**Análisis de Algoritmos**

**Grupo 50**

**Profesora**

Ana Lorena Valerio Solís

**Instituto Tecnológico de Costa Rica**

**Proyecto programado # 1**

**Análisis de complejidad algorítmica para el problema de ubicación de restaurantes de una cadena comercial**

**Integrantes**

Nelson Andrey Vega Soto

Yuliana Rojas

**Introducción:**

**Análisis del Problema:**

[ Debe analizar todos los puntos a tratar en el proyecto, considerando estos como requerimientos de un proyecto de software o como las preguntas o situaciones a resolver en un proceso de indagatoria (investigación). Incluir una descripción eficiente de la situación y las metodologías o medios necesarios para brindar una posible solución o abordaje al problema.]

**Solución del Problema:**

* la última solución, indique cuales son las estructuras o clases utilizadas.
* lógica de cómo se realizó cada algoritmo, indique todas las mejoras realizadas a los algoritmos para mejorar su eficiencia.

**Análisis de Resultados:**

* Resultados finales, indique que partes están completas, cuales defectuosos, y cuáles no se realizaron y el porqué, que aspectos se pueden mejorar.
* Adjunte las tablas de todas las mediciones realizadas a sus algoritmos (empírica, analítica y la medición gráfica).
* Adjunte el cálculo del factor de crecimiento.
* Analice los datos obtenidos, a que se debe que un algoritmo sea mejor que otro. Indique las características o ventajas de cada algoritmo sobre el otro.

**Algoritmo A**

**Medición Empírica**

| Operaciones | Cantidad de datos de entrada por defecto | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 10 | 15 | 20 | 25 | - |
| Asignaciones | 47 | 109 | 184 | 340 | 505 | - |
| Comparaciones | 53 | 156 | 271 | 554 | 858 | - |
| Cantidad de líneas ejecutadas | 100 | 265 | 455 | 894 | 1363 | - |
| Cantidad de líneas del código | 12 | | | | | |

**Factor de Crecimiento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Talla | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas |
| De 6 a 10 | 1.6 | 2.25 | 2.94 | 2.56 |
| De 10 a 15 | 1.5 | **1.68** | **1.73** | **1.77** |
| De 15 a 20 | 1.3 | 1.84 | 2.04 | 1.93 |
| De 20 a 25 | 1.2 | 1.48 | 1.54 | 1.52 |
| De 15 a 25 | 1.6 | 2.74 | 3.16 | 2.99 |
| De 10 a 20 | 2 | 3.11 | 3.55 | 3.37 |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** | Cuadrática |  |
| Clasificación del comportamiento de las **comparaciones** | Cuadrática |

|  |  |  |  |
| --- | --- | --- | --- |
| Clasificación según su entrada de los datos use la notación  Theta, O Grande, y Omega según corresponda | | | |
| **Entrada de los datos** |  |  |  |
| Clasificación |  |  |  |

**Factor de Crecimiento**

|  |  |
| --- | --- |
| Código fuente  Solo se analiza el código del método de resolver el rompecabezas. | Medición de líneas ejecutadas en el peor de los casos  (línea por línea) |
| public void greedy(Grafo grafoM){  ArrayList<Vertice> grafo = grafoM.grafo;  int beneficioTotal = 0;  for(int i=0 ; i< grafo.size() ; i++){  if(!grafo.get(i).visitado){  grafo.get(i).visitado = true;  beneficioTotal += grafo.get(i).getBeneficio();  for(int j=0 ; j<grafo.get(i).listaArcos.size() ; j++){  grafoM.buscarVertice(grafo.get(i).listaArcos.get(j).destino).visitado = true;  }  }  }  }  public Vertice buscarVertice(String lugar){  for (int i=0; i<grafo.size() ; i++){  if(grafo.get(i).nombre == lugar)  return grafo.get(i);  }  return null;  } | 1  1  1 + 1+n + n  n  n  n  n+n+n2+n2  n (4n+3) = 4n2+3n  2n+2  n  n  1 |
| Total (la suma de todos los pasos) | 6n2 + 10n + 4 |
| Clasificación en notación O Grande | O(6n2) cuadrática |

**Nombre del algoritmo #2: máximo Beneficio B**

**Medición Empírica**

| Operaciones | Cantidad de datos de entrada por defecto | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 10 | 15 | 20 | 25 | - |
| Asignaciones | 360 | 835 | 1117 | 545 | 1042 | - |
| Comparaciones | 618 | 1502 | 2687 | 4808 | 7303 | - |
| Cantidad de líneas ejecutadas | 909 | 2222 | 3972 | 7134 | 10850 | - |
| Cantidad de líneas del código |  | | | | | |

