Java 최적화

2025-06-22

## 문제

<https://www.acmicpc.net/problem/1325>

효율적인 해킹 (실버1)

그래프 크기를 계산하는 DFS 구현 문제.

시간초과 제한이 매우 엄격해서 성능 최적화 구현 기법이 필요하다.

구현#1 시간초과 실패

구현#2 부터 통과

구현#2 메모리: 279,944kb 시간: 8,448ms

구현#3 메모리: 138,620kb 시간: 4,968ms

구현#4 메모리: 40,844kb 시간: 5,088ms

## 구현 #1 - 시간 초과

|  |
| --- |
| import java.io.\*;  import java.util.\*;  public class Main {    static List<Integer>[] neighbors;    static boolean[] visited;    static int size;    static void DFS(int node) {      if (visited[node]) return;      visited[node] = true;      ++size;      if (neighbors[node] != null)        for (int neighbor : neighbors[node])          if (!visited[neighbor])            DFS(neighbor);    }    @SuppressWarnings("unchecked")    public static void main(String[] args) throws IOException {      var scanner = new Scanner(new BufferedInputStream(System.in));      int N = scanner.nextInt();      neighbors = new ArrayList[N+1];      int E = scanner.nextInt();      for (int i = 0; i < E; ++i) {        int a = scanner.nextInt();        int b = scanner.nextInt();        if (neighbors[b] == null) neighbors[b] = new ArrayList<>();        neighbors[b].add(a);      }      scanner.close();      int maxSize = 0;      int[] sizes = new int[N+1];      for (int node = 1; node <= N; ++node) {        visited = new boolean[N+1];        size = 0;        DFS(node);        sizes[node] = size;        if (size > maxSize) maxSize = size;      }      var builder = new StringBuilder();      for (int node = 1; node <= N; ++node)        if (sizes[node] == maxSize)          builder.append(node).append(' ');      System.out.println(builder.toString());    }  } |

## 구현 #2 - 통과

|  |
| --- |
| import java.io.\*;  import java.util.\*;  public class Main {    static IntArray[] neighbors;    static boolean[] visited;    static int size;    static class IntArray {      int[] a;      int count = 0;      public IntArray(int size) { a = new int[size]; }      public int get(int i) { return a[i]; }      public int size() { return count; }      public void add(int value) {        if (count == a.length)          a = Arrays.copyOf(a, a.length \* 2);        a[count++] = value;      }    }    static void DFS(int node) {      if (visited[node]) return;      visited[node] = true;      ++size;      if (neighbors[node] != null) {        int end = neighbors[node].size();        for (int i = 0; i < end; ++i) {          int neighbor = neighbors[node].get(i);          if (!visited[neighbor])            DFS(neighbor);        }      }    }    public static void main(String[] args) throws IOException {      var scanner = new Scanner(new BufferedInputStream(System.in));      int N = scanner.nextInt();      neighbors = new IntArray[N+1];      int E = scanner.nextInt();      for (int i = 0; i < E; ++i) {        int a = scanner.nextInt();        int b = scanner.nextInt();        if (neighbors[b] == null) neighbors[b] = new IntArray(8);        neighbors[b].add(a);      }      scanner.close();      int maxSize = 0;      int[] sizes = new int[N+1];      for (int node = 1; node <= N; ++node) {        visited = new boolean[N+1];        size = 0;        DFS(node);        sizes[node] = size;        if (size > maxSize) maxSize = size;      }      var builder = new StringBuilder();      for (int node = 1; node <= N; ++node)        if (sizes[node] == maxSize)          builder.append(node).append(' ');      System.out.println(builder.toString());    }  } |

A screenshot of a computer

AI-generated content may be incorrect.

## 구현 #3 - 큰 성능 개선

|  |
| --- |
| import java.io.\*;  import java.util.\*;  public class Main {    static IntArray[] neighbors;    static boolean[] visited;    static int size;    static class IntArray {      int[] a;      int count = 0;      public IntArray(int size) { a = new int[size]; }      public int get(int i) { return a[i]; }      public int size() { return count; }      public void add(int value) {        if (count == a.length)          a = Arrays.copyOf(a, a.length \* 2);        a[count++] = value;      }    }    static void DFS(int node) {      if (visited[node]) return;      visited[node] = true;      ++size;      if (neighbors[node] != null) {        int end = neighbors[node].size();        for (int i = 0; i < end; ++i) {          int neighbor = neighbors[node].get(i);          if (!visited[neighbor])            DFS(neighbor);        }      }    }    public static void main(String[] args) throws IOException {      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));      StringTokenizer tokenizer = new StringTokenizer(reader.readLine());      int N = Integer.parseInt(tokenizer.nextToken());      int E = Integer.parseInt(tokenizer.nextToken());      neighbors = new IntArray[N+1];      for (int i = 0; i < E; ++i) {          tokenizer = new StringTokenizer(reader.readLine());          int a = Integer.parseInt(tokenizer.nextToken());          int b = Integer.parseInt(tokenizer.nextToken());          if (neighbors[b] == null) neighbors[b] = new IntArray(8);          neighbors[b].add(a);      }      reader.close();      int maxSize = 0;      int[] sizes = new int[N+1];      for (int node = 1; node <= N; ++node) {        visited = new boolean[N+1];        size = 0;        DFS(node);        sizes[node] = size;        if (size > maxSize) maxSize = size;      }      var builder = new StringBuilder();      for (int node = 1; node <= N; ++node)        if (sizes[node] == maxSize)          builder.append(node).append(' ');      System.out.println(builder.toString());    }  } |

A screenshot of a computer

AI-generated content may be incorrect.

메모리, 시간 모두 크게 감소함

## 구현 #4 - 메모리 절약

|  |
| --- |
| import java.io.\*;  import java.util.\*;  public class Main {    static IntArray[] neighbors;    static boolean[] visited;    static int size;    static class IntArray {      int[] a;      int count = 0;      public IntArray(int size) { a = new int[size]; }      public int get(int i) { return a[i]; }      public int size() { return count; }      public void add(int value) {        if (count == a.length)          a = Arrays.copyOf(a, a.length \* 2);        a[count++] = value;      }    }    static void DFS(int node) {      if (visited[node]) return;      visited[node] = true;      ++size;      if (neighbors[node] != null) {        int end = neighbors[node].size();        for (int i = 0; i < end; ++i) {          int neighbor = neighbors[node].get(i);          if (!visited[neighbor])            DFS(neighbor);        }      }    }    public static void main(String[] args) throws IOException {      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));      StringTokenizer tokenizer = new StringTokenizer(reader.readLine());      int N = Integer.parseInt(tokenizer.nextToken());      int E = Integer.parseInt(tokenizer.nextToken());      neighbors = new IntArray[N+1];      for (int i = 0; i < E; ++i) {          tokenizer = new StringTokenizer(reader.readLine());          int a = Integer.parseInt(tokenizer.nextToken());          int b = Integer.parseInt(tokenizer.nextToken());          if (neighbors[b] == null) neighbors[b] = new IntArray(8);          neighbors[b].add(a);      }      reader.close();      int maxSize = 0;      int[] sizes = new int[N+1];      visited = new boolean[N+1];      for (int node = 1; node <= N; ++node) {        Arrays.fill(visited, false);        size = 0;        DFS(node);        sizes[node] = size;        if (size > maxSize) maxSize = size;      }      var builder = new StringBuilder();      for (int node = 1; node <= N; ++node)        if (sizes[node] == maxSize)          builder.append(node).append(' ');      System.out.println(builder.toString());    }  } |



시간은 아주 쬐금 늘었지만, 메모리가 크게 줄었음.