Αλγόριθμοι και πολυπλοκότητα

1η Σειρά Γραπτών ασκήσεων

Χειμερινό Εξάμηνο 2013-2014 Μπογιόκας Δημήτριος - ΜΠΛΑ

Άσκηση 1(α) Ισχύουν τα:

(i) $5n = \Theta(n)$

(ii) $(\log n)^{\log n} = 2^{\log n \log\log n} = n^{\log\log n} = \Theta\left(n^{\log\log n}\right)$

(iii) $\frac{\log(n!)}{n} = \Theta\left(\frac{n\log n}{n}\right) = \Theta\left(\log n\right)$

(iv) $n \cdot 2^{2^{2^{100}}} = \Theta\left(n\right)$

(v) $\log\left(\binom{n}{6}\right) = \log\left(\frac{n(n-1)\cdots(n-5)}{6!}\right) = \Theta\left(\log\left(n^6\right)\right) = \Theta\left(\log n\right)$

(vi) $\sum_{k=1}^{n} k^5 = \Theta\left(n^6\right)$

(vii) $\log^4 n = \Theta\left(\log^4 n\right)$

(viii) $\sqrt{n!} = \Theta\left(\sqrt{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}\right) = \Theta\left(\frac{n^{\frac{1}{4}} n^{\frac{n}{2}}}{\sqrt{e^n}}\right)$ (από τον τύπο του Stirling)

(ix) $e^n = \Theta\left(e^n\right)$

(x) $\frac{n^2}{\log^{10} n} = \Theta\left(\frac{n^2}{\log^{10} n}\right)$

(xi) $(\log n)^{\log(16n)} = (\log n)^4 (\log n)^{\log n} = \Theta\left(n^{\log\log n}\log^4 n\right)$

(xii) $\log\left(\binom{2n}{n}\right) = \log\left((2n)!\right) - 2\log\left(n!\right) = \Theta\left(n\right)$

Πράγματι, από το Θεώρημα Cesáro-Stolz, ισχύει:

$$\begin{aligned}
\lim_{n\to\infty} \frac{\log((2n)!) - 2\log(n!)}{n} &= \lim_{n\to\infty} \frac{\log((2n+2)!) - 2\log((n+1)!) - \log((2n)!) + 2\log(n!)}{n+1-n} = \\
&= \lim_{n\to\infty}\left(\log\left((2n+1)(2n+2)\right) - \log\left((n+1)^2\right)\right) \\
&= \lim_{n\to\infty} \log\frac{4n^2+5n+2}{n^2+2n+1} \\
&= \log 4 = 2 > 0
\end{aligned}$$

(xiii) $n\left(2.5\right)^n = \Theta\left(n\left(2.5\right)^n\right)$

(xiv) $\binom{2n}{n} = \Theta\left(\frac{4^n}{n^{\frac{1}{2}}}\right)$

Πράγματι, από τον τύπο του Stirling, ισχύει:

$$\frac{(2n)!}{(n!)^2} = \Theta\left(\frac{\sqrt{2\pi 2n}\left(\frac{2n}{e}\right)^{2n}}{\sqrt{2\pi n}^2\left(\frac{n}{e}\right)^{2n}}\right) = \Theta\left(\frac{1}{\sqrt{\pi n}} \cdot 2^{2n}\right) = \Theta\left(\frac{4^n}{n^{\frac{1}{2}}}\right)$$

(xv) $\sum_{i=1}^{n} i2^i = \Theta\left(n2^n\right)$

Πράγματι, ισχύει: $\sum_{i=1}^{n} i2^i = \sum_{i=1}^{n}\sum_{j=1}^{i} 2^i = \sum_{j=1}^{n}\sum_{i=j}^{n} 2^i = \sum_{j=1}^{n} 2^j \sum_{i=j}^{n} 2^{i-j} = \sum_{j=1}^{n} 2^j \sum_{s=0}^{n-j} 2^s = \sum_{j=1}^{n} 2^j \frac{2^{n-j+1}-1}{2-1} = \sum_{j=1}^{n} 2^{n+1} - \sum_{j=1}^{n} 2^j = n2^{n+1} - 2\frac{2^{n+1}-1}{2-1} = (n-1)\,2^{n+1} - 1 = \Theta\left(n2^n\right)$

(xvi) $\sum_{i=1}^{n} \frac{i}{2^i} = \Theta\left(1\right)$

Πράγματι, ισχύει: $\sum_{i=1}^{n} \frac{i}{2^i} = \sum_{i=1}^{n}\sum_{j=1}^{i}\left(\frac{1}{2}\right)^i = \sum_{j=1}^{n}\sum_{i=j}^{n}\left(\frac{1}{2}\right)^i = \sum_{j=1}^{n}\left(\frac{1}{2}\right)^j \sum_{i=j}^{n}\left(\frac{1}{2}\right)^{i-j} =$

$\sum_{j=1}^{n}\left(\frac{1}{2}\right)^{j}\sum_{s=0}^{n-j}\left(\frac{1}{2}\right)^{s} = \sum_{j=1}^{n}\left(\frac{1}{2}\right)^{j}\frac{1-\left(\frac{1}{2}\right)^{n-j+1}}{1-\frac{1}{2}} = \sum_{j=1}^{n}\left(\frac{1}{2}\right)^{j-1} - \sum_{j=1}^{n}\left(\frac{1}{2}\right)^{n} = \frac{1-\left(\frac{1}{2}\right)^{n}}{1-\frac{1}{2}} - \left(\frac{1}{2}\right)^{n} n = 2-(n+2)\left(\frac{1}{2}\right)^{n}$, όπου $\frac{1}{2} \leq 2-(n+2)\left(\frac{1}{2}\right)^{n} \leq 2$, για $n \geq 1$.

