Mẫu thuật toán Pascal

Dưới đây là các đoạn mã mẫu cho các thuật toán được liệt kê trong file của bạn, viết bằng Pascal.

---

1. Thuật toán Sắp xếp

Sắp xếp nổi bọt (Bubble Sort)

pascal

```
procedure BubbleSort(var A: array of Integer; N: Integer);
var
  i, j, temp: Integer;
begin
  for i := N - 1 downto 1 do
    for j := 0 to i - 1 do
      if A[j] > A[j + 1] then
      begin
        temp := A[j];
        A[j] := A[j + 1];
        A[j + 1] := temp;
      end;
end;
```

Sắp xếp chọn (Selection Sort)

pascal

```
procedure SelectionSort(var A: array of Integer; N: Integer);
var
  i, j, minIdx, temp: Integer;
begin
  for i := 0 to N - 2 do
  begin
    minIdx := i;
    for j := i + 1 to N - 1 do
      if A[j] < A[minIdx] then
        minIdx := j;
```

```
    temp := A[i];
    A[i] := A[minIdx];
    A[minIdx] := temp;
  end;
end;
```

Sắp xếp chèn (Insertion Sort)

pascal

```
procedure InsertionSort(var A: array of Integer; N: Integer);
var
  i, j, key: Integer;
begin
  for i := 1 to N - 1 do
  begin
    key := A[i];
    j := i - 1;
    while (j >= 0) and (A[j] > key) do
    begin
      A[j + 1] := A[j];
      Dec(j);
    end;
    A[j + 1] := key;
  end;
end;
```

Sắp xếp nhanh (Quick Sort)

pascal

```
procedure QuickSort(var A: array of Integer; L, R: Integer);
var
  i, j, pivot, temp: Integer;
begin
  i := L;
  j := R;
  pivot := A[(L + R) div 2];
  repeat
    while A[i] < pivot do Inc(i);
    while A[j] > pivot do Dec(j);
    if i <= j then
    begin
```

```
      temp := A[i];
      A[i] := A[j];
      A[j] := temp;
      Inc(i);
      Dec(j);
    end;
  until i > j;
  if L < j then QuickSort(A, L, j);
  if i < R then QuickSort(A, i, R);
end;
```

Sắp xếp trộn (Merge Sort)

pascal

```
procedure Merge(var A: array of Integer; L, M, R: Integer);
var
  Left, Right: array of Integer;
  i, j, k, n1, n2: Integer;
begin
  n1 := M - L + 1;
  n2 := R - M;
  SetLength(Left, n1);
  SetLength(Right, n2);

  for i := 0 to n1 - 1 do Left[i] := A[L + i];
  for j := 0 to n2 - 1 do Right[j] := A[M + 1 + j];

  i := 0; j := 0; k := L;
  while (i < n1) and (j < n2) do
  begin
    if Left[i] <= Right[j] then
    begin
      A[k] := Left[i];
      Inc(i);
    end
    else
    begin
      A[k] := Right[j];
      Inc(j);
    end;
    Inc(k);
```

```
  end;

  while i < n1 do
  begin
    A[k] := Left[i];
    Inc(i);
    Inc(k);
  end;

  while j < n2 do
  begin
    A[k] := Right[j];
    Inc(j);
    Inc(k);
  end;
end;

procedure MergeSort(var A: array of Integer; L, R: Integer);
var
  M: Integer;
begin
  if L < R then
  begin
    M := (L + R) div 2;
    MergeSort(A, L, M);
    MergeSort(A, M + 1, R);
    Merge(A, L, M, R);
  end;
end;
```

---

2. Thuật toán Tìm kiếm

Tìm kiếm tuyến tính (Linear Search)

pascal

```
function LinearSearch(A: array of Integer; N, X: Integer): Integer;
var
  i: Integer;
```

```
begin
  for i := 0 to N - 1 do
    if A[i] = X then
    begin
      Result := i;
      Exit;
    end;
  Result := -1;
end;
```

Tìm kiếm nhị phân (Binary Search)

pascal

```
function BinarySearch(A: array of Integer; L, R, X: Integer): Integer;
var
  mid: Integer;
begin
  while L <= R do
  begin
    mid := L + (R - L) div 2;
    if A[mid] = X then Exit(mid);
    if A[mid] < X then
      L := mid + 1
    else
      R := mid - 1;
  end;
  Result := -1;
end;
```

