Análisis de Algoritmos
Grado en Ingeniería Informática, UAM

Problemas

- **(E)**: ejercicio; **(E+)**: ejercicio avanzado.
- **(P)**: problema; **(P+)**: problema avanzado.
- **(R)**: recomendado; se resolverá en clase.

1 TAEs de algoritmos básicos

Ejercicios

1. **(E; R)** Denotando como `MMul` el algoritmo tradicional de multiplicación de matrices cuadradas, definir una OB adecuada para el mismo, y estimar para la misma $n_{\texttt{MMul}}(A, B)$, para cualquier par de matrices de dimensión N.

2. **(E; R)** El siguiente pseudocódigo corresponde a un algoritmo para encontrar el mínimo de una tabla

```
indice Min(tabla T, indice P, indice U)
  Min=T[P];
  iMin=P;
  para i de P+1 a U:
    si T[i] < Min:
       Min = T[i];
       iMin=i;
  devolver iMin;
```

   Analizar su tiempo de ejecución del algoritmo usando como operación básica la comparación de claves.

   Solución: Suponiendo $P = 1, U = N$ y a la vista del psc, se tiene

$$n_{min}(T, 1, N) = \sum_{2}^{N} 1 = N - 1$$

3. **(E)** Analizar el tiempo de ejecución del algoritmo trivial de potenciación de pseudocódigo

```
pot(int x, int N)
  p = x;
  para i de 2 a N:
    p = p*x;
  devolver p;
```

Solución: Escogemos $*$ como OB y de nuevo

$$n_{pot}(x, N) = \sum_{2}^{N} 1 = N - 1$$

4. **(E; R)** Para los siguientes fragmentos de código, establecer una operación básica adecuada y estimar sus tiempos de ejecución en función de n:

   - ```
     sum = 0;
     for (i=1; i<=n; i++)
       sum++;
     ```

     ```
     sum = 0;
     for (i=1; i<=n; i++)
       for (j=0; j<n; j++)
         sum++;
     ```

     Solución: Tomando como OB la suma, tenemos que

     $$tae(n) = \sum_{1}^{n} \sum_{0}^{n-1} 1 = \sum_{1}^{n} n = n^2$$

   - ```
     sum = 0;
     for (i=1; i<=n; i++)
       for (j=0; j<n*n; j++)
         sum++;
     ```

     Solución: Tomando como OB la suma, tenemos que

     $$tae(n) = \sum_{1}^{n} \sum_{0}^{n^2-1} 1 = \sum_{1}^{n} n^2 = n^3$$

Problemas

5. **(P; R)** Para los siguientes fragmentos de código, establecer una operación básica adecuada y estimar sus tiempos de ejecución en función de n:

- ```
  sum = 0;
  for (i=1; i<=n; i++)
    for (j=0; j<i; j++)
      sum++;
  ```

  Solución: Tomando como OB la suma, tenemos que

$$tae(n) = \sum_{i=1}^{n} \sum_{j=0}^{i-1} 1 = \sum_{1}^{n} i = \frac{n(n+1)}{2}$$

- ```
  sum = 0;
  for (i=1; i<=n; i++)
    for (j=0; j<i*i; j++)
      for (k=0; k<j; k++)
        sum++;
  ```

  Solución: Tomando como OB la suma, tenemos que

$$\begin{aligned} tae(n) &= \sum_{i=1}^{n} \sum_{j=0}^{i^2-1} \sum_{k=0}^{j-1} 1 = \sum_{i=1}^{n} \sum_{j=0}^{i^2-1} j \\ &= \sum_{i=1}^{n} \frac{1}{2}(i^2-1)\, i^2 = \frac{1}{2} \sum_{i=1}^{n} i^4 - \frac{1}{2} \sum_{i=1}^{n} i^2 \\ &= \frac{1}{10} n^5 + O(n^4) - \left(\frac{1}{6} n^3 + O(n^2) \right) = \frac{1}{10} n^5 + O(n^4) \end{aligned}$$

- ```
  sum = 0;
  for (i=1; i<=n; i++)
    for (j=0; j<i*i; j++)
      if (j%i == 0)
        for (k=0; k<j; k++)
          sum++;
  ```