Άρα, η λύση είναι:

$\mathcal{O}\left(\sum_{i=1}^{n}\frac{i}{2^i}\right) \subset \mathcal{O}\left(\frac{\log(n!)}{n}\right) = \mathcal{O}\left(\log\left(\binom{n}{6}\right)\right) \subset \mathcal{O}\left(\log^4 n\right) \subset \mathcal{O}\left(5n\right) = \mathcal{O}\left(n \cdot 2^{2^{2^{100}}}\right) = \mathcal{O}\left(\log\left(\binom{2n}{n}\right)\right) \subset \mathcal{O}\left(\frac{n^2}{\log^{10} n}\right) \subset \mathcal{O}\left(\sum_{k=1}^{n} k^5\right) \subset \mathcal{O}\left((\log n)^{\log n}\right) \subset \mathcal{O}\left((\log n)^{16\log n}\right) \subset \mathcal{O}\left(\sum_{i=1}^{n} i2^i\right) \subset \mathcal{O}\left(n\left(2.5\right)^n\right) \subset \mathcal{O}\left(e^n\right) \subset \mathcal{O}\left(\binom{2n}{n}\right) \subset \mathcal{O}\left(\sqrt{n!}\right)$

Άσκηση 1(β)

(i) Έστω $m = \log_5 n$, τότε:

$$\begin{aligned}
T(n) &= 5T\left(\frac{n}{5}\right) + n\log n \\
&= 5^2 T\left(\frac{n}{5^2}\right) + n\log\frac{n}{5} + n\log n \\
&\vdots \\
&= 5^k T\left(\frac{n}{5^k}\right) + n\sum_{i=0}^{k-1}\log\frac{n}{5^i} \quad \forall k \\
&\overset{k=m}{=} nT(1) + n\sum_{i=0}^{m-1}\log\frac{5^m}{5^i} \\
&= nT(1) + n\log 5\sum_{i=0}^{m-1}(m-i) \\
&= nT(1) + n\frac{m(m+1)}{2}\log 5 = \Theta\left(n\log^2 n\right)
\end{aligned}$$

(ii) Έστω $m = \log_{10} n$, τότε:

$$\begin{aligned}
T(n) &= 9T\left(\frac{n}{10}\right) + \log^3 n \\
&= 9^2 T\left(\frac{n}{10^2}\right) + 9\log^3\frac{n}{10} + \log^3 n \\
&\vdots \\
&= 9^k T\left(\frac{n}{10^k}\right) + \sum_{i=0}^{k-1} 9^i \log^3\frac{n}{10^i} \quad \forall k \\
&\overset{k=m}{=} 9^{\log_{10} n} T(1) + \sum_{i=0}^{m-1} 9^i \log^3\frac{10^m}{10^i} \\
&= n^{\log_{10} 9} T(1) + \sum_{i=0}^{m-1} 9^i (m-i)^3 \\
&= n^{\log_{10} 9} T(1) + 9^m \log^3 10 \sum_{s=1}^{m}\frac{s^3}{9^s} \\
&= n^{\log_{10} 9} T(1) + n^{\log_{10} 9} c = \Theta\left(n^{\log_{10} 9}\right)
\end{aligned}$$

όπου $0 < \frac{1}{9}\log^3 10 < c < \sum_{i=1}^{\infty}\frac{s^3}{9^s}\log^3 10 < +\infty \Rightarrow c \in \Theta(1)$

(iii) Έστω $m = \log_3 n$, τότε:

$$
\begin{aligned}
T(n) &= 2T\left(\tfrac{n}{3}\right)+\tfrac{n}{\log^2 n}\\
&= 2^2T\left(\tfrac{n}{3^2}\right)+\tfrac{3}{2}\tfrac{n}{\log^2\frac{n}{3}}+\tfrac{n}{\log^2 n}\\
&\vdots\\
&= 2^kT\left(\tfrac{n}{3^k}\right)+n\textstyle\sum_{i=0}^{k-1}\left(\tfrac{2}{3}\right)^i\tfrac{1}{\log^2\frac{n}{3^i}}\quad\forall k\\
&\overset{k=m}{=} 2^{\log_{10}n}T(1)+n\textstyle\sum_{i=0}^{m-1}\left(\tfrac{2}{3}\right)^i\tfrac{1}{\log^2\frac{3^m}{3^i}}\\
&= n^{\log_3 2}T(1)+n\log^{-2}3\textstyle\sum_{i=0}^{m-1}\left(\tfrac{2}{3}\right)^i\tfrac{1}{(m-i)^2}\\
&= n^{\log_3 2}T(1)+\tfrac{n}{\log_3^2 n}\log^{-2}3\textstyle\sum_{i=0}^{m-1}\left(\tfrac{2}{3}\right)^i\tfrac{1}{\left(1-\frac{i}{m}\right)^2}\\
&= n^{\log_3 2}T(1)+\tfrac{n}{\log_3^2 n}c=\Theta\left(\tfrac{n}{\log^2 n}\right)
\end{aligned}
$$

