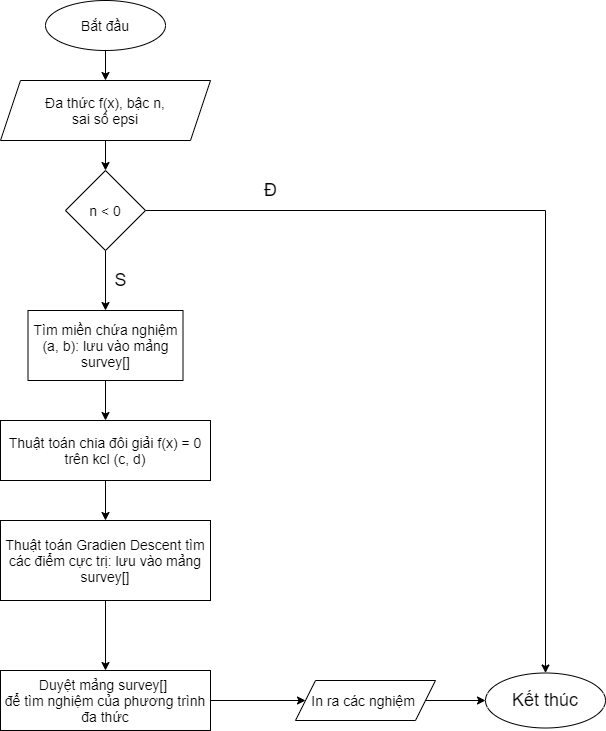
1. **Thuật toán tổng thể**
2. Thuật toán tổng thể chương trình

Input: bậc đa thức n, đa thức f(x), sai số epsi

Output: các nghiệm của phương trình đa thức f(x) = 0



1. **Thuật toán chi tiết**

|  |
| --- |
| 1. Hàm nhập đa thức bậc n: f(x)   input: n  output: mảng a[n] là mảng chứa các hệ số của đa thức f(x)  Function nhap\_DT:  for i = 0 to n:  nhập a[i]   1. Hàm tính giá trị f(x)   input: x, n, a[n]  output: giá trị của f(x) tại x  Function f:  sum = 0  for i = 0 to n:  sum = sum + a[i] \* xn – i  return sum   1. Hàm tính giá trị đạo hàm của f(x)   input: x, n, a[n]  output: giá trị của đạo hàm f(x) tại x là df(x)  Function df:  sum = 0  for i = 0 to n - 1:  sum = sum + a[i] \* (n – i) \* xn – 1 - i  return sum   1. Hàm tìm miền chứa nghiệm của đa thức f(x)   input:n, a[n], địa chỉ low, up  output: miền chứa nghiệm (low, up)  Function domain\_solution:  low = 1, up = -1  for i = 0 to n:  temp[n] = a[n] //mảng tạm thời temp[n] lưu trữ các hệ số  for j = 1 to 2:  max = 0  k = 0  if tem[0] < 0:  for i = 0 to n:  temp[i] = - temp[i]  for i = 1 to n:  if temp[i] < 0:  k = i  thoát khỏi vòng lặp  for i = 1 to n:  if temp[i] < 0:  if |temp[i]| > max:  max = |temp[i]|  if max = 0:  if up = -1:  up = 0  else:  low = 0  else:  if up = -1:  up = 1 + (max / temp[0])1/k  else:  low = -(1 + (max / temp[0])1/k )  for i = 0 to n:  if n chia dư i = 1:  temp[n – i] = - temp[n – i]   1. Thuật toán chia đôi giải phương trình f(x) = 0 trong kcl (c, d)   input: n, a[n], c, d, epsi  output: nghiệm gần đúng x  function bisection:  do:  mid = (c + d) / 2  if f(a, n, mid) = 0:  return mid  if f(a, n, mid) \* f(a, n ,c) > 0:  c = mid  else:  d = mid  denta = |d – c|  while (denta > epsi)  return mid   1. Giải phương trình đa thức f(x) = 0   input: n, a[n], epsi  output: các nghiệm của phương trình đa thức f(x) = 0  Function solve:  eta = e-7  k = 0  survey[n] //Mảng để chứa 2 các cực trị và miền chứa nghiệm  if n = 0:  if a[0] = 0:  print “phương trình đa thức vô số nghiệm”  else:  print “phương trình đa thức vô nghiệm”  if n = 1:  print “Nghiệm duy nhất là: -a[1]/a[0]  domain\_solution(a, n, low, up)  if low = up:  if f(a, n, low) = 0:  print “Nghiệm là:” low  else:  print “Phương trình vô nghiệm”  else:  //Thuật toán Gradient Descent tìm cực trị  x1 = low  x0 = x1  k = 1  while (x0 < up):  if df(f, x0) > 0:  sign = 1  else:  sign = -1  eta = 10-7  x1 = x0 + sign \* eta \*df(f, x0)  while (|df(f, x1)| > epsi):  if df(f, x0) \* df(f, x1) > 0:  while eta < 0.008:  eta = eta \* 2  x1 = x0 + sign \* eta \* df(f, x0)  if df(f, x1) \* df(f, x0) < 0  eta = eta / 2  thoát vòng lặp while  else:  while eta > 0:  eta = eta / 2  x1 = x0 + sign \* eta \*df(f, x0)  if df(f, x1) \* df(f, x0) > 0:  thoát vòng lặp while  x1 = x0 + sign \* eta \* df(f, x0)  x0 = x1  eta = 10-7  Thêm x1 vào mảng survey: survey[k] = x1  x1 = x1 + 0.001  k = k + 1  Thêm low và up vào mảng survey[]: survey[0] = low  survey[k] = up    // Bắt đầu tìm nghiệm của phương trình đa thức  for i = 0 to k – 1:  value1 = f(a, n, survey[i])  value2 = f(a, n, survey[i + 1])  if |value1| <= epsi:  print “nghiệm là: “ survey[i]  if |value2| <= epsi:  print “nghiệm là: “ survey[i+1]  else:  if value1 \* value2 < 0:  print “nghiệm là: “  bisection(a, n, survey[i], survey[i+1], epsi) |