  Solución: Tomando como OB la suma, tenemos que

$$\begin{aligned} tae(n) &= \sum_{i=1}^{n} \sum_{\{j=0,i,2i,\ldots,(i-1)i\}} j = \sum_{i=1}^{n} i \sum_{j=0}^{i-1} j \\ &= \sum_{i=1}^{n} \frac{1}{2} i\,(i-1)\, i = \frac{1}{2} \sum_{i=1}^{n} i^3 - \frac{1}{2} \sum_{i=1}^{n} i^2 \\ &= \frac{1}{8} n^4 + O(n^3) - \left(\frac{1}{6} n^3 + O(n^2) \right) = \frac{1}{8} n^4 + O(n^3) \end{aligned}$$
  ```

6. **(P)** Justificar las desigualdades $(A + B)/2 \leq \max(A, B) \leq A + B$. ¿Qué suponen para el cálculo del número de OBs sobre selecciones?

2 Crecimiento de funciones

Ejercicios

7. **(E; R)** Para un cierto problema se disponen de cinco algoritmos cuyos tiempos de ejecución en microsegundos para una entrada de tamaño N son respectivamente $10.000N$, $1.000N \log_{10} N$, $100N^2$, $10N^3$ y $10^{-3}10^{N/10}$. Dar una tabla con sus tiempos de ejecución para los valores de N 10, 100, 1.000, 10.000, 100.000 y 1.000.000.

   ¿Cuál es para cada algoritmo el tamaño máximo de una entrada para que ésta se ejecute en 1 minuto? ¿Y en un día?

   Solución: Consideramos la siguiente tabla de funciones y valores

	100	10000	1000000
$10000N$	10^6	10^8	10^{10}
$1000N \log_{10} N$	$2\,10^5$	$4\,10^7$	$6\,10^9$
$100N^2$	10^5	10^9	10^{13}

8. **(E; R)** Ordenar las siguientes funciones según su crecimiento: n, $\sqrt{n}$, $n^{1.5}$, n^2, $n \log n$, $n \log \log n$, $n \log^2 n$, $2/n$, 2^n, $2^{n/2}$, 37, $n^2 \log n$, n^3.

9. **(E)** Los tiempos de ejecución de tres algoritmos para la resolución de un mismo problema, para entradas de tamaño N, son N^2, $10N(\log_2 N)^2$ y $\frac{1}{8}N^2 \log_2 N$, respectivamente. Dar los rangos de valores de N en los que cada uno de estos algoritmos es más conveniente. (Considerar potencias de 2 para los límites de dichos rangos.)

10. **(E; R)** El análisis de dos algoritmos, A y B, muestra que $W_A(N) = \mathrm{O}(1000N \log_{10}(N))$ y $W_B(N) = \mathrm{O}(N^2)$. ¿Cuál es mejor para problemas "pequeños"? ¿Cuál es mejor para problemas "grandes"? Especificar qué sería pequeño y grande en este contexto.

    Solución: Consideramos la siguiente tabla de funciones y valores

    Aunque B es asintóticamente peor, en la práctica solo lo es cuando aproximadamente $N \geq 10^4$

11. **(E; R)** Probar que para todo $a > 0$, $\log n = \mathrm{o}(n^a)$.

    Solución: Se tiene

	10	100	1000	10000
$1000N \log_{10} N$	10^4	$2\,10^5$	$3\,10^6$	$4\,10^7$
N^2	10^2	10^4	10^6	10^8

$$\frac{\lg n}{n^a} = \frac{1}{\log 2}\frac{\log n}{n^a}$$

y por L'Hôpital,

$$\lim \frac{\log n}{n^a} = \lim \frac{1/n}{an^{a-1}} = \lim \frac{1}{an^a} = 0.$$

12. **(E+)** Probar que $\lfloor N \rfloor = \Theta(N)$ y que también $\lceil N \rceil = \Theta(N)$ (¿puede hacerse mediante límites?).

13. **(E+)** ¿Cuál es el orden de crecimiento teórico de la función $f(N) = N \log N + 100N$? ¿Cuál es su orden de crecimiento "real"?

    Solución: En teoría, $100N = o(N \log N)$, pero $100N < N \log N$ sii $\log N > 100$ sii $N > 10^{100}$, esto es, si N es mayor que un googol. Así que en la práctica el rendimiento real es $100N$.

14. **(E; R)** Se sabe que $T_1 \approx f$ y $T_2 \approx f$; decidir razonadamente la verdad o falsedad de las siguientes afirmaciones (dar un breve argumento en las ciertas y un contraejemplo o contraargumento en las falsas):

    (a) $T_1 + T_2 \approx f$;

    (b) $T_1 - T_2 = \mathrm{o}(f)$;

    (c) $T_1 \cdot T_2 \approx f^2$.

    Solución: En estos problemas

    (a) Si creemos que es verdad (V), hay que dar un argumento.

    (b) Si creemos que es falso (F), hay que dar un argumento o un contraejemplo.

    En este caso

    (a) Falso:

    $$\frac{T_1 + T_2}{f} = \frac{T_1}{f} + \frac{T_2}{f} \to 2 \neq 1$$

(b) Verdadero:

$$\frac{T_1 - T_2}{f} = \frac{T_1}{f} - \frac{T_2}{f} \to 0$$

(c) Verdadero:

$$\frac{T_1 \cdot T_2}{f^2} = \frac{T_1}{f} \cdot \frac{T_2}{f} \to 1 \cdot 1 = 1$$

15. **(E; R)** Se sabe que $T_1 \approx f$ y $T_2 \approx f$; decidir razonadamente la verdad o falsedad de las siguientes afirmaciones (dar un breve argumento en las ciertas y un contraejemplo o contraargumento en las falsas):

(a) $T_1 + T_2 \approx 2f$;

(b) $T_1^2 - T_2^2 = \mathrm{o}(f)$;

(c) $T_1/T_2 \approx 1$.

Solución: En este caso

(a) Verdadero:

$$\frac{T_1 + T_2}{2f} = \frac{T_1}{2f} + \frac{T_2}{2f} \to \frac{1}{2} + \frac{1}{2} = 1$$

(b) Podemos pensar que puede ser verdad y calculamos

$$\lim \frac{T_1^2 - T_2^2}{f} = \lim \frac{T_1 + T_2}{f} \cdot (T_1 - T_2) = 2 \lim (T_1 - T_2).$$

Pero si $T_1 = N^2 + N$ y $T_2 = f = N^2$, $T_1 - T_2 = N$, que tiende a ∞, luego es falso.

(c) Verdadero:

$$\frac{T_1}{T_2} = \frac{T_1}{f} \cdot \frac{1}{T_2/f} \to \frac{1}{1} = 1$$

Problemas

16. **(P; R)** Se sabe que $T_1(N) = \mathrm{O}(f(N))$ y que $T_2(N) = \mathrm{O}(f(N))$. ¿Cuál de las siguientes afirmaciones es cierta y cuál falsa?

(a) $T_1(N) + T_2(N) = \mathrm{O}(f(N))$;

(b) $T_1(N) - T_2(N) = \mathrm{o}(f(N))$;

(c) $\frac{T_1(N)}{T_2(N)} = \mathrm{O}(1)$;

(d) $T_1(N) = \mathrm{O}(T_2(N))$.

(dar un breve argumento para las afirmaciones ciertas y un contraejemplo para las falsas).

Solución: En este caso la argumentación puede ser más complicada pues no vamos a poder usar argumentos sobre límites.

(a) Fácilmente verdadero: por el resultado de una transparencia en teoría sabemos que

$$T_1 + T_2 = O(f + f) = O(2f) = O(f)$$

(b) Podemos pensar que "pinta" mal: las condiciones no involucran límites mientras que la afirmación sí. De hecho es fácil ver que es falsa: tomamos $T_1 = 2N^2$, $T_2 = f = N^2$. Entonces

$$\frac{T_1 - T_2}{f} = \frac{N^2}{N^2} = 1 \neq 0$$