όπου $0<\log^{-2}3<c<\sum_{i=0}^{\infty}\left(\frac{2}{3}\right)^i\frac{1}{\left(1-\frac{i}{m}\right)^2}\log^{-2}3<+\infty\Rightarrow c=\Theta(1)$

(iv) Θα δείξω ότι $T(n)=\Theta(n)$

Ειδικότερα, αρκεί να δείξω ότι υπάρχει n_0, ώστε $4n\le T(n)\le 5n$, για κάθε $n\ge n_0$. Έστω ότι υπάρχει n_0, για το οποίο ισχύει η ζητούμενη σχέση, για κάθε n με $n_0\le n<k$ (όπου $k>2n_0$), τότε, ισχύει:

$$\frac{1}{6}4k+\frac{3}{5}4k+k\le T(k)\le\frac{1}{6}5k+\frac{3}{5}5k+k$$

$$\Rightarrow 4k<\frac{61}{15}k\le T(k)\le\frac{29}{6}k<5k$$

Με χρήση επαγωγής έπεται το ζητούμενο.

(v) Το δέντρο που δημιουργείται, με την αναδρομική κλήση του αλγορίθμου, έχει σε κάθε επίπεδο ακριβώς n κόμβους, μέγιστο ύψος $\log_2 n$ και ελάχιστο $\log_6 n$, άρα τελικά, ο χρόνος που χρειάζεται είναι $\Theta(n\log n)$. Αυτό, γράφεται αυστηρά ως εξής: Έστω $m_1=\log n$ και $m_2=\log_6 n$, τότε:

$$
\begin{aligned}
T(n) &= T\left(\tfrac{n}{6}\right)+T\left(\tfrac{n}{2}\right)+T\left(\tfrac{n}{3}\right)+n\\
&= T\left(\tfrac{n}{6^2}\right)+T\left(\tfrac{n}{6\cdot 2}\right)+T\left(\tfrac{n}{6\cdot 3}\right)+T\left(\tfrac{n}{2\cdot 6}\right)+T\left(\tfrac{n}{2^2}\right)+T\left(\tfrac{n}{2\cdot 3}\right)+\\
&\quad +T\left(\tfrac{1}{3\cdot 6}\right)+T\left(\tfrac{1}{3\cdot 2}\right)+T\left(\tfrac{1}{3^2}\right)+2n\\
&\ \ \vdots\\
&= \sum_{\substack{(r,s,t)\in\{0,\dots,k\}^3\\ r+s+t=k}} \tfrac{k!}{r!s!t!}T\left(\tfrac{n}{6^r\cdot 2^s\cdot 3^t}\right)+kn \quad \forall k \qquad (*)\\
&\overset{k=m_1}{=} \sum_{\substack{(r,s,t)\in\{0,\dots,m_1\}^3\\ r+s+t=m_1\\ 6^r2^s3^t\le n}} \tfrac{m_1!}{r!s!t!}T\left(\tfrac{n}{6^r\cdot 2^s\cdot 3^t}\right)+m_1 n\\
&\le \sum_{\substack{(r,s,t)\in\{0,\dots,m_1\}^3\\ r+s+t=m_1\\ 6^r2^s3^t\le n}} \tfrac{m_1!}{r!s!t!}T\left(\tfrac{n}{2^r\cdot 2^s\cdot 2^t}\right)+m_1 n\\
&= T(1)\sum_{\substack{(r,s,t)\in\{0,\dots,m_1\}^3\\ r+s+t=m_1\\ 6^r2^s3^t\le n}} \tfrac{m_1!}{r!s!t!}+n\log n\\
&= T(1)\,n+n\log n=\Theta(n\log n)
\end{aligned}
$$

Επίσης, τελείως όμοια:

$$
\begin{aligned}
(*) &\overset{k=m_2}{=} \sum_{\substack{(r,s,t)\in\{0,\dots,m_1\}^3\\ r+s+t=m_2\\ 6^r2^s3^t\le n}} \tfrac{m_2!}{r!s!t!}T\left(\tfrac{n}{6^r\cdot 2^s\cdot 3^t}\right)+m_2 n\\
&\ge \sum_{\substack{(r,s,t)\in\{0,\dots,m_2\}^3\\ r+s+t=m_2\\ 6^r2^s3^t\le n}} \tfrac{m_2!}{r!s!t!}T\left(\tfrac{n}{6^r\cdot 6^s\cdot 6^t}\right)+m_2 n\\
&= T(1)\sum_{\substack{(r,s,t)\in\{0,\dots,m_2\}^3\\ r+s+t=m_2\\ 6^r2^s3^t\le n}} \tfrac{m_2!}{r!s!t!}+n\log_6 n\\
&= T(1)\,n+n\log_6 n=\Theta(n\log n)
\end{aligned}
$$

Άρα, τελικά $T(n)=\Theta(n\log n)$.

