Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

# Algoritmos y Complejidad

## Trabajo Práctico 3
## **Algoritmos Greedy**

Primer cuatrimestre de 2019

1. Generalidades de los algoritmos greedy

   *a*) Identifique los elementos y explique las características generales de los algoritmos *greedy*.

   *b*) Defina *conjunto de candidatos*, *conjunto de rechazados*, *función de viabilidad*, *función de selección* y *función objetivo*.

   *c*) Describa el esquema general de un algoritmo greedy.

   *d*) ¿Puede un algoritmo greedy no encontrar una solución óptima? Explique por qué ocurre esto.

2. *Problema del cambio con monedas*

   Dada una cantidad ilimitada de monedas con denominaciones $1 = d_1 < d_2 < \ldots < d_n$ y un monto de centavos a pagar $P$.

   *a*) Diseñar una estrategia greedy para hallar una forma de pagar el monto requerido.

   *b*) Mostrar un ejemplo en que la estrategia dada no es optimal, es decir, se pueden utilizar menos monedas utilizando una estrategia diferente.

   *c*) Demostrar que si para todo $i$, $d_i = 2^{i-1}$, entonces la estrategia greedy es optimal.

3. El problema de la mochila. Para la versión del problema analizada en [BB96]:

   *a*) Determinar cuál de las siguientes estrategias *greedy* permite hallar la solución óptima: elegir el objeto de mayor valor, elegir el objeto de menor peso o elegir el objeto de mayor beneficio por unidad de peso. Justificar formalmente.

   *b*) Mostrar que si no se permite fraccionar los objetos, entonces ninguna de estas heurísticas encuentra la solución optimal.

4. *Selección de actividades* [CLRS09, Capítulo 16.1]

   Dado un conjunto de intervalos $C = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ sobre la recta de los números reales, se desea encontrar un subconjunto de $C$, de máxima cardinalidad [1], formado por intervalos disjuntos.

   Por ejemplo, sea $C = \{(1, 10), (2, 5), (4, 6), (5, 8), (7, 9), (8, 10)\}$, un subconjunto de $C$ formado por intervalos disjuntos con máxima cardinalidad es $S_1 = \{(2, 5), (5, 8), (8, 10)\}$, con cardinalidad $= 3$.

   *a*) Diseñar una estrategia de selección *greedy* que resuelva este problema. Analice si la estrategia dada es optimal. Justifique adecuadamente.

   *b*) Dar un algoritmo en base a la estrategia anterior determinando claramente las estructuras de datos utilizadas. Analizar el tiempo de ejecución.

5. Problemas de scheduling.

[1] cardinalidad: número o cantidad de elementos del conjunto

*a*) Supongamos que se desea almacenar $n$ programas en una cinta magnética de longitud $L$, siendo $l_p$ la longitud de cada programa $p$. Para leer un programa hay que posicionarse en el inicio de la cinta. Por lo tanto, si los programas son almacenados en el orden $p_1, p_2, \ldots p_n$, el tiempo necesario para leer el programa $p_j$ es proporcional a $\sum_{k=1}^{j} l_{p_k}$.

   I Enunciar una estrategia *greedy* optimal para almacenar los programas en la cinta de forma tal que se minimice el tiempo total de lectura de todos estos programas:

$$\sum_{j=1}^{n} \sum_{k=1}^{j} l_{p_k}$$

   II Mostrar un algoritmo que utilice esa estrategia, determinar las estructuras de datos adecuadas y analizar el tiempo de ejecución de ese algoritmo.

   III Demostrar formalmente que el uso de esa estrategia minimiza la expresión anterior.

*b*) En el problema de la asignación de trabajos se dispone de un conjunto de $n$ personas y un conjunto de $n$ trabajos. Supongamos que se dispone de una matriz $C$ de costos, donde $\mathtt{C}[i, j]$ representa el costo de encargar a la persona $i$ el trabajo $j$. El objetivo es encontrar una asignación biyectiva de trabajos a personas, que minimize el costo total, es decir la suma de los costos de asignación individuales.

   I Escribir un algoritmo basado en una heurística *greedy* que permita obtener una solución razonable (no necesariamente la óptima). Analizar el tiempo de ejecución.

   II Mostrar un ejemplo en el que la heurística adoptada no encuentre una solución optimal.

6. *Algoritmo de Huffman*[CLRS09, Capítulo 16.3]

   *a*) Demostrar que todo árbol binario que representa un código optimal debe ser completo, es decir, que todo nodo interno debe tener exactamente dos hijos.

   *b*) Probar utilizando *inducción generalizada* que todo árbol binario completo de $C$ hojas tiene $C - 1$ nodos internos.

   *c*) Dado un alfabeto compuesto por los caracteres *A,B,C,D,E,F,G,H* con la siguiente distribución de frecuencias

| Letra | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| Frecuencia | 2 % | 32 % | 5 % | 21 % | 15 % | 12 % | 5 % | 8 % |

   *a*) Determinar una codificación de longitud variable optimal utilizando el *Algoritmo de Huffman.*

   *b*) Calcular cuántos bits por caracter son utilizados con:

   - Una codificación de longitud fija.
   - La codificación de longitud variable obtenida en el inciso anterior.

   *c*) Calcular la tasa de compresión que logra la codificación de Huffman.

7. Supongamos que tenemos dos arreglos $A$ y $B$, cada uno conteniendo $n$ enteros positivos. Tenemos la posibilidad de reordenar los elementos en cada conjunto de la manera que queramos. Luego de reordenar nos pagarán un monto $P = \prod_{i=1}^{n} a_i^{b_i}$, donde $a_i$ es el i-ésimo elemento de $A$ y $b_i$ es el i-ésimo elemento de $B$.

   *a*) Dar un algoritmo que reordene los elementos de $A$ y $B$ de manera de maximizar el monto $P$.

   *b*) Probar que el algoritmo encuentra una solución optimal.

   *c*) Analizar el tiempo de ejecución del algoritmo.

# Referencias


[BB96] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics.* Prentice Hall, 1996.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction To Algorithms.* The MIT Press, 3rd edition, 2009.