(c) Tampoco parece que vaya a ser verdad: tomamos $T_1 = f = N^2$ y $T_2 = f$. Entonces $\frac{T_1}{T_2} = N$, que no es $O(1)$.

(d) Otra vez falso, y basta usar el mismo ejemplo de $T_1 = f = N^2$ y $T_2 = f$.

17. **(P; R)** Se sabe que $T_1(N) = \Theta(f(N))$ y que $T_2(N) = \Theta(f(N))$. ¿Cuál de las siguientes afirmaciones es cierta y cuál falsa?

(a) $T_1(N) + T_2(N) = \Theta(f(N))$;

(b) $T_1(N) - T_2(N) = \mathrm{o}(f(N))$;

(c) $\frac{T_1(N)}{T_2(N)} = \Theta(1)$;

(d) $T_1(N) = \Theta(T_2(N))$.

(dar un breve argumento para las afirmaciones ciertas y un contraejemplo para las falsas).

Solución: De nuevo, en este caso la argumentación va a ser más complicada pues tampoco vamos a poder razonar usando límites y hay que recurrir a las definiciones, lo que es más engorroso. Tenemos por tanto que usar argumentos teóricos o contraejemplos.

(a) Sabemos que $T_1 + T_2 = O(f + f) = O(f)$, pero también por ejemplo $f = O(T_1) = O(T_1 + T_2)$, esto es, $T_1 + T_2 = \Omega(f)$.
Juntando ambas, $T_1 + T_2 = \Theta(f)$

(b) Si hemos hecho el problema anterior, es claro que esta afirmación va a ser falsa: tomamos $T_1 = 2N^2$, $T_2 = f = N^2$. Entonces $T_1 - T_2 = N^2 = f$ que obviamente no es $o(f)$.

(c) Aquí hay que encadenar resultados parciales. Por ejemplo, como $T_1 = \Theta(T_2)$, $T_1 = O(T_2)$ y para un par C_1, N_1, $T_1(N) \leq C_1 T_2(N)$ para todo $N \geq N_1$, esto es

$$\frac{T_1}{T_2} \leq C_1 \implies \frac{T_1}{T_2} = O(1).$$

Y análogamente, como $T_2 = \Theta(T_1)$, $T_2 = O(T_1)$ y para un par C_2, N_2, $T_2(N) \leq C_2 T_1(N)$ para todo $N \geq N_2$, esto es

$$\frac{T_1}{T_2} \geq \frac{1}{C_2} \implies \frac{T_1}{T_2} = \Omega(1).$$

Por tanto, es cierto.

(d) Fácilmente verdadero: como $T_1 = O(f)$ y $f = O(T_2)$, se tiene $T_1 = O(T_2)$. Y análogamente, como $T_1 = \Omega(f)$ y $f = \Omega(T_2)$, se tiene $T_1 = \Omega(T_2)$.

18. **(P)** Diremos que $F = f + \mathrm{O}(g)$ si $g = \mathrm{o}(f)$ y $|F - f| = \mathrm{O}(g)$. Comprobar que $\sum_1^N i^k = N^{k+1}/(K+1) + O(N^K)$.

19. **(P+)** Probar que para todo N, $\log N! = \Theta(N \log N)$ y $N! = \mathrm{o}(N^N)$.

20. **(P; R)** ¿Es cierto que para cualquier f y cualquier constante C, $f(Cn) = \mathrm{O}(f(n))$? ¿Es cierto que $f(n + C) = \mathrm{O}(f(n))$?

Solución: En Teoría hemos mencionado que "en O, Θ las constantes no importan". Pero eso sólo es así cuando las constantes están "fuera", multiplicando f. Cuando están dentro la cosa cambia mucho.

Por ejemplo, si $f(n) = e^n$, $f(2n) = e^{2n} = (e^n)^2 = f^2$, esto es, aquí $f(n) = o(f(2n))$ y, por tanto, no es cierto que para toda f se tenga que $f(Cn) = o(f)$.

Sin embargo,

$$f(n + C) = e^{n+C} = e^C \, e^n = O(e^n).$$

y e^n no nos sirve para refutar la segunda afirmación.

El truco en el caso anterior ha sido coger una f que crezca muy rápido, pero ese crecimiento no ha sido suficiente en la segunda cuestión.

Pero tomando una f que crezca aun más rápido podemos ver que tampoco es cierto que $f(n+C) = O(f(n))$: basta tomar $f(n) = n!$ pues entonces

$$f(n+1) = (n+1)! = (n+1)\,n! = (n+1)\,f(n)$$

y entonces $f(n) = o(f(n+1))$.

3 Casos peor, mejor y medio de algoritmos básicos

Ejercicios

21. **(E; R)** Estimar razonadamente el crecimiento de la siguiente función:

$$S(N) = \sum_{1}^{N} \frac{1}{i^{1/3}}.$$

Solución: En estas estimaciones el objetivo es llegar a $S_N \sim g(N)$ y hay que dar casi siempre los mismos pasos:

(a) Comprobar si f es o no creciente.

(b) Acotar la suma entre dos integrales definidas y calcular éstas.

(c) "Opinar" sobre la posible función $g(N)$ escogiendo la opción más simple.

(d) Comprobar mediante un límite que la opinión ha sido correcta.

En este caso tomamos $f(x) = x^{1/3}$, que es creciente, por lo que

$$\begin{aligned}
\int_0^N x^{1/3}\,dx \;&\le S_N \le\; \int_1^{N+1} x^{1/3}\,dx \implies \\
\left[\frac{3}{4}x^{4/3}\right]_0^N dx \;&\le S_N \le\; \left[\frac{3}{4}x^{4/3}\right]_1^{N+1} \implies \\
\frac{3}{4}N^{4/3} \;&\le S_N \le\; \frac{3}{4}(N+1)^{4/3} - \frac{3}{4}
\end{aligned} \tag{1}$$

Esto sugiere que $S_N \sim \frac{3}{4}N^{4/3}$ y dividiendo todos los términos en (3), tenemos

$$1 \le\; \le \frac{S_N}{\frac{3}{4}N^{4/3}} \le\; \frac{\frac{3}{4}(N+1)^{4/3}}{\frac{3}{4}N^{4/3}} - \frac{\frac{3}{4}}{\frac{3}{4}N^{4/3}} = \left(\frac{N+1}{N}\right)^{4/3} - \frac{1}{N^{4/3}}$$

y como el lado derecho $\to 1$, $S_N \sim \frac{3}{4}N^{4/3}$.

22. **(E)** De una cierta tabla de tamaño N se sabe que $P(T[i]) = \frac{i^2}{C_N}$, donde C_N es una constante normalizadora. ¿Cuál será el número medio de comparaciones de clave que realizará sobre la misma el algoritmo de búsqueda lineal en búsquedas con éxito?

    Solución: En estas estimaciones el objetivo es llegar a $A^e_{BL}(N) \sim g(N)$ y hay que dar casi siempre los mismos pasos:

    (a) Escribir $A^e_{BL}(N)$ como $A^e(N) = S_N/C_N$.

    (b) Estimar $S_N \sim f(N)$, $C_N \sim g(N)$.

    (c) Demostrar que $A^e(N) \sim f(N)/g(N)$.

    En este caso, tenemos

$$A^e_{BLin}(N) = \frac{1}{C_N}\sum_1^N k \times k^2 = \frac{1}{C_N}\sum_1^N k^3 = \frac{S_N}{C_N}.$$

    Para C_N y S_N aplicando los resultados de teoría tenemos

$$C_N = \sum_1^N k^2 \sim \frac{1}{3}N^3 \quad S_N = \sum_1^N k^3 \sim \frac{1}{4}N^4$$

    Esto suigiere que $A^e(N) \sim \frac{\frac{N^4}{4}}{\frac{N^3}{3}} = \frac{3}{4}N$ y en efecto

$$\frac{A^e(N)}{\frac{3}{4}N} = \frac{S_N}{\frac{1}{4}N^4}\frac{1}{\frac{C_N}{\frac{1}{3}N^3}} \to 1 \cdot \frac{1}{1} = 1.$$

23. **(E; R)** Se quiere usar el siguiente pseudocódigo para búsquedas lineales en tablas ordenadas