(vi) Έστω $m=n-1$, τότε:

$$
\begin{aligned}
T(n) &= T(n-1)+\tfrac{1}{n}\\
&= T(n-2)+\tfrac{1}{n-1}+\tfrac{1}{n}\\
&\ \ \vdots\\
&= T(n-k)+\sum_{i=0}^{k-1}\tfrac{1}{n-k} \quad \forall k\\
&\overset{k=m}{=} T(1)+\sum_{s=2}^{n}\tfrac{1}{s}=\Theta(\log n)
\end{aligned}
$$

(vii) Έστω $m=\log\log n$ και c_1,c_2,n_0, κατάλληλα, ώστε:
$T(n)=2T(\sqrt{n})+f(n)$, με $c_1\log n\le f(n)\le c_2\log n$, $\forall n\ge n_0$. Έχουμε τότε:

$$
\begin{aligned}
T(n) &= 2T(\sqrt{n})+f(n)\\
&= 2^2T\left(n^{\frac{1}{2^2}}\right)+2f\left(n^{\frac{1}{2}}\right)+f(n)\\
&\ \ \vdots\\
&= 2^kT\left(n^{\frac{1}{2^k}}\right)+\sum_{i=0}^{k-1}2^if\left(n^{\frac{1}{2^i}}\right)\\
&\overset{k=m}{=} \log n\,T(2)+g(n)
\end{aligned}
$$

όπου $g\left(n\right)=\sum_{i=0}^{\log\log n-1}2^{i}f\left(n^{\frac{1}{2^{i}}}\right)=\Theta\left(\log n\log\log n\right)$
Πράγματι, αν $m>>n_0$, ισχύει:

$$\sum_{i=0}^{\log\log n-1-n_0}2^{i}c_1\log n^{\frac{1}{2^{i}}}\leq g\left(n\right)\leq\sum_{i=0}^{\log\log n-1-n_0}2^{i}c_2\log n^{\frac{1}{2^{i}}}$$

$$\Rightarrow c_1\log n\left(\log\log n-n_0\right)\leq g\left(n\right)\leq c_2\log n\left(\log\log n-n_0\right)$$

άρα, ισχύει και $T\left(n\right)=\Theta\left(\log n\log\log n\right)$.

(viii) Έστω $m=\frac{n}{10}$, τότε:

$$\begin{aligned}
T\left(n\right) &= T\left(n-10\right)+\log n\\
&= T\left(n-2\cdot 10\right)+\log\left(n-10\right)+\log n\\
&\vdots\\
&= T\left(n-k\cdot 10\right)+\textstyle\sum_{i=0}^{k-1}\log\left(n-10i\right)\quad\forall k\\
&\overset{k=m}{=} T\left(0\right)+\textstyle\sum_{i=0}^{m-1}\log\left(10\left(m-i\right)\right)\\
&= T\left(0\right)+\tfrac{n}{10}\log 10+\textstyle\sum_{s=1}^{m}\log s=\Theta\left(n\log n\right)
\end{aligned}$$

Άσκηση 2(α) Χρησιμοποιόντας Counting Sort, γίνεται ταξινόμηση σε χρόνο $\mathcal{O}\left(n+M\right)$. Πράγματι:

```
int* sort(int* A) {
  int B[M];
  for(int i=0;i<M;i++) B[i]=0;
  for(int i=1;i<=n;i++) {
    B[A[i]]++;
  }
  return B;
}
```

Το κάτω φράγμα δεν ισχύει, γιατί ο αλγοριθμος δεν είναι συγκριτικός και έχουμε και επιπλέον πληροφορίες, δηλαδή: α) ότι το $A\left[i\right]$ είναι θετικός ακέραιος για κάθε i και β) την ασυμπτωτική συμπεριφορά του μέγιστου στοιχείου, ως προς n.

Άσκηση 2(β) Τα φύλλα ενός συγκριτικού αλγορίθμου, για τον A πρέπει να περιλαμβάνουν κάθε δυνατή σχέση μεταξύ των $A[i]$. Οι $A[1],\ldots,A[n]$ είναι ακέραιοι στο $\{1,\ldots,M\}$. Μία τέτοια σχέση είναι της μορφής:

$$A[i_1]=A[i_2]=\cdots=A[i_{j_1}]<$$

$$<A[i_{j_1+1}]=A[i_{j_1+2}]=\cdots=A[i_{j_2}]<$$

$$< \cdots <$$
$$< A[i_{j_{m-1}+1}] = A[i_{j_{m-1}+2}] = \cdots = A[i_{j_m} = i_n]$$

