UNIVERSIDAD DE BUENOS AIRES
Facultad de Ingeniería
Departamento de Computación

Algoritmos y complejidad computacional

Este es un modesto aporte para los alumnos de la fácultad de ingeniería de la UBA de las carreras de licenciatura en análsis de sistemas e ingeniería informática. De ninguna manéra pretende ser una guía de estudio, ni remplaza las clases presenciales, el material oficial de la catedra esta disponible en el web site de la máteria.
http://materias.fi.uba.ar/7510/

Autor: Isaac Edgar Camacho Ocampo
Carrera: Licenciatura en Análisis de sistemas

Buenos Aires, 2019

Índice general

Resumen 5
0.1. ¿Qué es un algoritmo? 5
0.2. ¿Qué es un buen algoritmo? 5
0.3. ¿Cuando un problema está bien resuelto? 5
0.4. Repaso - Complejidad computacional 5
Modelo uniforme 6
Modelo logarítmico 6

1. Algoritmos dividir y conquistar **7**
1.1. Conocimientos previos 7
1.2. Estado del arte 7

2. Grafos algoritmos sobre grafos **9**
2.1. Teoría clásica 9
2.1.1. Definición de variables 9
2.1.2. Pruebas y refutaciones 9
2.2. Hipótesis 9

3. Problema de árbol generador mínimo **11**
3.1. Simulación de resultados 11
3.1.1. Suposiciones 11
3.1.2. Modelos 11
3.2. Resultados preliminares 11
3.3. Resultados postprocesados 11
3.3.1. Valores atípicos 11
3.3.2. Correlaciones 11

4. Problema de flujo máximo **13**

5. Problema de camino mínimo **15**

6. Algoritmos de fuerza bruta y backtracking **17**

7. Problemas NP completos **19**

8. Heurísticas y algoritmos golosos **21**

9. Algoritmos genéticos **23**

10. Algoritmos evolutivos **25**

Abstract

0.1. ¿Qué es un algoritmo?

Podemos decir que es la especificación de una secuencia de instrucciones para a partir de datos de entrada resolver un problema con un mecanismo automáticos es decir resolver un problema tomando tus entradas y entregando datos de salida o resultados la especificación puede ser de diversas maneras en lenguaje natural en pseudocódigo etcétera Pero esencialmente siempre se llega a que la especificación es que hay que hacer a partir de los datos de entrada para obtener los datos de salida

0.2. ¿Qué es un buen algoritmo?

la primera idea que tenemos es que un algoritmo funciona cuando resuelve el problema en cuestión, bueno ahora supongamos que funciona el algoritmo qué criterios debemos tomar para decir que un algoritmo es bueno, podemos ir con algoritmos bueno cuando es claro es decir fácil entendible, o cuando es fácil de implementar o qué es un buen algoritmo cuando es eficiente o cuando es rápido todos estos criterios son válidos el más popular para determinar la calidad de un algoritmo es el tiempo es decir un algoritmo es bueno cuando funciona rápidamente

Dados algoritmos que resuelven un mismo problema ? ¿Cuál es mejor? Según el criterio del tiempo podemos decir que es mejor el que funciona más rápido, evidentemente en la actualidad es preferible un algoritmo que resolver rápidamente un problema a otro que lo hace en más tiempo

0.3. ¿Cuando un problema está bien resuelto?

Bueno esencialmente un problema está bien resultó cuando el algoritmo es bueno es decir cuando el algoritmo es aceptable según los criterios que hayamos seleccionado generalmente en nuestros problemas de grafos el tiempo es el factor decisivo.

0.4. Repaso - Complejidad computacional

El objetivo es querer medir el tiempo de ejecución de los algoritmos queremos saber cuánto tarda en función de los datos de entrada. Mesa virtual utilizar una forma de medir el tiempo de ejecución de un algoritmo utilizando el peor caso en función del tamaño de la entrada del algoritmo

Para una entrada de tamaño n que es lo peor que le puede pasar es decir para una entrada genérica el peor caso posible

Y cómo medimos el tamaño de la entrada del algoritmo Para esto y dos posibilidades

Modelo uniforme

Asumimos que cada dato individual es decir un entero o un booleano ocupa una posición individual de memoria por ejemplo decimos que un arreglo de n elementos tiene tamaño n

En el siguiente ejemplo podemos ver un algoritmo para determinar si un número ingresado por parámetro es primo un error muy común consiste en creer que el tamaño de la entrada es el parámetro de entrada sin embargo podemos ver que para este algoritmo el parámetro de entrada es el número a calcular y el tamaño de la entrada también es uno

Uno estaría tentado a decir que función de complejidad de algoritmos sería o dn pero podemos ver que el tamaño de la entrada es 1

Pero podemos ver que si el número es lo suficientemente grande este algoritmo tardará mucho y su función de complejidad no será o dn

Modelo logarítmico

Se mide el tamaño en bits de cada dato individual por ejemplo el tamaño de un entero k se define como n igual a logaritmo en base 2 de ca más uno y tomar la parte entera

Podemos darnos cuentas que este algoritmo es malo porque para números grandes lo peor que le puede pasar es que recorra todos los números intermedios y después de un cálculo minucioso podemos determinar que el orden de complejidad de este algoritmo es de 2 a la n El orden de complejidad es exponencial respecto de la cantidad de bits que ocupa el número

Primo.java

```
1  package test;
2
3  public class Primo
4  {
5      public boolean esPrimo(int k)
6      {
7          for(int i=2; i<k-1; ++i)
8          {
9              if( k % i == 0 )
10                 return false;
11         }
12
13         return true;
14     }
15 }
16
```

Capítulo 1

Algoritmos dividir y conquistar

1.1. Conocimientos previos

1.2. Estado del arte

Capítulo 2

Grafos algoritmos sobre grafos

2.1. Teoría clásica

2.1.1. Definición de variables

2.1.2. Pruebas y refutaciones

2.2. Hipótesis

Capítulo 3

Problema de árbol generador mínimo

3.1. Simulación de resultados

3.1.1. Suposiciones

3.1.2. Modelos

3.2. Resultados preliminares

3.3. Resultados postprocesados

3.3.1. Valores atípicos

3.3.2. Correlaciones

Capítulo 4

Problema de flujo máximo

Capítulo 5

Problema de camino mínimo

Capítulo 6

Algoritmos de fuerza bruta y backtracking

Capítulo 7

Problemas NP completos

Capítulo 8

Heurísticas y algoritmos golosos

Capítulo 9

Algoritmos genéticos

Capítulo 10

Algoritmos evolutivos