```
int BLin(tabla T, dim N, clave K)
    i=1;
    mientras i <= N y T[i] < K:
        i++;
    si i > N:
        devolver error;
    else si T[i] != K:
        devolver error;
    else:
        devolver i;
```

Suponiendo que $P(K=i) = P(i<K<i+1) = P(i<1) = P(i>N) = \frac{1}{2N+1}$, calcular $A_{BLin}(N)$ considerando tanto búsquedas con acierto como con error.

Solución: Como en otros casos, definimos el espacio de entradas en función de las distintas claves que dan lugar a búsquedas con/sin éxito con diferentes costes. Viendo el seudocódigo y suponiendo $T = [1, 2, \ldots, N]$, se tiene

- Las claves con éxito son $\{1, \ldots, N\}$ y buscar con éxito la clave k cuesta k cdcs.
- Tomamos como claves de fracaso $k_0, k_1, \ldots, k_N$, con $k_0 < 1$, $j < k_j < j+1, 1 \leq j \leq N$-1, $N < k_N$, con costes de búsqueda 1 para k_0, $j+1$ para $k_j, 1 \leq j \leq N-1$, y N para k_N.

Por tanto

$$\begin{aligned} A_{BLO}(N) &= \sum_1^N k \times \frac{1}{2N+1} + \sum_1^N k \times \frac{1}{2N+1} + N \times \frac{1}{2N+1} \\ &= \frac{1}{2N+1}\frac{N(N+1)}{2} + \frac{1}{2N+1}\frac{N(N+1)}{2} + \frac{N}{2N+1} \\ &= \frac{N(N+1)}{2N+1} + \frac{N}{2N+1} = \frac{N}{2} + O(1). \end{aligned}$$

24. **(E; R)** Dada una tabla T, se sabe que

$$P(K = T[i]) = \frac{1}{C_N}\frac{\log i}{i},$$

donde $C_N = \sum_1^N \frac{\log i}{i}$.
Calcular razonadamente $A_{BLin}(N)$.

25. **(E)** Dada una tabla T, se sabe que

$$P(K = T[i]) = \frac{1}{C_N} i \log i,$$

donde $C_N = \sum_1^N i \log i$. Calcular razonadamente $A^e_{BLin}(N)$.

Solución: En este caso, tenemos

$$A^e_{BLin}(N) = \frac{1}{C_N}\sum_1^N k \times k \log k = \frac{1}{C_N}\sum_1^N k^2 \log k = \frac{S_N}{C_N}.$$

Para C_N tenemos $C_N = \sum_1^N k \log k$ y por tanto

$$\begin{aligned}
\int_0^N x \log x \, dx \quad &\leq C_N \leq \quad \int_1^{N+1} x \log x dx \implies \\
\left[\frac{1}{2}x^2 \log x - \frac{1}{4}x^2\right]_0^N dx \quad &\leq C_N \leq \quad \left[\frac{1}{2}x^2 \log x - \frac{1}{4}x^2\right]_1^{N+1} \implies \\
\frac{N^2}{2}\log N - \frac{N^2}{4} \quad &\leq C_N \leq \quad \frac{(N+1)^2}{2}\log(N+1) - \frac{(N+1)^2}{4} + \frac{1}{4} \qquad (2)
\end{aligned}$$

Esto sugiere que $C_N \sim \frac{N^2}{2}\log N$, lo que se demuestra dividiendo todo por $\frac{N^2}{2}\log N$ y calculando el límite para $N \to \infty$.

Análogamente para S_N tenemos $S_N = \sum_1^N k^2 \log k$ y por tanto

$$\begin{aligned}
\int_0^N x^2 \log x \, dx \quad &\leq S_N \leq \quad \int_1^{N+1} x^2 \log x dx \implies \\
\left[\frac{1}{3}x^3 \log x - \frac{1}{9}x^3\right]_0^N dx \quad &\leq S_N \leq \quad \left[\frac{1}{3}x^3 \log x - \frac{1}{9}x^3\right]_1^{N+1} \implies \\
\frac{N^3}{3}\log N - \frac{N^3}{9} \quad &\leq S_N \leq \quad \frac{(N+1)^3}{3}\log(N+1) - \frac{(N+1)^3}{9} + \frac{1}{9} \qquad (3)
\end{aligned}$$

Esto sugiere que $S_N \sim \frac{N^3}{3}\log N$, lo que se demuestra dividiendo todo por $\frac{N^3}{3}\log N$ y calculando el límite para $N \to \infty$.

En consecuencia cabe esperar que $A^e(N) \sim \left(\frac{N^3}{3}\log N\right) / \left(\frac{N^2}{2}\log N\right) = \frac{2}{3}N$, lo que comprobamos como

$$\frac{A^e(N)}{\frac{2}{3}N} = \frac{S_N}{\frac{1}{3}N^3 \log N} \frac{1}{\frac{C_N}{\frac{1}{2}N^2 \log N}} \to 1 \cdot \frac{1}{1} = 1.$$

Problemas

26. **(E)** Comprobar por inducción la veracidad de la identidad $\sum_1^N a + ib = N(a + \frac{N+1}{2}b)$.

27. **(P; R)** Demostrar por inducción las siguientes fórmulas:
$\sum_1^N n^2 = n(n+1)(2n+1)/6$,
$\sum_1^N n^3 = (n(n+1)/2)^2$.

28. **(P+)** Dar una fórmula para la suma $\sum_1^N nx^n$, y también para $\sum_1^N n^2x^n$. (Observése que la primera se parece a una derivada y la segunda no está demasiado lejos.)