όπου $i_1, \ldots, i_n$ αναδιάταξη των $1, \ldots, n$. Δηλαδή, αν $M \leq n$, κάθε πιθανή σχέση χαρακτηρίζεται από έναν αριθμό $0 < m \leq M$, που είναι το πλήθος των διαφορετικών τιμών που λαμβάνουν τα $A[i]$, από τον χωρισμό των $A[i]$ σε m μη κενές ομάδες και από κάθε πιθανή αναδιάταξη μεταξύ των m ομάδων. Δηλαδή, το πλήθος των φύλλων είναι ακριβώς ίσο με

$$l = \sum_{m=1}^{M} m! S(n, m)$$

όπου $S(n, m)$ οι αριθμοί Stirling δεύτερου είδους ($S(n, m) = \sum_{i=0}^{m} (-1)^i \binom{m}{i} (m-i)^n$). Για κάθε σταθερό m ισχύει $S(n, m) = \Theta\left(\frac{m^n}{m!}\right)$, άρα, τελικά, ισχύει

$$l = \sum_{m=1}^{M} \Theta(m^n)$$

Άρα, το ύψος του δέντρου του βέλτιστου συγκριτικού αλγορίθμου είναι της τάξης $\log l = \Theta(n \log M)$. Άρα, για να υπάρχει συγκριτικός αλγόριθμος ταξινόμησης με χρόνο εκτέλεσης $\mathcal{O}(n \log\log n)$, πρέπει $M = \mathcal{O}(\log n)$. Ο παρακάτω συγκριτικός αλγόριθμος βασίζεται στη λογική της Quicksort. Συγκεκριμένα, ο αλγόριθμος σε κάθε αναδρομική κλήση του, χρησιμοποιεί γραμμικό χρόνο για να χωρίσει τον πίνακα σε 3 μέρη. Αριστερά τοποθετούνται τα στοιχεία μικρότερα του pivot, δεξιά τα μεγαλύτερα και στο κέντρο όσα στοιχεία είναι ίσα με το pivot. Ο κώδικας που υλοποιεί τον αλγόριθμο είναι:

```
int* sort(int* A,int start, int end) {
  int left=start,right=end;
  int pivot=A[(right+left)/2];
  int centerleft=(right+left)/2,centerright=(right+left)/2;
  while(left<centerleft&&right>centerright) {
    while(A[centerleft]==pivot&&centerleft>start) centerleft--;
    while(A[centerright]==pivot&&centerright<end) centerright++;
    while(A[left]<pivot) left++;
    while(A[right]>pivot) right--;
    if(A[left]==pivot&&left<centerleft) swap(A[left],A[centerleft]);
    if(A[right]==pivot&&right>centerright) swap(A[right],A[centerright]);
    if(A[right]<pivot&&A[left]>pivot) swap(A[left],A[right]);
  }
  while(left>=centerleft&&right>centerright) {
    while(A[centerright]==pivot&&centerright<end) centerright++;
```

```
    while(A[right]>pivot) right--;
    if(A[right]==pivot&&right>centerright) swap(A[right],A[centerright]);
    if(A[right]<pivot) {
      swap(A[left],A[right]);
      left++;
    }
  }
  while(left<centerleft&&right<=centerright) {
    while(A[centerleft]==pivot&&centerleft>start) centerleft--;
    while(A[left]<pivot) left++;
    if(A[left]==pivot&&left<centerleft) swap(A[left],A[centerleft]);
    if(A[left]>pivot) {
      swap(A[left],A[right]);
      right--;
    }
  }
  A=sort(A,start,left-1);
  A=sort(A,right+1,end);
  return A;
}
```

Συγκεκριμένα, στο πρώτο βήμα, ο αλγόριθμος χρειάζεται χρόνο $T_0(n) = n$, στις 2 επόμενες αναδρομικές κλήσεις, ο αλγόριθμος θα χρειαστεί συνολικά χρόνο $T_1(n) = n - m_1$, όπου m_1 το πλήθος στοιχείων που είναι ίσα με το πρώτο pivot. Στο επόμενο βήμα (4 επιπλέον αναδρομικές κλήσεις), θα χρειαστεί χρόνο $T_2(n) = n - m_1 - m_2 - m_3$, όπου m_2, m_3 το πλήθος των στοιχείων για τα επόμενα 2 pivots, αντιστοίχως. Γενικά, στο βήμα k, θα ισχύει

$$T_k(n) = n - \sum_{i=1}^{2^k-1} m_i$$

όπου m_i, το πλήθος των στοιχείων που ταυτίζονται με το i-οστό pivot που χρησιμοποιείται. Γενικά, αν $T(n)$ ο χρόνος εκτέλεσης του προγράμματος, αθροίζοντας όλα τα βήματα, για $k \in \{0, \ldots, s = \log(M+1)\}$ (M είναι το πλήθος των διαφορετικών στοιχείων, δηλαδή το πλήθος των m_i), ισχύει:

$$\begin{aligned}
T(n) &= n + \sum_{k=1}^{s}\left(n - \sum_{i=1}^{2^k-1} m_i\right) \\
&= (s+1)\,n - \sum_{k=1}^{s}\sum_{i=1}^{2^k-1} m_i \\
&= (s+1)\,n - \sum_{i=1}^{2^s-1}\sum_{k=\lceil\log(i+1)\rceil}^{s} m_i \\
&= (s+1)\,n - \sum_{i=1}^{2^s-1} m_i \sum_{k=\lceil\log(i+1)\rceil}^{s} 1 \\
&= (s+1)\,n - \sum_{i=1}^{2^s-1} m_i\left(s - \lceil\log(i+1)\rceil\right) \\
&= (s+1)\,n - s\sum_{i=1}^{2^s-1} m_i + \sum_{i=1}^{2^s-1} m_i\lceil\log(i+1)\rceil \\
&\le (s+1)\,n - sn + \sum_{i=1}^{2^s-1} m_i\lceil\log(2^s)\rceil \\
&= (s+1)\,n - sn + s\sum_{i=1}^{2^s-1} m_i \\
&= (s+1)\,n \\
&= n\left(\log(M+1)+1\right) = \mathcal{O}(n\log M) = \mathcal{O}(n\log\log n)
\end{aligned}$$

όπου χρησιμοποιώ το γεγονός $n = \sum_{i=1}^{M} m_i$.

