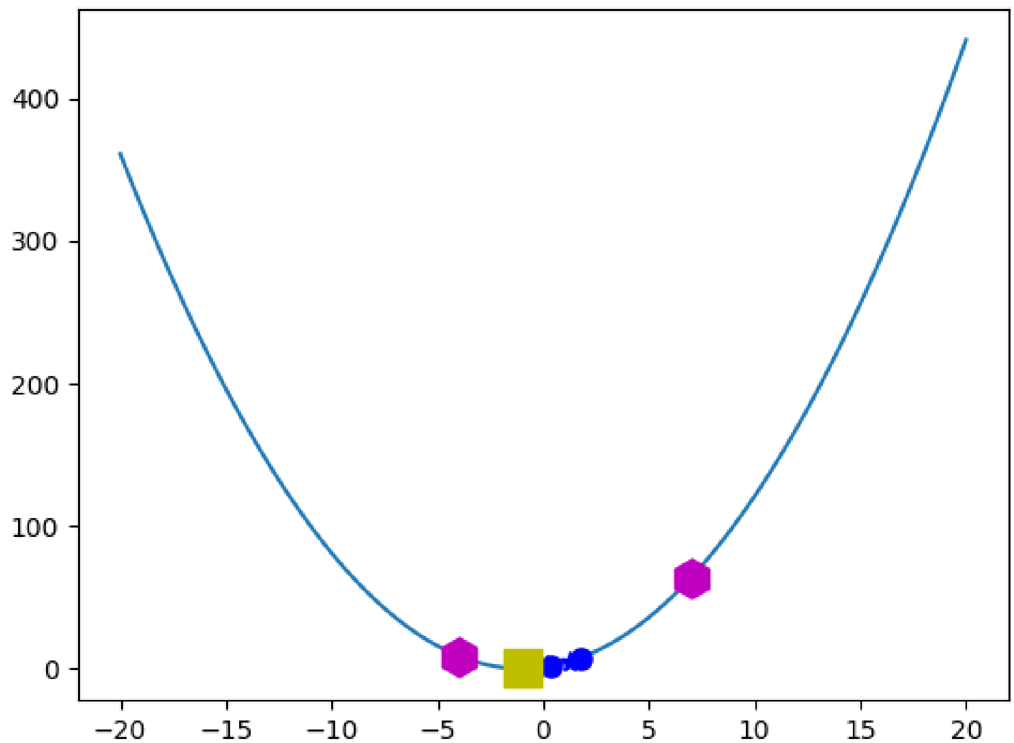
20th iteration
Xl = -1.000030737838159
Xr = -0.9999777203131552

Our final value for the given objective after 20 iterations is

(X_R+ X_L)/2 ~ - 0.9999 which is extremely close to -1(i.e, the minimum value for the given objective function)

The final representation of the iterations is given below.

Here the violet hexagons represent the initial starting points and the yellow square represents the final solution for the minimum value. The other blue points represent the X_R and X_L points for all 20 iterations



What do we infer?

The Dichotomous search is a successful optimization technique and we have found the minimum point in the given unimodal function.

Given the correct starting range for the objective function, we can attain the optimal value. Also increasing the number of iterations also increases the accuracy of the final optimal value.