4 Evolución de algoritmos locales

Ejercicios

29. **(E)** Dada la lista `[20, 3, 10, 6, 8, 2, 13, 17]`, indicar el estado de la misma tras cada una de las iteraciones del método de la burbuja y del de inserción.

    Solución: Lo hacemos sobre la tabla `[20, 3, 10, 6, 8]`. Para la burbuja marcamos el elemento burbuja inicial como e_b y tenemos

i	tabla inicial	tabla final	num. cdcs	flag swap
5	$[20_b, 3, 10, 6, 8]$	$[3, 10, 6, 8\|20]$	4	1
4	$[3_b, 10, 6, 8\|20]$	$[3, 6, 8\|10, 20]$	3	1
3	$[3_b, 6, 8\|10, 20]$	$[3, 6\|8, 10, 20]$	2	0

    y no se entra en la última iteración. El número de cdcs es 9.

    Para la inserción marcamos como e_i el elemento a insertar y tenemos

i	tabla inicial	tabla final	num. cdcs
2	$[20\|3_i, 10, 6, 8]$	$[3, 20\|10, 6, 8]$	1
3	$[3, 20\|10_i, 6, 8]$	$[3, 10, 20\|6, 8]$	2
4	$[3, 10, 20\|6_i, 8]$	$[3, 6, 10, 20\|8]$	3
5	$[3, 6, 10, 20\|8_i]$	$[3, 6, 8, 10, 20]$	3

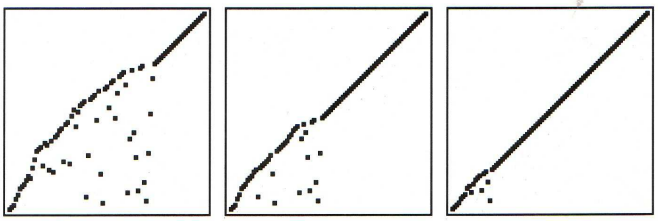
    El número de cdcs es también 9.

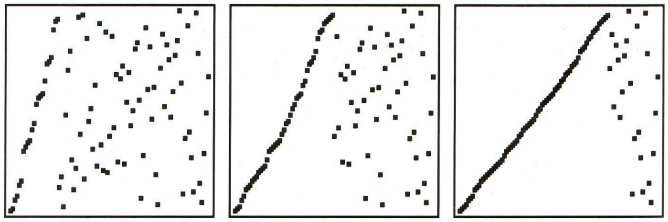
30. **(E)** Sobre la lista `16 6 3 4 9 31 1 8 2 7`, dar razonadamente la evolución del método de inserción, indicando el estado de la tabla tras la ubicación de cada elemento, y el número de comparaciones de elementos realizadas en dicha ubicación.

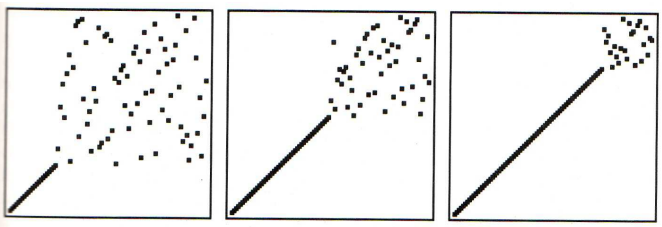
31. **(E)** Sobre la lista `8 18 5 6 10 3 4 11 31 7`, dar razonadamente la evolución del método de la burbuja, indicando para la primera iteración qué elementos se usan como burbuja y detallando los estados parciales de la tabla, y para las demás iteraciones el estado final de la tabla tras las mismas. ¿Cuántas iteraciones se harán?

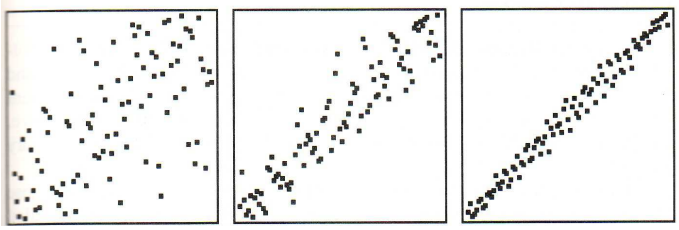
Problemas

32. **(P; R)** En los cuadros de la figura inferior se representan tres fases consecutivas de la evolución de una lista de 50 números entre 1 y 50 a la que se aplican ciertos métodos de ordenación. Las coordenadas (i, j) de un punto del gráfico indican que el número de la posición i–ésima de la lista es precisamente j (por ejemplo, un número i está en su posición correcta si $j = i$, esto es, si el par (i, j) está en la diagonal). ¿Cuáles de los métodos de ordenación estudiados en el curso podráa generar estos gráficos?









5 Eficacia de algoritmos locales

Ejercicios

33. **(E; R)** Dadas las permutaciones de S_6 (6 1 5 2 4 3), (6 5 4 1 2 3) y (2 3 6 4 5 1), calcular sus traspuestas, encontrar las inversiones de ambas y comprobar el resultado del problema anterior.

    Si se les aplica un algoritmo de ordenación local, ¿cuántas cdcs efectuará como mínimo sobre ellas?

    Solución: Lo hacemos con $\sigma = (615243)$:

$$\begin{aligned} inv(\sigma) &= inv(6) + inv(1) + inv(5) + inv(2) + inv(4) + inv(3) \\ &= 5 + 0 + 3 + 0 + 1 + 0 = 9. \end{aligned}$$

    Por tanto, si A es un algoritmo local

$$n_A(\sigma) \geq inv(\sigma) = 9.$$

34. **(E; R)** ¿Cuántas comparaciones de clave realizará como mínimo un algoritmo de ordenación local sobre la permutación de $N = 2K$ elementos `(2K 2K-1 2K-2 ... K+1 1 2 3 ... K)`?

35. **(E; R)** Expresar en función de N cuantas comparaciones de clave realizará como mínimo un algoritmo local si se aplica a la permutación de $N = 3K$ elementos `(2K+1, 2K+2, ..., 3K, 2K, 2K-1, ... K+2, K+1, 1, 2, ..., K)`

    Solución: Al ser A local, se cumple $n_A(\sigma) \geq inv(\sigma)$ por lo que calculamos $inv(\sigma)$.

    En principio, $inv(\sigma)$ ha de calcularse sumando las inversiones de cada uno de sus elementos. Sin embargo, en este ejercicio y los siguientes σ tiene una estructura de bloques que simplifica el cálculo. En este caso tenemos

$$\sigma = (\underbrace{2K+1, 2K+2, ..., 3K}_{\text{bloque 1}}, \underbrace{2K, 2K-1, ...K+2, K+1}_{\text{bloque 2}}, \underbrace{1, 2, ..., K}_{\text{bloque 3}}),$$