Άσκηση 2(γ) Εφαρμόζοντας radix sort για τους αριθμούς στο n-αδικό σύστημα αρίθμησης, ταξινομείται ο A σε γραμμικό χρόνο (δεδομένου ότι το d είναι σταθερό). Συγκεκριμένα, ο κώδικας για τον αλγόριθμο είναι:

```
void radixsort() {
  int B[n][n];
  int pointer[n];
  for(int r=0;r<d;r++) {
    for(int i=0;i<n;i++) point[i]=0;
    for(int k=1;k<=n;k++) {
      int digit=(A[k]/pow(n,d))%n;
      B[digit][pointer[digit]]=A[k];
      pointer[digit]++;
```

```
    }
    int k=1;
    for(int s=0;s<n;s++) {
      for(int i=0;i<pointer[s];i++) {
        A[k]=B[s][i];
        k++;
      }
    }
  }
}
```

Άσκηση 3(α)

(i) Έστω $b_i = A[i] - i$ για $i \in \{1, \ldots, n\}$, τότε ισχύει $b_1 \geq 0$ και $b_n \leq 0$. Έστω m ο μικρότερος δείκτης, για τον οποίο ισχύει $b_m \geq 0$ και $b_{m+1} \leq 0$. Ισχύει $b_m - b_{m+1} = A[m] - m - A[m+1] + m + 1 = |A[m] - A[m+1]| + 1 \leq k + 1$, από την εκφώνηση. Αν $b_m > \frac{k+1}{2}$ και $b_{m+1} < -\frac{k+1}{2}$, τότε θα είχαμε άτοπο, άρα τουλάχιστον για ένα από τα m και $m+1$ ισχύει το ζητούμενο.

(ii) Ένας αποδοτικός αλγόριθμος είναι η δυαδική αναζήτηση στον πίνακα $A[i]$, (όπως στη μέθοδο για την εύρεση ρίζας σε συνεχή συνάρτηση). Η πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(\log n)$. Πράγματι, ο κώδικας του αλγορίθμου είναι:

```
int findroot(int* A,int start, int end, int k) {
  int med=(start+end)/2;
  if(abs(A[med]-med)<=(k+1)/2) return med;
  else return findroot(A,start,med);
  else if(A[med]>med) return findroot(A,med,end);
}
```

Άσκηση 3(β) Ένας αποδοτικός αλγόριθμος είναι ο εξής: Έστω A κάποιος πίνακας με n γραμμές και m στήλες. Με δυαδική αναζήτηση στη γραμμή $\frac{n}{2}$ βρίσκω k_1 με $A[n/2][k_1] \leq k$ και $A[n/2][k_1+1] > k$. Ομοίως ορίζω το k_2 στην στήλη $\frac{m}{2}$. Όποια από τα $A[n/2][k_1], A[k_2][m/2]$ είναι ίσα με k τα εμφανίζω και αναδρομικά καλώ τον ίδιο αλγόριθμο 3 φορές, στους πίνακες:

$$A[\max\{k_2, n/2\} \ldots n][1 \ldots \min\{k_1, m/2\}]$$

$$A[\min\{k_2, n/2\} \ldots \max\{k_2, n/2\}][\min\{k_1, m/2\} \ldots \max\{k_1, m/2\}]$$

$$A[1 \ldots \min\{k_2, n/2\}][\max\{m/2, k_1\} \ldots m]$$

Επίσης, αν $a = min\{n, m\}$ και $b = max\{n, m\}$, τότε η δυαδική αναζήτηση στην διάσταση a θα τελειώσει πιο νωρίς από τη διάσταση b. Προφανώς, από εκεί και πέρα, το πρόβλημα σε πίνακες γραμμές (αντίστοιχα πίνακες στήλες) λύνεται σε χρόνο $T'(k) = T(k,1) = T(1,k) = \Theta(\log k)$ Ο κώδικας που αντιστοιχεί στον αλγόριθμο αυτό είναι:

```
int findk(int* array, int start, int end, int k) {
  if(start==end) return start;
  if(start==end-1&&array[end]<=k) return end;
  int med=(start+end)/2;
  if(array[med]<=k&&array[med+1]>k) return med;
  else if(array[med]<=med) return findk(A,med,end,k);
  else return findk(A,start,med,k);
}
void printvaluek(int** A, int startn, int endn, int startm, int endm,
    int k) {
  int medn=(startn+endn)/2;
  int medm=(startm+endm)/2;
  int Am[n],An[m];
  for(int i=startn;i<=endn;i++) Am[i]=A[i][medm];
  for(int i=startm;i<=endm;i++) An[i]=A[medn][i];
  int kn=findk(Am,startn,endn,k);
  int km=findk(An,startm,endm,k);
  if(Am[kn]==k) cout<<"("<<kn<<";"<<medm<<")"<<endl;
  if(An[km]==k) cout<<"("<<medn<<";"<<km<<")"<<endl;
  if(startn!=endn&&startm!=endm) {
    printvaluek(A,max(kn,medn),endn,startm,min(km,medm));
    printvaluek(A,min(kn,medn),max(kn,medn),min(km,medm),max(km,medm));
    printvaluek(A,startn,min(kn,medn),max(km,medm),endm);
  }
}
```

