Ana Cristina Sánchez Vázquez
179484

Tarea

1. Para cada función $f(n)$ y tiempo t determine el tamaño máximo del problema (la n) que puede resolverse en tiempo t. Suponga que el algoritmo usado para resolver el problema toma $f(n)$ microsegundos (reporte sólo el orden de magnitud si los números son demasiado grandes)

	1 (1) Segundo	1 (60) Minuto	1 (3600) Hora	1 (86400) Día	1 (1,123,200) Mes	1 (13,478,400) Año	1 (1.3 x 10^9) Siglo
$\log_2(n)$	2^{10^8}	$2^{6\times10^9}$	$2^{3.6\times10^{11}}$	$2^{8.6\times10^{12}}$	$2^{(8.6\times10^{12})(30)}$	$2^{(8.6\times10^{12})(12)(30)}$	$2^{(8.6\times10^{12})(100)(12)(30)}$
$\sqrt{n}$	10^{16}	3.6×10^{9}	$((3.6)\times10^{9})^2$	$(8.6\times10^{12})^2$	$(8.6\times10^{12})^2(900)$	$(8.6\times10^{12})^2(900)(144)$	$(8.6\times10^{12})^2(900)(10000)(144)$
N	10^8	6×10^9	3.6×10^{11}	8.6×10^{12}	$(8.6\times10^{12})(30)$	$(8.6\times10^{12})(30)(12)$	$(8.6\times10^{12})(30)(12)(100)$
$n\log_2(n)$	$n\log_2(n)=10^8$	$n\log_2(n)=6\times10^9$	$n\log_2(n)=3\times10^{9}$	$n\log_2(n)=8.6\times10^{12}$	$n\log_2(n)=(8.6\times10^{12})(30)$	$n\log_2(n)=(8.6\times10^{12})(30)(12)$	$n\log_2(n)=(8.6\times10^{12})(30)(12)(100)$
n^2	$\sqrt{10^8}$	$\sqrt{6\times10^9}$	$\sqrt{3.6\times10^{9}}$	$\sqrt{8.6\times10^{12}}$	$\sqrt{(8.6\times10^{12})(30)}$	$\sqrt{(8.6\times10^{12})(30)(12)}$	$10\sqrt{(8.6\times10^{12})(30)(12)}$
n^3	$\sqrt[3]{10^8}$	$\sqrt[3]{6\times10^9}$	$\sqrt[3]{3.6\times10^{9}}$	$\sqrt[3]{8.6\times10^{12}}$	$\sqrt[3]{(8.6\times10^{12})(30)}$	$\sqrt[3]{(8.6\times10^{12})(30)(12)}$	$\sqrt[3]{(8.6\times10^{12})(30)(12)(100)}$
2^n	$\log_2(10^8)$	$\log_2(6\times10^9)$	$\log_2(3.6\times10^{9})$	$\log_2(8.6\times10^{12}$	$\log_2((8.6\times10^{12})(30))$	$\log_2 (8.6\times10^{12})(30)(12)$	$\log_2((8.6\times10^{12})(30)(12)(100))$
$n!$	$n!=10^8$	$n!=6\times10^9$	$n!=3.6\times10^{9}$	$n!=8.6\times10^{12}$	$n!=(8.6\times10^{12})(30)$	$n!=(8.6\times10^{12})(30)(12)$	$n!=(8.6\times10^{12})(30)(12)(100)$

2. Supongamos que estamos comparando el desempeño de dos algoritmos de ordenamiento. Para entradas de tamaño n, el algoritmo A toma $8n^2$ operaciones mientras que el algoritmo B toma $64n\log_2(n)$. ¿Para qué valores de n es mejor el desempeño de A?
3. ¿Cuál es el valor más chico de n para el cual un algoritmo que toma $100n^2$ es más rápido que uno que toma 2^n (en la misma máquina)?
4. Demuestre que $2^n = O(n^2)$

---

2. A es mejor cuando $\boxed{1 < n < 43}$

3. $100n^2$ es mejor que 2^n cuando $\boxed{n > 14}$

4. Demuestre que:

$2^n = O(n^2)$

$O(g(n)) = \{f(n) \mid \exists\ c>0,\ n_0>0 \ \ni\ 0 \le f(n) \le c\,g(n)\ \forall n > n_0\}$

$g(n) = n^2$
$f(n) = 2^n$

$0 \le 2^n \le cn^2$

$0 \le cn^2 - 2^n$

$0 \le \ln\left(\frac{cn^2}{2^n}\right)$

cualquier logaritmo es ≥ 0 $\therefore$ c y n pueden tomar cualquier valor ≥ 0

Ana Cristina Sánchez Vázquez
179484

Tarea