    y en cada bloque podemos considerar sus inversiones internas (esto es, dentro del propio bloque) y externas (esto es, respecto a los demás bloques. Por ejemplo

    (a) Para el bloque 1 tenemos

- $inv_{internas}(\text{bloque } 1) = 0$, pues el bloque está ordenado,
- $inv_{externas}(\text{bloque } 1) = K \times 2K = 2K^2$, pues cada uno de sus K elementos está invertido con cada uno de los $2K$ elementos de los bloques 2 y 3.

(b) Para el bloque 2 tenemos

- $inv_{internas}(\text{bloque } 2) = \frac{K(K-1)}{2}$, pues el bloque está totalmente desordenado,
- $inv_{externas}(\text{bloque } 2) = K \times K = K^2$, pues cada uno de sus K elementos está invertido con cada uno de los K elementos del bloque 3.

(c) Y para el bloque 3 tenemos

- $inv_{internas}(\text{bloque } 3) = 0$ pues el bloque está ordenado,
- $inv_{externas}(\text{bloque } 3) = 0$, pues es el último bloque.

En total se tiene entonces

$$inv(\sigma) = 2K^2 + \frac{K^2}{2} - \frac{K}{2} + K^2 = \frac{7}{2}K^2 - \frac{K}{2}$$

y como $K = N/3$,

$$inv(\sigma) = \frac{7}{2}\frac{N^2}{9} - \frac{N}{6} = \frac{7N^2}{18} - \frac{N}{6}$$

y, por tanto

$$n_A(\sigma) \geq \frac{7N^2}{18} - \frac{N}{6}.$$

36. **(E)** Expresar en función de N cuántas comparaciones de clave realizará como mínimo un algoritmo local si se aplica a la permutación de $N = 4K$ elementos

`(3K+1, 3K+2, ..., 4K, 3K, 3K-1, ..., 2K+2, 2K+1, 2K, 2K-1, ... K+2, K+1, 1, 2, ..., K)`

Solución: Razonamos como en un problema anterior. Al ser A local, se cumple $n_A(\sigma) \geq inv(\sigma)$ por lo que calculamos $inv(\sigma)$ tras descomponerla en bloques, en concreto

$$\sigma = (\underbrace{3K+1, 3K+2, ..., 4K}_{\text{bloque 1}}, \underbrace{3K, 3K-1, ..., 2K+2, 2K+1, 2K, 2K-1, ...K+2, K+1}_{\text{bloque 2}}, \underbrace{1, 2, ..., K}_{\text{bloque 3}}),$$

y en cada bloque podemos considerar sus inversiones internas (esto es, dentro del propio bloque) y externas (esto es, respecto a los demás bloques.

(a) Para el bloque 1 tenemos

- $inv_{internas}(\text{bloque } 1) = 0$, pues el bloque está ordenado,
- $inv_{externas}(\text{bloque } 1) = K \times 3K = 3K^2$, pues cada uno de sus K elementos está invertido con cada uno de los $3K$ elementos de los bloques 2 y 3.

(b) Para el bloque 2 tenemos

- $inv_{internas}(\text{bloque } 2) = \frac{2K(2K-1)}{2}$, pues el bloque tiene $2K$ ekementos y está totalmente desordenado,
- $inv_{externas}(\text{bloque } 2) = 2K \times K = 2K^2$, pues cada uno de sus $2K$ elementos está invertido con cada uno de los K elementos del bloque 3.

(c) Y para el bloque 3 tenemos

- $inv_{internas}(\text{bloque } 3) = 0$ pues el bloque está ordenado,
- $inv_{externas}(\text{bloque } 3) = 0$, pues es el último bloque.

En total y como $K = N/4$, se tiene entonces

$$inv(\sigma) = 3K^2 + \frac{4K^2}{2} - \frac{2K}{2} + 2K^2 = 7K^2 - K = \frac{7N^2}{32} - \frac{N}{4}$$

y $inv(\sigma) = \frac{7N^2}{16} - \frac{N}{4}$. y, por tanto

$$n_A(\sigma) \geq \frac{7N^2}{18} - \frac{N}{6}.$$

37. **(E; R)** Decidir razonadamente el número de comparaciones de clave que un algoritmo de ordenación local hará como mínimo sobre la permutación de $N = 3K$ elementos

$$(\mathtt{3K}, \mathtt{3K}-\mathtt{1}, , ..., \mathtt{2K}+\mathtt{1}, \mathtt{K}+\mathtt{1}, \mathtt{K}+\mathtt{2}, ..., \mathtt{2K}, \mathtt{K}, \mathtt{K}-\mathtt{1}, ..., \mathtt{2}, \mathtt{1}).$$

Problemas

38. **(P)** Comprobar mediante la definición de permutación traspuesta que dada una permutación σ, se cumple que $(\sigma^\tau)^\tau$ coincide con σ.

39. **(E+; R)** Se aplica un algoritmo local A de ordenación a la permutación de $N = 2K$ elementos