Η χειρότερη περίπτωση σε κάποια επανάληψη του αλγορίθμου, είναι ο μεσαίος πίνακας να εκφυλίζεται σε στοιχείο του πίνακα, να ισχύει δηλαδή $medn = kn$ (και άρα, κατά συνέπεια, $medm = km$). Άρα, αν $p = \log(\min\{m,n\})$ και $q = \log(\min\{m,n\})$ ισχύει:

$$
\begin{array}{rcl}
T(n,m) & \leq & 2T\left(\frac{n}{2},\frac{m}{2}\right) + \log n + \log m \\
 & = & 4T\left(\frac{n}{4},\frac{m}{4}\right) + 2\log\frac{n}{2} + 2\log\frac{m}{2} + \log n + \log m \\
 & \vdots & \\
 & = & 2^k T\left(\frac{n}{2^k},\frac{m}{2^k}\right) + \sum_{i=0}^{k-1} 2^i \log\frac{n}{2^i} + \sum_{i=0}^{k-1} 2^i \log\frac{m}{2^i} \\
 & \overset{k=p}{=} & \min\{n,m\} T'\left(\frac{\max\{n,m\}}{\min\{n,m\}}\right) + \sum_{i=0}^{p-1} 2^i (p-i) + \sum_{i=0}^{p-1} 2^i (q-i) \\
 & = & \max\{n,m\} \log\frac{\max\{n,m\}}{\min\{n,m\}} + 2^p \sum_{s=1}^{p} \frac{s}{2^s} + 2^p \sum_{s=q-p+1}^{p} \frac{s}{2^s} \\
 & = & \mathcal{O}\left(\min\{n,m\} \log\frac{\max\{n,m\}}{\min\{n,m\}}\right)
\end{array}
$$

Διακρίνω 2 περιπτώσεις:

- $\min\{n,m\} = \Theta(\max\{n,m\})$

Σε αυτή την περίπτωση $\log \frac{\max\{n,m\}}{\min\{n,m\}} = \Theta(1)$, άρα, τελικά:
$T(n,m) = \mathcal{O}(\min\{n,m\}) = \mathcal{O}(n)$

- $\min\{n,m\} = o(\max\{n,m\})$
  Σε αυτή την περίπτωση $\log \frac{\max\{n,m\}}{\min\{n,m\}} = \Theta(\log \max\{n,m\})$. Πράγματι:

$$\frac{\log \frac{\max\{n,m\}}{\min\{n,m\}}}{\log \max\{n,m\}} = \frac{\log \max\{n,m\}}{\log \max\{n,m\}} - \frac{\log \min\{n,m\}}{\log \max\{n,m\}} \to 1$$

  άρα, τελικά:
  $T(n,m) = \mathcal{O}(\min\{n,m\} \log \max\{n,m\})$

Άσκηση 3(γ) Κάθε συγκριτικός αλγόριθμος για το πρόβλημα (β), θα πρέπει να λαμβάνει υπ΄ όψιν όλες τις δυνατές θέσεις του k. Αυτές υπολογίζονται ως εξής: Διαλέγω s το πλήθος των φορών που εμφανίζεται το k στον πίνακα A, όπου $s \in \{0, \ldots, \min\{n,m\}\}$ και στη συνέχεια επιλέγω s γραμμές από τις n και s στήλες από τις m. Έτσι καθορίζονται πλήρως οι θέσεις των στοιχείων k, καθώς πρέπει να ισχύει ότι σε κάθε στήλη και σε κάθε γραμμή τα στοιχεία είναι αύξουσα ακολουθία. Άρα, το πλήθος των δυνατών φύλλων είναι:

$$l = \sum_{s=0}^{\min\{n,m\}} \binom{n}{s}\binom{m}{s}$$

Για κάθε φυσικό αριθμό N, ισχύει η ακόλουθη ανισοτική σχέση $\binom{N}{i} \leq \binom{N}{\lfloor \frac{N}{2} \rfloor}$ για κάθε $i \in \{0, \ldots, N\}$. Επίσης, η συνάρτηση $\phi(i) = \binom{N}{i}$ είναι αύξουσα όταν $i \in \{1, \ldots, \lfloor \frac{N}{2} \rfloor\}$. Αν λοιπόν $\min\{n,m\} = o(\max\{n,m\})$, τότε υπάρχει κάποιος φυσικός n_0, ώστε $\min\{n,m\} < \frac{\max\{n,m\}}{2}$ για κάθε (n,m) με $\max\{n,m\} > n_0$, τότε ισχύει:

$$\binom{\max\{n,m\}}{\min\{n,m\}} \leq l \leq \binom{\max\{n,m\}}{\min\{n,m\}} \sum_{s=0}^{\min\{n,m\}} \binom{\min\{n,m\}}{s} = \binom{\max\{n,m\}}{\min\{n,m\}} 2^{\min\{n,m\}}$$