1. **Ưu và nhược điểm của phương pháp giải phương trình đa thức theo thuật toán chia đôi + Gradient Descen với phương pháp chia nhỏ miền chứa nghiệm rồi tính giá trị tại từng điểm.**
2. Ưu điểm

* Độ phức tạp thuật toán thấp hơn
* Tốc độ tìm ra các nghiệm của phương trình đa thức nhanh hơn
* Thuật toán GD có sử dụng eta động giúp việc tìm ra các cực trị nhanh hơn rất nhiều và hiệu quả hơn dùng eta tĩnh

1. Nhược điểm

* Khó cài đặt thuật toán trên máy tính hơn so với phương pháp chia nhỏ miền nghiệm
* Một số đa thức với độ dốc đồ thì quá lớn thì tìm rất lâu và có thể không tìm ra được

1. **Tóm tắt phương pháp**
2. Chú ý: phương pháp Gradient Descent với eta động tìm các cực trị trong khoảng đóng [a, b]

* Dãy lặp để tìm cực trị: x1 = x0 + sign \* eta \*f’(x0).

Với giá trị x0 là điểm xuất phát bất kì: x0 = a

+ sign = 1 nếu đi tìm cực đại ( f’(x) > 0)

+ sign = - 1 nếu đi tìm cực tiểu ( f’(x) < 0)

* Thực hiện lặp để tìm ra cực trị tiếp theo và tính eta động bằng cách:
* Nếu mà f’(x1) \* f’(x0) > 0 và f’(x1) < 0.008 thì ta sẽ tính lại giá trị eta = eta \* 2 và kết thúc vòng lặp khi f’(x1) \* f’(x0) < 0
* Nếu mà f’(x1) \* f’(x0) < 0 => ta sẽ tính lại giá trị eta = eta / 2 và kết thúc vòng lặp khi f’(x1) \* f’(x0) > 0
* Lưu toàn bộ giá trị cực trị và 2 đầu mút của miền chứa nghiệm vào mảng survey[]

1. Cài đặt được thuật toán tìm miền chứa nghiệm

+ upper – bround = 1 + , với max là giá trị âm có trị tuyệt đối lớn nhất, k là chỉ số của phần tử âm đầu tiên.

+ lower – bround = , tương tự như khi đi tìm upper\_bround, nhưng ta thay giá trị của x = - x.

1. Cài đặt thuật toán chia đôi
2. Duyệt mảng survey[] để tìm nghiệm của đa thức

* Duyệt từng giá trị trong mảng survey[]:
* Nếu mà f(survey[i]) = 0 => giá trị đó chính là nghiệm của đa thức
* Nếu mà f(survey[i]) \* f(survey[i+1]) < 0 => 2 giá trị đó thỏa mãn khoảng cách li nghiệm và ta sẽ dùng thuật toán chia đôi để tìm nghiệm của f(x) = 0