```
(1  2K  2  2K-1  3  2K-2  ...   K  K+1).
```

¿Cuántas comparaciones de clave efectuará A sobre dicha permutación?

Solución: En este caso la estructura de bloques disjuntos no se produce. Por ello calculamos las inversiones de σ sumando las correspondientes a cada elemento. En concreto, escribimos las inversiones debajo de cada elemento:

1	$2K$	2	$2K-1$	3	$2K-2$	$\ldots$	$K-2$	$K+3$	$K-1$	$K+2$	K	$K+1$
0	$2K-2$	0	$2K-4$	0	$2K-6$	$\ldots$	0	4	0	2	0	0

Por tanto, sumando la fila inferior de derecha a izquierda y como $K = N/2$, se tiene

$$\begin{aligned} inv(\sigma) &= 2+4+6+\ldots+2K-6+2K-4+2K-2 \\ &= 2\times 1+2\times 2+2\times 3+\ldots+2\times(K-3)+2\times(K-2)+2\times(K-1) \\ &= 2(1+2+3+\ldots+K-3+K-2+K-1) = 2\frac{K(K-1)}{2} = \frac{N^2}{4}-\frac{N}{2} \end{aligned}$$

y para cualquier A local, $n_A(\sigma) \geq \frac{N^2}{4} - \frac{N}{2}$.

40. **(E+; R)** Decidir razonadamente el número de comparaciones de clave que un algoritmo de ordenación local hará como mínimo sobre la permutación de $N = 2K$ elementos `(2K, 1, 2K-1, 2, ..., 2K-i, i+1, ..., K+1, K)`.

Solución: Procedemos como en el problema anterior: escribimos las inversiones debajo de cada elemento:

$2K$	1	$2K-1$	2	$2K-2$	3	$\ldots$	$2K-i$	$i+1$	$\ldots$	$K+2$	$K-1$	$K+1$	K
$2K-1$	0	$2K-3$	0	$2K-5$	0	$\ldots$	$2K-1-2i$	0	$\ldots$	3	0	1	0

Por tanto, sumando la fila inferior de derecha a izquierda y como $K = N/2$, se tiene

$$\begin{aligned} inv(\sigma) &= 1+3+5+\ldots+2K-5+2K-3+2K-1 \\ &= (2\times 0+1)+(2\times 1+1)+(2\times 2+1)+\ldots+(2\times(K-2)+1)+(2\times(K-1)+1) \\ &= [2\times(0+1+2+3+\ldots+K-3+K-2+K-1)]+K = K+2\sum_{0}^{K-1} i \\ &= K+\frac{K(K-1)}{2} = \frac{N^2}{4}+\frac{N}{2} \end{aligned}$$

y para cualquier A local, $n_A(\sigma) \geq \frac{N^2}{4} + \frac{N}{2}$.

41. **(P)** Calcular el valor de $W_{Burbuja}(N)$, $A_{Burbuja}(N)$, $W_{Seleccion}(N)$ y $A_{Seleccion}(N)$.

42. **(E)** Se define $B_{\mathcal{A}}(N)$ como $B_{\mathcal{A}}(N) = \min\{n_{\mathcal{A}}(I) : I \in E_N^{\mathcal{A}}\}$. Analizar $B_{Insert}(N)$, $B_{Selec}(N)$, $B_{Burbuja}(N)$.

43. **(E+)** Modificar el psc del método de Inserción para que pueda trabajar con índices iniciales P y finales U cualesquiera.

44. **(P; R)** Analizar $B_{Insert}(N)$, $B_{Selec}(N)$, $B_{Burbuja}(N)$ usando el intercambio de elementos como operación básica. Analizar también $W_{Select}(N)$, y $W_{Burbuja}(N)$ bajo este punto de vista.

    Solución: Consideramos InserSort. Denotando ahora $n_{IS}(\sigma)$ y $n_{IS}(\sigma, i)$ el número total de swaps que hace InserSort sobre σ y las que hace en la iteración i. Obviamente tenemos

    $$0 \leq n_{IS}(\sigma, i) \leq i - 1,$$

    y por tanto

    $$n_{IS}(\sigma) = \sum_{2}^{N} n_{IS}(\sigma, i) \geq 0;$$

    además $n_{IS}([1, 2, \ldots, N]) = 0$, por lo que $B_{IS}(N) = 0$.

45. **(P+; R)** Dar una estimación aproximada de $A_{Burbuja}(N)$ suponiendo que en la misma se utiliza el siguiente psc:

```
Burbuja(tabla T, dim N)
  switch = 1;
  long   = N;
  mientras switch == 1 y long > 1:
    switch = 0;
    para i de 1 a long-1:
      si T[i] > T[i+1]:
        intercambiar T[i] y T[i+1];
        switch = 1;
    long--;
```

46. **(P; R)** El siguiente pseudocódigo es una ligera variante del habitualmente utilizado en la ordenación por Inserción:

```
InsertSort(tabla T, dim N)
  para i de 2 a N:
    j=i;
    mientras j > 1 y T[j-1] > T[j]:
      intercambiar T[j-1] y T[j];
      j--;
  volver;
```

Utilizando el intercambio de elementos como operación básica, calcular razonadamente $W_{InserSort}(N)$ y $A_{InserSort}(N)$.

Solución: Razonando como en un problema anterior,

$$n_{IS}(\sigma) = \sum_{2}^{N} n_{IS}(\sigma, i) \leq \sum_{2}^{N} (i-1) = \sum_{1}^{N-1} j = \frac{N(N-1)}{2}.$$

y como $W_{IS}(N) \geq n_{IS}([N, N-1, N-2, \ldots, 2, 1]) = \frac{N(N-1)}{2}$,

$$W_{IS}(N) = \frac{N(N-1)}{2}.$$

Para el caso medio tenemos que en la iteración i se pueden hacer $0, 1, 2, \ldots, i-1$ swaps. Suponiendo todas ellas equiprobables, su probabilidad individual es $1/i$, y el número medio de swaps en la iteración i sería

$$A_{IS}(N, i) = \sum_{j=0}^{i-1} j \times \frac{1}{i} = \frac{1}{i} \sum_{0}^{i-1} j = \frac{1}{i} \frac{i(i-1)}{2} = \frac{i-1}{2}$$

y por tanto,

$$A_{IS}(N) = \sum_{i=2}^{N} A_{IS}(N, i) = \frac{1}{2} \sum_{i=2}^{N} (i-1) = \frac{1}{4} N(N-1).$$

47. **(P)** Un algoritmo de ordenación se dice **estable** si al aplicarlo a una tabla con elementos tal vez repetidos, dos elementos con la misma clave (y por tanto relativamente ya bien ordenados) mantienen sus posiciones relativas al finalizar el algoritmo. ¿Cuáles de los métodos de ordenación estudiados son estables? (Modificar su psc si fuera necesario.)