**Factor de Crecimiento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Talla | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas |
| De 6 a 10 | 1.67 | **2.32** | **2.43** | **2.44** |
| De 10 a 15 | 1.50 | **1.34** | **1.79** | **1.79** |
| De 15 a 20 | 1.33 | 0.49 | 1.79 | 1.80 |
| De 20 a 25 | 1.25 | 1.91 | 1.52 | 1.52 |
| De 15 a 25 | 1.67 | 0.93 | 2.72 | 2.73 |
| De 10 a 20 | 2 | 0.65 | 3.20 | 3.21 |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** |  |  |
| Clasificación del comportamiento de las **comparaciones** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Clasificación según su entrada de los datos use la notación  Theta, O Grande, y Omega según corresponda | | | |
| **Entrada de los datos** |  |  |  |
| Clasificación |  |  |  |

**Factor de Crecimiento**

|  |  |
| --- | --- |
| Código fuente  Solo se analiza el código del método de resolver el rompecabezas. | líneas ejecutadas |
| public void maximoBeneficio(Grafo grafo, Vertice vertice){  if((vertice.visitado)&&(esMarcado(grafo,vertice)==false)){  if(!vertice.marcar){  benTotal += vertice.beneficio;  vertice.marcar = true;  marcar(grafo, vertice);  }  }  if(vertice.visitado){  return;  }  vertice.visitado = true;  if(maxBeneficioAux(grafo,vertice)){  benTotal += vertice.beneficio;  vertice.marcar = true;  marcar(grafo, vertice);  }  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  maximoBeneficio(grafo,grafo.buscarVertice(vertice.listaArcos.get(i).destino));  }  }  public boolean maxBeneficioAux(Grafo grafo, Vertice vertice){  int cont = 0;  boolean siCon = false;  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  if(!grafo.buscarVertice(vertice.listaArcos.get(i).destino).visitado){  grafo.buscarVertice(vertice.listaArcos.get(i).destino).beneficio;  }  if(grafo.buscarVertice(vertice.listaArcos.get(i).destino).marcar){  return false;  }  if(grafo.buscarVertice(vertice.listaArcos.get(i).destino).notM){  siCon = true;  }  }  if(vertice.beneficio >= cont){  return true;  }else{  if(vertice.listaArcos.size() == 1){  if(siCon)  return true;  }  return false;  }  }  public void marcar(Grafo grafo, Vertice vertice){  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  grafo.buscarVertice(vertice.listaArcos.get(i).destino).notM = true;  }  }  public boolean esMarcado(Grafo grafo, Vertice vertice){  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  if(grafo.buscarVertice(vertice.listaArcos.get(i).destino).marcar){  return true;  }  }  return false;  } | n\*(1+7n+4n2+2)  n  n  n  n\*(4n2+2n+5)  n  n  n  n\*(16n2+16n+10)  n  n  n\*(4n2+2n+5)  n\*(2n+2)  n2\*(4n+3)  ``````````````````  1  1  2n+2  n\*(4n+3)  n\*(4n+3)  n\*(4n+3)  n  n\*(4n+3)  n  1  1  1  1  1  1  16n2+16n+10  `````````````````  2n+2  n\*(4n+3)  4n2+2n+5  ```````````````````  2n+2  n\*(4n+3)  n  n  7n+4n2+2 |
| Total (la suma de todos los pasos) | 28n3+34n2+33n |
| Clasificación en notación O Grande | O(28n3) |

**Nombre del algoritmo #3:**

**Medición Empírica**

| Operaciones | Cantidad de datos de entrada por defecto | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 10 | 15 | 20 | 25 | - |
| Asignaciones | 47 | 109 | 185 | 340 | 505 | - |
| Comparaciones | 53 | 156 | 282 | 554 | 858 | - |
| Cantidad de líneas ejecutadas | 100 | 265 | 467 | 894 | 1363 | - |
| Cantidad de líneas del código |  | | | | | |

**Factor de Crecimiento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Talla | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas |
| De 6 a 10 | 1.67 | 2.32 | 2.94 | 2.65 |
| De 10 a 15 | 1.50 | **1.70** | **1.81** | **1.76** |
| De 15 a 20 | 1.33 | 1.84 | 1.96 | 1.91 |
| De 20 a 25 | 1.25 | 1.49 | 1.55 | 1.52 |
| De 15 a 25 | 1.67 | 2.73 | 3.04 | 2.92 |
| De 10 a 20 | 2 | 3.12 | 3.55 | 3.37 |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** |  |  |
| Clasificación del comportamiento de las **comparaciones** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Clasificación según su entrada de los datos use la notación  Theta, O Grande, y Omega según corresponda | | | |
| **Entrada de los datos** |  |  |  |
| Clasificación |  |  |  |

