Universidad Nacional de San Agustín de Arequipa
Escuela Profesional de Ciencia de la Computación
Curso: Análisis y Diseño de Algoritmos

LABORATORIO 3

Análisis Asintótico

Docente: Rolando Jesús Cárdenas Talavera

1 Competencia del Curso

El alumno comprenderá e identificará el uso adecuado de diferentes algoritmos para dar solución a problemas de manera eficiente teniendo en consideración el tiempo de procesamiento y la cantidad de recursos empleados.

2 Competencia del Laboratorio

El alumno deberá de analizar y comprender las diferentes técnicas de diseño de Algoritmos

3 Equipos y Materiales

- Un computador.
- Compilador del lenguaje C++

4 Actividad

4.1 Análisis Asintótico

Describir en que situaciones se puede presentar los siguiente complejidades de los algoritmos, incluya un ejemplo de un problema:

Complejidad	Situación-Ejemplo
$O(1)$	Operación Aritmética
$O(\log(n))$	Búsqueda Binaria
$O(\sqrt{n})$	for(i=1,p=0;p<=n;i++) p+=i;
$O(n)$	Búsqueda Lineal
$O(n\log(n))$	Merge Sort
$O(n^2)$	Insertion Sort
$O(n^3)$	Multiplicación de Matrices
$O(2^n)$	Fibonacci Recursivo
$O(n!)$	Problema del Viajero - Fuerza Bruta

4.2 Análisis de Algoritmos

Detalle el tiempo de ejecución linea a linea y total de los siguientes algoritmos: (Utilice Notación-O). (Capítulo 3[1]). En el caso de encontrar llamadas a funciones del cual no se encuentra la referencia de código, asumir un tiempo $O(1)$.

Algorithm 1: . O(N)

```
1  double matching(int bitmask) {
2
3    if (memo[bitmask] > - 0.5)          O(1)
4      return memo[bitmask];             O(1)
5    if (bitmask == target)              O(1)
6      return memo[bitmask] = 0;         O(1)
7
8    double ans = 2000000000.0;          O(1)
9    int p1, p2;                         O(1)
10   for (p1 = 0 ; p1 < 2*N ; p1++)      O(N)
11     if ( !(bitmask & (1 << p1)) )     O(N)
12       break;                          O(1)
13   for (p2 = p1 + 1 ; p2 < 2*N ; p2++) O(N)
14     if ( !(bitmask & (1 << p2)) )     O(N)
15       ans = min(ans,                  O(N)
16           dist[p1][p2] + matching(bitmask | (1 << p1) | (1 << p2)));
17
18   return memo[bitmask] = ans;         O(1)
19 }
```

Algorithm 2: . O(N*N*N*N*N*N)

```
1  for (int i = 0 ; i < k ; i++)                          O(N)
2    scanf("%d" , &S[i]);                                 O(N)
3                                                         O(N)
4  for (int a = 0 ; a < k - 5 ; a++)                      O(N*N)
5    for(int b = a + 1 ; b < k - 4; b++)                  O(N*N*N)
6      for(int c = b + 1 ; c < k - 3; c++)                O(N*N*N*N)
7        for(int d = c + 1 ; d < k - 2; d++)              O(N*N*N*N*N)
8          for(int e = d + 1 ; e < k - 1; e++)            O(N*N*N*N*N*N)
9            for(int f = e + 1 ; f < k ; f++)
10             printf("%d %d %d %d %d %d \n", S[a], S[b], S[c], S[d], S[e], S[f])
```

Algorithm 3: . O(N)

```
1  int shop(int money, int g){
2    if ( money < 0 ) return -100000000;   O(1)
3    if ( g == C) return M - money;        O(1)
4    int &ans = memo[moeny][g];            O(1)
5    if ( ans != -1 ) return ans;          O(1)
6    for (int model = 1 ; model <= price[g][0] : model++)   O(N)
7      ans = max(ans, debt(money - price[g][model], g++));  O(N)
8    return ans;                           O(1)
9  }
```

Algorithm 4: . O(N)

```
1  int n=9, A[] = {4, -5, 4, -3, 4 , 4 , -4, 4, -5};   O(1)
2  int sum = 0, ans = 0;                               O(1)
3
4  for (int i = 0 ; i < n ; i++){                      O(N)
5    sum += A[i];                                      O(N)
6    ans = max( ans, sum );                            O(N)
7    if ( sum < 0 ) sum = 0;                           O(N)
8  }
9  printf("Max 1D Range sum = %d\n " , ans);           O(1)
```

Algorithm 5: . O(N*N*N*N*N*N)

```
1  maxSubRect = −127∗100∗100;                                          O(1)
2  for ( int i = 0 ; i < n ; i++ ) for ( int j = 0 ; j < n ; j++ )     O(N*N)
3    for ( int k = i ; k < n ; k++ ) for ( int l = 0 ; l < n ; l++ ) {  O(N*N*N*N)
4      subRect = 0;                                                    O(N*N*N*N)
5      for ( int a = i ; a <= k ; a++ ) for ( int b = j ; b <= l ; b++)  O(N*N*N*N*N*N)
6        subRect += A[a][b];                                           O(N*N*N*N*N*N)
7      maxSubRect = max(maxSubRect, subRect) ;}                        O(N*N*N*N)
```

Algorithm 6: . O(log(N)*log(N)*N)

```
1  for ( int i= n ; i > 0 ; i /= 2) {                  O(log(N))
2    for ( int j= 1 ; j < n ; j *= 2 ){                O(log(N)*log(N))
3      for ( int k = 0 ; k < n ; k += 2 ){             O(log(N)*log(N)*N)
4        ... // constant number of operations
5      }
6    }
7  }
```

5 Entregables

Al finalizar el estudiante deberá:

- Elaborar un documento, en donde se registre la resolución de cada uno de los ejercicios planteados.
- Deberán de subir a la plataforma Classroom el documento elaborado en **formato PDF** (se recomienda el uso de *LaTeX*) y el archivo comprimido con los códigos elaborados

6 Rúbrica de Evaluación

Esta actividad no tiene puntuación, se necesita verificar el nivel de conocimientos en cuanto a programación.

Rúbrica	**Cumple**	**Cumple con Observaciones**	**No cumple**
Informe: Desarrolla un informe, con un formato limpio y fácil de leer.	4	2	0
Ejercicios: Resuelve correctamente cada ejercicio	16	8	0
Errores ortográficos: Se descontará 0.5 puntos de encontrarse errores			

- **IMPORTANTE** En caso de copia o plagio o similares todos los alumnos implicados tendrán sanción en toda la evaluación del curso.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.