---

3. Thuật toán Số học

Tính giai thừa

pascal

```
function Factorial(N: Integer): Integer;
begin
  if N = 0 then Result := 1
```

```
  else Result := N * Factorial(N - 1);
end;
```

Dãy Fibonacci

- **Đệ quy**

```pascal
function Fibonacci(N: Integer): Integer;
begin
  if N <= 1 then Result := N
  else Result := Fibonacci(N - 1) + Fibonacci(N - 2);
end;
```

- **Không đệ quy**

```pascal
function FibonacciIterative(N: Integer): Integer;
var
  a, b, temp, i: Integer;
begin
  a := 0; b := 1;
  for i := 2 to N do
  begin
    temp := a + b;
    a := b;
    b := temp;
  end;
  Result := b;
end;
```

Ước chung lớn nhất (GCD - Euclid Algorithm)

```pascal
function GCD(A, B: Integer): Integer;
begin
  while B <> 0 do
  begin
    A := A mod B;
```

```
    if A = 0 then Exit(B);
    B := B mod A;
  end;
  Result := A;
end;
```

Các thuật toán còn lại rất dài, bạn có muốn tôi tiếp tục gửi không?

Dưới đây là các thuật toán tiếp theo được viết bằng Pascal.

---

3. Thuật toán Số học (Tiếp tục)

Bội chung nhỏ nhất (LCM)

pascal

```
function LCM(A, B: Integer): Integer;
begin
  Result := (A div GCD(A, B)) * B;
end;
```

Kiểm tra số nguyên tố (Sàng Eratosthenes)

pascal

```
procedure SieveOfEratosthenes(N: Integer);
var
  prime: array of Boolean;
  p, i: Integer;
begin
  SetLength(prime, N + 1);
  for i := 0 to N do prime[i] := True;

  p := 2;
  while p * p <= N do
  begin
```

```
    if prime[p] then
      for i := p * p to N do
        if (i mod p = 0) then prime[i] := False;
    Inc(p);
  end;

  for i := 2 to N do
    if prime[i] then Write(i, ' ');
end;
```

Phân tích số nguyên thành thừa số nguyên tố

pascal

```
procedure PrimeFactorization(N: Integer);
var
  i: Integer;
begin
  i := 2;
  while (N > 1) do
  begin
    while (N mod i = 0) do
    begin
      Write(i, ' ');
      N := N div i;
    end;
    Inc(i);
  end;
end;
```

Số hoàn hảo

pascal

```
function IsPerfectNumber(N: Integer): Boolean;
var
  sum, i: Integer;
begin
  sum := 0;
  for i := 1 to N div 2 do
    if N mod i = 0 then sum := sum + i;
```

```pascal
  Result := sum = N;
end;
```

Số Armstrong

```pascal
function IsArmstrong(N: Integer): Boolean;
var
  temp, sum, digit, numDigits: Integer;
begin
  temp := N;
  sum := 0;
  numDigits := Trunc(Log10(N)) + 1;
  while temp > 0 do
  begin
    digit := temp mod 10;
    sum := sum + Round(Power(digit, numDigits));
    temp := temp div 10;
  end;
  Result := sum = N;
end;
```

Số Palindrome

```pascal
function IsPalindrome(N: Integer): Boolean;
var
  temp, reversed, digit: Integer;
begin
  temp := N;
  reversed := 0;
  while temp > 0 do
  begin
    digit := temp mod 10;
    reversed := reversed * 10 + digit;
    temp := temp div 10;
  end;
  Result := reversed = N;
end;
```

4. Thuật toán Đồ thị

Tìm kiếm theo chiều sâu (DFS)

pascal

```
procedure DFS(v: Integer; var visited: array of Boolean; adj: array of array of
Integer);
var
  i: Integer;
begin
  visited[v] := True;
  Write(v, ' ');
  for i := 0 to High(adj[v]) do
    if (not visited[adj[v][i]]) then
      DFS(adj[v][i], visited, adj);
end;
```

Tìm kiếm theo chiều rộng (BFS)

pascal

```
procedure BFS(start: Integer; var adj: array of array of Integer);
var
  queue: array of Integer;
  visited: array of Boolean;
  front, rear, i, node: Integer;
begin
  SetLength(queue, Length(adj));
  SetLength(visited, Length(adj));

  front := 0;
  rear := 0;
  queue[rear] := start;
  visited[start] := True;

  while front <= rear do
  begin
    node := queue[front];
    Inc(front);
    Write(node, ' ');
```

```
    for i := 0 to High(adj[node]) do
      if not visited[adj[node][i]] then
      begin
        Inc(rear);
        queue[rear] := adj[node][i];
        visited[adj[node][i]] := True;
      end;
  end;
end;
```