Δηλαδή

$$\Theta\left(\max\{n,m\}^{\min\{n,m\}}\right) \leq l \leq \Theta\left((2\max\{n,m\})^{\min\{n,m\}}\right)$$

Άρα, ο χρόνος εκτέλεσης σε αυτή την περίπτωση είναι: $\log l = \Theta(\min\{n,m\} \log \max\{n,m\})$.
Διαφορετικά, αν $\min\{n,m\} = \Theta(\max\{n,m\})$, τότε υπάρχουν κάποια $M, n_0 \in \mathbb{N}$ ώστε $\left\lfloor \frac{\max\{n,m\}}{M} \right\rfloor \leq \min\{n,m\}$ για κάθε (n,m) με $\max\{n,m\} > n_0$. Άρα, ισχύει:

$$\binom{\max\{n,m\}}{\left\lfloor \frac{\max\{n,m\}}{M} \right\rfloor} \leq l \leq \binom{\max\{n,m\}}{\left\lfloor \frac{\max\{n,m\}}{M} \right\rfloor} \sum_{s=0}^{\min\{n,m\}} \binom{\min\{n,m\}}{s} = \binom{\max\{n,m\}}{\left\lfloor \frac{\max\{n,m\}}{M} \right\rfloor} 2^{\min\{n,m\}}$$

Όμως, χρησιμοποιόντας τον τύπο του Stirling, ισχύει

$$\binom{\max\{n,m\}}{\left\lfloor \frac{\max\{n,m\}}{M} \right\rfloor} = \Theta\left(\frac{1}{\sqrt{\max\{n,m\}}} \frac{M^{\max\{n,m\}}}{(M-1)^{\frac{M-1}{M}\max\{n,m\}}}\right)$$

Δηλαδή, ο χρόνος εκτέλεσης σε αυτή την περίπτωση είναι: $\log l = \Theta(\max\{n,m\})$
Αν $m \leq n$, τότε ισχύει $m = \min\{n,m\}$ και $n = \max\{n,m\}$, άρα οι περιπτώσεις γίνονται:

- Αν $m = \Theta(n)$, τότε $T(n,m) = \Theta(n)$
- Αν $m = o(n)$, τότε $T(n,m) = \Theta(m \log n)$

Αν τα n, m δεν συγκρίνονται ασυμπτωτικά, τότε δεν έχω βρει κάτω φράγμα. Σε κάθε άλλη περίπτωση, ο αλγόριθμος του 4(β) είναι βέλτιστος.

Άσκηση 4(α) Με δυαδική αναζήτηση στο $\{1,\ldots,M\}$, βρίσκω l τέτοιο ώστε $F_S(l) \leq k$ και $F_S(l+1) > k$. Ο κώδικας που υλοποιεί τον αλγόριθμο αυτό είναι:

```
int findk(int start, int end, int k) {
  int med=(start+end)/2;
  if(F_S(med)<=k&&F_S(med+1)>k) return med;
  else if(F_S(med)<=k) return findk(med,end,k);
  else return findk(start,med,k);
}
```

Ο αλγόριθμος, δηλαδή, χρειάζεται χρόνο (πλήθος κλήσεων της F_S) $\mathcal{O}(\log M)$. (ανεξάρτητο του n).

Άσκηση 4(β) Για την υλοποίηση της F_S εφαρμόζω counting sort στον πίνακα A και προκύπτει πίνακας B, σε χρόνο $\mathcal{O}(n+M)$. Κατασκευάζω μετά πίνακα C, με $C[i] = \sum_{j=1}^{i} B[j]$ (σε χρόνο $\mathcal{O}(M)$) και στη συνέχεια, υπολογίζω την F_S με τον τύπο: $F_S(l) = \sum_{i=1}^{M-l} (C[i+l] - C[i]) \cdot B[i]$ (σε χρόνο $\mathcal{O}(M)$). Δηλαδή πολλαπλασιάζω το πλήθος των στοιχείων του A, που είναι από $i+1$ εως $i+l$ με το πλήθος των στοιχείων που είναι ακριβώς i, και στη συνέχεια αθροίζω πάνω από όλα τα i. Ο κώδικας που αντιστοιχεί σε αυτό είναι:

```
int F_S(int l) {
  int result=0;
  for(int i=1;i<=M-l;i++) result+=(C[i+l]-C[i])*B[i];
  return result;
}
int findk(int start, int end, int k) {
  int med=(start+end)/2;
  if(F_S(med)<=k&&F_S(med+1)>k) return med;
```

```
  else if(F_S(med)<=k) return findk(med,end,k);
  else return findk(start,med,k);
}
int main() {
  int B[M],C[M];
  for(int i=1;i<=M;i++) B[i]=0;
  for(int i=1;i<=n;i++) B[A[i]]++;
  C[1]=B[1];
  for(int i=2;i<=M;i++) C[i]=C[i-1]+B[i];
  cout<<findk(1,n,k)<<endl;
}
```

Τελικά, ο αλγόριθμος κάνει χρόνο $\mathcal{O}\left(n + M \log M\right)$