**Factor de Crecimiento**

|  |  |
| --- | --- |
| Código fuente | Líneas ejecutadas |
| public void cantMaxRest(Grafo grafoM) {  int cant = 0;  ArrayList<Vertice> grafo = grafoM.grafo;  for(int i=0 ; i<grafo.size() ; i++){  if(!grafo.get(i).visitado){  grafo.get(i).visitado = true;  cant++;  for(int j=0 ; j < grafo.get(i).listaArcos.size() ; j++){  grafoM.buscarVertice(grafo.get(i).listaArcos.get(j).destino).visitado = true;  }}}}  public void ordenarGrafo(){  ArrayList<Vertice> grafoOrdenado = new ArrayList<>();  for(int j = 0; j<grafo.size();j++){  if(grafoOrdenado.isEmpty()){  grafoOrdenado.add(grafo.get(j));  }  else{  for(int i = 0; i<grafoOrdenado.size(); i++){  if(grafo.get(j).listaArcos.size()<= grafoOrdenado.get(i).listaArcos.size()){  grafoOrdenado.add(i, grafo.get(j));  }  else{  grafoOrdenado.add(grafo.get(j));  }  break;  }  }  }  this.grafo = grafoOrdenado;  } | 1  1  2n+2  n  n.  n  n (2n+2)  n2 \* (4n+3)  1  2n+2  n  n  2n+2  n  n  n  n  1 |
| Total (la suma de todos los pasos) | 4n3+5n2+17n+10 |
| Clasificación en notación O Grande | O(4n3) |

**Nombre del algoritmo #4:**

**Medición Empírica**

| Operaciones | Cantidad de datos de entrada por defecto | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | 10 | 15 | 20 | 25 |  |
| Asignaciones |  | 3401 |  |  | 27458 |  |
| Comparaciones |  | 6232 |  |  | 51972 |  |
| Cantidad de líneas ejecutadas |  | 9476 |  |  | 78607 |  |
| Tiempo de ejecución |  |  |  |  |  |  |
| Cantidad de líneas del código |  | | | | | |

**Factor de Crecimiento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Talla | Factor talla | Factor Asig | Factor Comp | Factor Cantidad de líneas ejecutadas |
| De 6 a 10 |  |  |  |  |
| De 10 a 15 |  |  |  |  |
| De 15 a 20 |  |  |  |  |
| De 20 a 25 |  |  |  |  |
| De 15 a 25 |  |  |  |  |
| De 10 a 20 |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Clasificación del comportamiento de las **asignaciones** |  |  |
| Clasificación del comportamiento de las **comparaciones** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Clasificación según su entrada de los datos use la notación  Theta, O Grande, y Omega según corresponda | | | |
| **Entrada de los datos** |  |  |  |
| Clasificación |  |  |  |

|  |  |
| --- | --- |
| Código fuente. | Líneas ejecutadas. |
| public static void maxBeneficioD(Grafo grafo){  for(int i=0; i<grafo.grafo.size() ; i++){  recorre(grafo, grafo.grafo.get(i));  if(maxBeneficioTotal < maxBeneficioParcial){  maxBeneficioTotal = maxBeneficioParcial;  }  maxBeneficioParcial = 0;  grafo.desmarcar();  }  maxBeneficioTotal = 0;  maxBeneficioParcial = 0;  }  public static void recorre(Grafo grafo, Vertice vertice){  if(vertice.visitado == true){  return;  }  if(verifica(grafo,vertice)){  maxBeneficioParcial += vertice.beneficio;  vertice.marcar = true;  recorridoP+=(vertice.nombre+" ");  }  vertice.visitado = true;  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  recorre(grafo,grafo.buscarVertice(vertice.listaArcos.get(i).destino));  }  }  public static boolean verifica(Grafo grafo, Vertice vertice){  for(int i=0 ; i<vertice.listaArcos.size() ; i++){  if(grafo.buscarVertice(vertice.listaArcos.get(i).destino).marcar){  return false;  }  }  return true;  } | 2n+2  n \* (4n3 + 13n2 + 13n)  n  n  n  n  n  n  ````````````````````````````````````  n  n  n\*(4n2+7n+2) = 4n3+7n2+2n  n  n  n  n  n \* (2n+2) = 2n2 + 2n  n \* (4n+3) = 4n2 + 3n  4n3 + 13n2 + 13n  ````````````````````````````````````  2n+2  n \* (4n+3)  n  n  4n2 + 7n + 2 |
| Total (la suma de todos los pasos) | 4n4 +13n3 +13n2 +8n+2 |
| Clasificación en notación O Grande | O(4n4) |

**Factor de Crecimiento**

**Conclusiones:**

* Según la medición realizada indique cuál de los algoritmos es más eficiente analítica y empíricamente; tanto en comparaciones y asignaciones y según la entrada de los datos.

**Recomendaciones:**

* Aspectos o ámbitos de interés para ampliar el alcance del proyecto.
* Áreas o campos de estudio que pueden complementar las temáticas analizadas.
* Ideas que sean de utilidad para interesados en el área en que se desenvuelve el proyecto.

**Literatura citada**

* Mínimo de debe incluir 4 referencias, use el formato APA, incluya las consultas realizadas en la web.