Dijkstra - Tìm đường đi ngắn nhất

pascal

```
procedure Dijkstra(var graph: array of array of Integer; src, n: Integer);
var
  dist: array of Integer;
  visited: array of Boolean;
  i, count, u, v, minDist: Integer;
begin
  SetLength(dist, n);
  SetLength(visited, n);

  for i := 0 to n - 1 do
  begin
    dist[i] := MaxInt;
    visited[i] := False;
  end;

  dist[src] := 0;

  for count := 0 to n - 1 do
  begin
    minDist := MaxInt;
    u := -1;
    for i := 0 to n - 1 do
      if (not visited[i]) and (dist[i] < minDist) then
      begin
        minDist := dist[i];
        u := i;
      end;
```

```
    visited[u] := True;

    for v := 0 to n - 1 do
      if (not visited[v]) and (graph[u][v] <> 0) and (dist[u] <> MaxInt) and
         (dist[u] + graph[u][v] < dist[v]) then
        dist[v] := dist[u] + graph[u][v];
  end;

  for i := 0 to n - 1 do
    WriteLn('Khoảng cách từ ', src, ' đến ', i, ' là: ', dist[i]);
end;
```

Thuật toán Floyd-Warshall

pascal

```
procedure FloydWarshall(var graph: array of array of Integer; n: Integer);
var
  dist: array of array of Integer;
  i, j, k: Integer;
begin
  SetLength(dist, n, n);
  for i := 0 to n - 1 do
    for j := 0 to n - 1 do
      dist[i][j] := graph[i][j];

  for k := 0 to n - 1 do
    for i := 0 to n - 1 do
      for j := 0 to n - 1 do
        if (dist[i][k] <> MaxInt) and (dist[k][j] <> MaxInt) and
           (dist[i][k] + dist[k][j] < dist[i][j]) then
          dist[i][j] := dist[i][k] + dist[k][j];

  for i := 0 to n - 1 do
  begin
    for j := 0 to n - 1 do
      if dist[i][j] = MaxInt then
        Write('INF ')
      else
        Write(dist[i][j], ' ');
    Writeln;
```

```
  end;
end;
```

Còn rất nhiều thuật toán như **Quy Hoạch Động**, **Xử lý Chuỗi**, **Xử lý Mảng & Ma Trận**, **Thuật toán Tính Độ Bền** mà bạn yêu cầu. Bạn có muốn tôi tiếp tục gửi nốt không?

Dưới đây là phần tiếp theo với các thuật toán bạn yêu cầu, cùng với các thuật toán tương tự.

---

5. Quy Hoạch Động & Quay Lui

Bài toán Balo 0/1 (Knapsack 0/1)

pascal

```
function Knapsack(W: Integer; wt, val: array of Integer; n: Integer): Integer;
var
  i, w: Integer;
  K: array of array of Integer;
begin
  SetLength(K, n + 1, W + 1);

  for i := 0 to n do
    for w := 0 to W do
    begin
      if (i = 0) or (w = 0) then
        K[i][w] := 0
      else if wt[i - 1] <= w then
        K[i][w] := Max(val[i - 1] + K[i - 1][w - wt[i - 1]], K[i - 1][w])
      else
        K[i][w] := K[i - 1][w];
    end;

  Result := K[n][W];
end;
```

Dãy con chung dài nhất (LCS - Longest Common Subsequence)

pascal

```
function LCS(X, Y: string): Integer;
var
  m, n, i, j: Integer;
  L: array of array of Integer;
begin
  m := Length(X);
  n := Length(Y);
  SetLength(L, m + 1, n + 1);

  for i := 0 to m do
    for j := 0 to n do
    begin
      if (i = 0) or (j = 0) then
        L[i][j] := 0
      else if X[i] = Y[j] then
        L[i][j] := L[i - 1][j - 1] + 1
      else
        L[i][j] := Max(L[i - 1][j], L[i][j - 1]);
    end;

  Result := L[m][n];
end;
```

