Algoritmos y Estructuras de Datos I
Segundo Cuatrimestre 2019

Guía Práctica 7
Tiempo de Ejecución de Peor Caso de un Programa

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Para los siguientes ejercicios suponer que tanto el return como asignaciones, operadores aritméticos, operadores lógicos, y las operaciones sobre vectores `.size()`, `.push_back()`, `.pop_back()` e indexación `v[i]` son operaciones elementales.

Ejercicio 1.
Contar la cantidad de operaciones elementales que realizan los siguientes programas.

```
int ultimo1(vector<int> s) {
  return s[s.size() - 1];
}

int ultimo2(vector<int> s) {
  int i = s.size();
  return s[i - 1];
}

int ultimo3(vector<int> s) {
  int i = 0;
  while (i < s.size()) {
    i++;
  }
  return s[i - 1];
}
```

Ejercicio 2.
Calcular el tiempo de ejecución de peor caso (en notación $\mathcal{O}$) de los siguientes programas con respecto al tamaño de los secuencias de entrada.

```
void f1(vector<int> s){
  int i = s.size() / 2;
  while( i >= 0){
    s[s.size() / 2 - i] = i;
    s[s.size() / 2 + i] = i;
    i--;
  }
}

// pre: |s| > 20000
void f2(vector<int> s){
  int i = 0;
  while(i < 10000){
    s[s.size() / 2 - i] = i;
    s[s.size() / 2 + i] = i;
    i++;
  }
}

int f3(vector<int> s) {
  int n = s.size();
  int acum = 0;
  for(int i = 0; i < n; i++) {
    if(i < 100) {
      for(int j = 0; j < n; j++) {
        acum = acum + i*j;
      }
    }
  }
  return acum;
}
```

```
void f4(vector<int> s){
  int rec = 0;
  int max_iter = 1000;
  if (max_iter > s.size()) {
    max_iter = s.size();
  }

  for(int i=0; i < max_iter; i++){
    for(int j=0; j < max_iter; j++){
      res += s[i] * s[j];
    }
  }
}

void f5(vector<int> s1, vector<int> s2){
  vector<int> res(s1.size()+s2.size(),0);
  // inicializa vector en O(|a|+|b|)
  for(int i=0; i < s1.size(); i ++){
    res[i]=s1[i]; // O(1)
  }
  for(int i=0; i < s2.size(); i ++){
    res[s1.size()+i]=s2[i]; // O(1)
  }
  return;
}
```

Ejercicio 3.

```
// pre: e pertenece a s1
int primer_indice(vector<int> s1, int e){
  int i = 0;
  while (s1[i] != e){
    i++;
  }
  return i ;
}
```

a) Identificar y dar al menos un ejemplo de peor y mejor caso para la función.

b) Calcular el tiempo de ejecución de peor caso de este programa en función del tamaño del vector.

Ejercicio 4.
Verdadero o falso (justificar).

a) A y B son dos programas que resuelven el mismo problema con un vector s como única entrada. Para un vector de tamaño 100 A demora 2 minutos en ejecutarse y bajo las mismas condiciones (misma entrada, computadora, recursos, etc) B demora 20 segundos. Por lo tanto, el programa B es más eficiente.

b) Si el tiempo de ejecución de peor caso de dos programas pertenece a $\mathcal{O}(n)$, entonces el tiempo de cómputo es equivalente (variando un poco según el estado de la computadora en el momento de ejecutar).

c) Se tienen dos programas cuyo tiempo de ejecución de peor caso perteneciente a $\mathcal{O}(n)$ y a $\mathcal{O}(log(n))$ respectivamente. Para cualquier entrada con tamaño mayor a 100, el segundo programa demorará menos tiempo en ejecutar.

d) Se tienen dos programas con tiempo de ejecución de peor caso perteneciente a $\mathcal{O}(n)$ y a $\mathcal{O}(log(n))$ respectivamente. Para cualquier entrada con tamaño mayor a un cierto número, el segundo programa demorará menos tiempo en ejecutar.

e) Si se hace un llamado a una función cuyo tiempo de ejecución de peor caso pertenece a $\mathcal{O}(n^2)$ dentro de un ciclo, el tiempo de ejecución del ciclo resultante pertenecerá a $\mathcal{O}(n^3)$.

Ejercicio 5.

```
int mesetaMasLarga(vector<int> s) {
  int i = 0;
  int maxMeseta = 0;
  int meseta;
  while (i < s.size()) {
    int j = i + 1;
    while (j < s.size() && s[i] == s[j]) {
      j++;
    }
    meseta = j - i;
    i = j;

    if (meseta > maxMeseta) {
      maxMeseta = meseta;
    }
  }
  return maxMeseta;
}
```

a) ¿Qué hace este programa?

b) Calcular el tiempo de ejecución de peor caso de este programa en función del tamaño del vector.

c) ¿Es posible escribir otro programa que resuelva el problema utilizando solo un ciclo?

Ejercicio 6.

```
vector<int> hacerAlgo(vector<int> &s) {
  vector<int> res;
  for (int i = 0; i < 100; i++) {
    res.push_back(contarApariciones(s, i+1));
  }

  return res;
}

int contarApariciones(vector<int> s, int elem) {
  int cantAp = 0;
  for (int i = 0; i < s.size(); i++) {
  if (s[i] == elem) {
      cantAp++;
    }
  }
  return cantAp;
}
```

a) Calcular el tiempo de ejecución de peor caso del programa `hacerAlgo` con respecto a $|s|$.

b) ¿Cambiaría la cantidad de operaciones que realiza el programa si `contarApariciones` tomara s por referencia?

c) ¿Cambiaría el tiempo de ejecución de peor caso en notación $\mathcal{O}$ si `contarApariciones` tomara s por referencia?

Ejercicio 7.
Dadas las siguientes funciones:

```
float f(vector<float> s) {
    float p;
    float res = 0;
    for (int i = 0; i < s.size(); i++) {
        p = g(h(s, i)));
        if (p > res) {
            res = p;
        }
    }
    return res;
}

float g(vector<float> v) {
    float p = 1;
    for (int i = 0; i < v.size(); i++) {
        p = p * v[i];
    }
    return p;
}
```

```
// Pre: n <= v.size()
vector<float> h(vector<float>& w, int n) {
    vector<float> res;
    int i = 0;
    while (i < n && w[i] > 0) {
        res.push_back(w[i]);
        i++;
    }
    return res;
}
```

1. Explicar brevemente con sus palabras qué cómputo realiza la función f. Hacerlo en a lo sumo un párrafo acompañado de un ejemplo (no explicar cómo lo hace).
2. Indicar el tiempo de ejecución de peor caso de h en función del n. Justificar.
3. Dar un ejemplo del peor caso y un ejemplo de mejor caso para la función h. Justificar.
4. Indicar el tiempo de ejecución de peor caso de f en función del tamaño de s. Justificar.

Ejercicio 8.

```
int sumarPotenciaHasta(int n) {
  int res = 0;
  int i = 1;
  while(i < n) {
    res = res + i;
    i = i * 2;
  }
  return res;
}
```

a) ¿Qué tiempo de ejecución de peor caso tiene el programa en función del valor n?

b) ¿Es posible calcular la suma de potencias hasta n con un tiempo de ejecución de peor caso asintóticamente mejor que el programa del punto anterior si la pre condición es True?

c) ¿Es posible calcular la suma de potencias hasta n con un tiempo de ejecución de peor caso asintóticamente mejor que el item a si la pre condición es $(\exists x : \mathbb{Z})\ n = 2^x + 1$?

Ejercicio 9.
Una matriz cuadrada se dice *triangular* si todos los elementos por debajo de la diagonal son iguales a cero.

a) Escribir un programa que calcule el determinante de una matriz triangular. Recordar que el determinante de una matriz triangular es el producto de los elementos de su diagonal.

b) Escribir un programa que determine si una matriz de $N \times N$ es o no triangular.

c) Calcular el tiempo de ejecución de peor caso de los programas:

   a) en función de la cantidad de elementos de la matriz.

   b) en función de la cantidad de *filas* de la matriz.