Chuỗi con đối xứng dài nhất

pascal

```
function LongestPalindromeSubstr(S: string): Integer;
var
  n, i, j, cl: Integer;
  L: array of array of Integer;
begin
  n := Length(S);
  SetLength(L, n, n);

  for i := 0 to n - 1 do L[i][i] := 1;

  for cl := 2 to n do
```

```
    for i := 0 to n - cl do
    begin
      j := i + cl - 1;
      if (S[i] = S[j]) and (cl = 2) then
        L[i][j] := 2
      else if S[i] = S[j] then
        L[i][j] := L[i + 1][j - 1] + 2
      else
        L[i][j] := Max(L[i][j - 1], L[i + 1][j]);
    end;

  Result := L[0][n - 1];
end;
```

Quay lui (Backtracking) - Bài toán N-Queen

pascal

```
const
  N = 8;
var
  board: array[1..N, 1..N] of Integer;

function IsSafe(row, col: Integer): Boolean;
var
  i, j: Integer;
begin
  for i := 1 to col - 1 do
    if board[row, i] = 1 then Exit(False);

  i := row;
  j := col;
  while (i > 0) and (j > 0) do
  begin
    if board[i, j] = 1 then Exit(False);
    Dec(i);
    Dec(j);
  end;

  i := row;
  j := col;
  while (i <= N) and (j > 0) do
```

```
  begin
    if board[i, j] = 1 then Exit(False);
    Inc(i);
    Dec(j);
  end;

  Result := True;
end;

procedure SolveNQueen(col: Integer);
var
  i: Integer;
begin
  if col > N then
  begin
    for i := 1 to N do
    begin
      for col := 1 to N do
        Write(board[i, col], ' ');
      Writeln;
    end;
    Writeln;
    Exit;
  end;

  for i := 1 to N do
    if IsSafe(i, col) then
    begin
      board[i, col] := 1;
      SolveNQueen(col + 1);
      board[i, col] := 0;
    end;
end;
```

---

6. Thuật toán Xử lý Chuỗi

Tìm kiếm chuỗi (KMP)

pascal

```
procedure ComputeLPSArray(pattern: string; var lps: array of Integer);
var
  len, i: Integer;
begin
  len := 0;
  lps[0] := 0;
  i := 1;
  while i < Length(pattern) do
  begin
    if pattern[i] = pattern[len] then
    begin
      Inc(len);
      lps[i] := len;
      Inc(i);
    end
    else if len <> 0 then
      len := lps[len - 1]
    else
    begin
      lps[i] := 0;
      Inc(i);
    end;
  end;
end;

procedure KMPSearch(text, pattern: string);
var
  lps: array of Integer;
  i, j: Integer;
begin
  SetLength(lps, Length(pattern));
  ComputeLPSArray(pattern, lps);

  i := 0;
  j := 0;
  while i < Length(text) do
  begin
    if pattern[j] = text[i] then
    begin
      Inc(i);
      Inc(j);
```

```
    end;

    if j = Length(pattern) then
    begin
      Writeln('Pattern found at index ', i - j);
      j := lps[j - 1];
    end
    else if (i < Length(text)) and (pattern[j] <> text[i]) then
    begin
      if j <> 0 then j := lps[j - 1]
      else Inc(i);
    end;
  end;
end;
```

---

7. Thuật toán Xử lý Mảng & Ma Trận

Tìm phần tử lớn nhất / nhỏ nhất trong mảng

pascal

```
function FindMax(A: array of Integer; N: Integer): Integer;
var
  maxVal, i: Integer;
begin
  maxVal := A[0];
  for i := 1 to N - 1 do
    if A[i] > maxVal then maxVal := A[i];
  Result := maxVal;
end;
```

---

8. Thuật toán Tính Độ Bền

Tìm ước bền vững nhất của 1 số

pascal

```pascal
function MostDurableDivisor(N: Integer): Integer;
var
  i, maxDiv, maxDurability, durability: Integer;
begin
  maxDiv := 1;
  maxDurability := 0;

  for i := 1 to N do
    if (N mod i = 0) then
    begin
      durability := 0;
      while (N mod i = 0) do
      begin
        Inc(durability);
        N := N div i;
      end;
      if durability > maxDurability then
      begin
        maxDurability := durability;
        maxDiv := i;
      end;
    end;

  Result := maxDiv;
end;
```

Tôi đã cung cấp đầy đủ các thuật toán trong danh sách và bổ sung một số thuật toán tương tự. Nếu bạn muốn thêm thuật toán nào khác, hãy cho tôi biết! 😊