Ejercicio 10.
Dadas dos matrices A y B se desea multiplicarlas siguiendo esta especificación:

`proc multiplicar` (in m1: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in m2: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, out res: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
    `Pre` $\{completar\}$
    `Post` $\{|res| = |m1| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m1| \rightarrow_L (|res[i]| = |m2[0]| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |m2[0]| \rightarrow_L res[i][j] = \sum_{k=0}^{|m2|-1} m1[i][k] * m2[k][j]))))\}$
}

a) Completar la precondición del problema.

b) Escribir un programa que retorne AB

c) Determinar el tiempo de ejecución de peor caso de este programa en función de:

   a) La cantidad de filas y columnas de cada una de las matrices.

   b) Suponiendo que $N = \text{filas}_{m1} = \text{filas}_{m2} = \text{columnas}_{m1} == \text{columnas}_{m2}$

Ejercicio 11.
Sea mat una matriz de $N \times N$ y la siguiente función:

```
void sumasAcumuladas(vector<vector<int>> &mat, int N){
  for(int i = N - 1; i >= 0, i--) {
    for(int j = N - 1, j >= 0; j--) {
      mat[i][j] = sumHasta(mat, i, j, N);
    }
  }
}

int sumHasta(vector<vector<int> &mat, int I, int J, int N) {
  int res = 0;
```

```
  int lim;
  for (int i = 0; i <= I; i++) {
    if (i == I) {
      lim = J
    } else {
      lim = N - 1;
    }
    for (int j <= lim; j++) {
      res += mat[i][j]
    }
  }
  return res;
}
```

1. Calcular el tiempo de ejecución de peor caso del programa `sumasAcumuladas` en función de la cantidad de elementos de la matriz.
2. Dar un programa que resuelva sumasAcumuladas cuyo tiempo de ejecución en peor caso sea $\mathcal{O}(n^2)$

Ejercicio 12. Dada una secuencia de n números enteros, dar un programa que encuentre la máxima cantidad de elementos impares consecutivos cuya tiempo de ejecución de peor caso pertenezca a $\mathcal{O}(n)$.

Ejercicio 13. Escribir un programa que sea correcto respecto de la siguiente especificación, y cuyo tiempo de ejecución de peor caso pertenezca a $\mathcal{O}(|s|)$.

proc `restarAcumulado` (in s: $seq\langle\mathbb{Z}\rangle$, in x: $\mathbb{Z}$, out res: $seq\langle\mathbb{Z}\rangle$) {
  Pre $\{True\}$
  Post $\{|res| = |s| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow_L res[i] = x - \sum_{j=0}^{i} s[j])\}$
}

Ejercicio 14. Sea m una matriz de $N \times N$ donde cada posición contiene el costo de recorrer dicha posición (dicho costo es positivo para todas las posiciones). Asumiendo que la única forma de recorrer la matriz es avanzando hacia abajo y hacia la derecha, escribir un programa que calcule el mínimo costo para llegar desde la posición $(1, 1)$ hasta la posición (N, N) y cuyo tiempo de ejecución de peor caso pertenezca a $\mathcal{O}(N^2)$.

Ejercicio 15.
Sea A una matriz cuadrada y n un número natural.

a) Escribir un programa que calcule $\prod_{i=1}^{n} A$ (es decir A^n) (reutilizar el programa del ejercicio 10) ¿Cuál es el tiempo de ejecución de peor caso de esta función?

b) Resolver el punto anterior suponiendo que $n = 2^m$ (n potencia de 2) ¿Se pueden reutilizar cuentas ya realizadas? ¿Cuál es el tiempo de ejecución de peor caso para cada programa?

Ejercicio 16.
Dada una matriz de booleanos de n filas y m columnas con n impar. Se sabe que hay exactamente una fila que no está repetida, y el resto se encuentra exactamente dos veces en la matriz.

a) Escribir un programa que devuelva un vector con los valores de la fila que no se repite. ¿Cuál es su tiempo de ejecución de peor caso descripto ?

b) ¿Es posible un programa que recorra cada casillero de la matriz sólo una vez? En caso de no haberlo resuelto con esta restricción, modificar el programa para que la cumpla. ¿Cuál es su tiempo de ejecución de peor caso ?

c) La solución al punto anterior, ¿utiliza vectores auxiliares?, en caso que lo haga, escribir un programa que no los necesite.