Εργασία 1 P3200262

① $(\log_2 n)^9 < \sqrt[4]{n} < n\log_2(\log_2 n) < n\log_2 n < \log_2 n^{\log_2 n} < 2^{(\log_2 n)^2} < n^{\log_2 n}$
$< n^{\sqrt{n}} < \sqrt{n}^{\,n} < n! < 2^{n\log_2 n} < n^n$

② $T(n) = aT(n/b) + f(n)$

Θα εφαρμόσουμε Master Theorem και θα καταλήξουμε σε άνω ρέαγμα ανάλογα:

$n^{\log_b a} > f(n) \Rightarrow T(n) = O(n^{\log_b a})$
$n^{\log_b a} = f(n) \Rightarrow T(n) = O(n^{\log_b a}\log_2 n)$
$n^{\log_b a} < f(n) \Rightarrow T(n) = O(f(n))$

(ΕΠΙΣΗΣ ΧΩΡΙΣ ΒΑΣΗ ΣΥΓΚΡ. $\log_b a$ με d όπου d βαθμός $f(n)$)

i) $T(n) = T(n/2) + O(n)$
$a=1 \quad b=2 \quad f(n)=n \quad d=1$

$n^{\log_2 1} = n^0 = 1 < n$ Άρα $\underline{T(n) = O(n)}$

ii) $T(n) = T(n/2) + O(1)$
$a=1 \quad b=2 \quad f(n)=1$

$n^{\log_2 1} = n^0 = 1 = f(n)$ Άρα $\underline{T(n) = O(n^0 \cdot \log_2 n) = O(\log_2 n)}$

iii) $T(n) = 2T(n/2) + O(1)$
$a=2, \quad b=2 \quad f(n)=1$
$n^{\log_2 2} = n^1 = n > f(n)$ Άρα $\underline{T(n) = O(n)}$

iv) $T(n) = 4T(n/2) + O(n^2)$

$a=4 \quad b=2 \quad f(n)=n^2$

$n^{\log_2 4} = n^2 = f(n)$ Άρα $\underline{T(n) = O(n^2 \log_2 n)}$

v) $T(n) = 4T(n/2) + O(n^3)$

$a=4 \quad b=2 \quad f(n)=n^3$

$n^{\log_2 4} = n^2 < f(n)$ Άρα $\underline{T(n) = O(n^3)}$

vi) $T(n) = 4T(n/2) + O(n)$

$a=4 \quad b=2 \quad f(n)=n$

$n^{\log_2 4} = n^2 > f(n)$ Άρα $\underline{T(n) = O(n^2)}$

3 Ο αλγόριθμος A πολλαπλασιάζει την τιμή a με την τιμή b.
Αρχικοποιούμε το $p \leftarrow 0$, και αρχικά $a \cdot b = result \Leftrightarrow (a \cdot b) + \overset{0}{p} = result$
Σε κάθε loop, διαίρεσε το $a/2$ και πολλαπλασίασε το $b*2$. Αν το a είναι μονός, πρόσθεσε το b στο p (αποτέλεσμα αρχικοποιημένο σε 0).

$\boxed{\text{Αναλλοίωτη Βρόχου}}$: Σε κάθε επανάληψη $a \cdot b + p = result$.
Αυτό γίνεται έως ότου $a=0$, άρα $0 \cdot b + p = result$ $\Rightarrow p = result$. Επομένως αποδείξαμε ότι στο τέλος αυτού του αλγορίθμου θα έχουμε το αποτέλεσμα ορθά.

④ Το πρόβλημα ορίζεται για $a \in [0,1]$ καθώς δεν υπάρχουν a-δημοφιλή στοιχεία για $a > 1$.

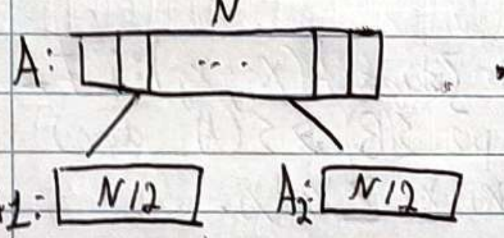
Αλγόριθμος #1

ΛΥΣΗ: Σπάω αναδρομικά τον πίνακα σε 2 υποπίνακες έως την βάση 1.

Καθώς επιστρέφουν οι αναδρομικές κλήσεις:

→ Επιστρέφω ένα σύνολο a-δημοφιλών τιμών. Στο επίπεδο της βάσης, όλα τα στοιχεία είναι a-δημοφιλή.

π.χ.



$S_1 = apop(A_1)$ $\quad$ $S_2 = apop(A_2)$

• Έστω S_1, S_2 τα σύνολα με τα a-δημοφιλή στοιχεία των A_1, A_2 respectively.

Έστω S το σύνολο του A.

• Καθώς κάνω merge $A_1 + A_2$ στον A:

i) → Αν $S_1 = S_2$, τότε $S = S_1 = S_2$ και κάθε $x \in S$ είναι a-δημοφιλή στον τελικό.

ii) → Αν $S_1 \neq S_2$ αυστηρά, ελέγχω για όλα τα στοιχεία $x \in (S_1 \cup S_2)$ εάν εμφανίζονται $size(A) \cdot a$ φορές στον πίνακα. Κρατάω μόνο όσα στοιχεία πληρούν την συνθήκη στο S.

iii) → Αν S_1 μερικώς διάφορο του S_2, τότε $S = S_1 \cap S_2$ και εκτελώ το ii) για κάθε $x \in (S_1 \cup S_2) - (S_1 \cap S_2)$, και τα προσθέτω στο S αν είναι a-δημοφιλή.

(ΣΥΝΕΧΕΙΑ ΠΙΣΩ)

⑤ Θα ταξινομήσουμε τον πίνακα σε $O(n \log_2 n)$ χρόνο. (Αν αταξινόμητος)

Στη συνέχεια, θα διαλέγαμε με δείκτες $i, j : (i=0, j=2n)$ τα i-th και j-th στοιχεία και θα δημιουργήσουμε το ζευγάρι $P(x_i, x_j)$. Αυξάνουμε το $i+1$ και μειώνουμε το $j-1$ έως ότου να έχω n ζευγάρια. Χρόνος $O(n)$.

π.χ. $(x_1, x_{2n}) \ldots (x_2, x_{2n-1}) \ldots$

Άρα ο Greedy αλγόριθμός μας τρέχει σε $O(n \log_2 n)$.

Η $O(n)$ αν είναι ήδη ταξινομημένος ο πίνακας ΚΑΙ ΤΟ ΓΝΩΡΙΖΟΥΜΕ.

Ορθότητα:

Έστω ΟΡΘΟΣ αλγόριθμος που φτιάχνει η ζεύγη απο τα 2n στοιχεία.

n=0,2 • Έστω ζεύγη ΟΡΘΟΥ == ζεύγη δικού μας, τότε ο δικός μας είναι ορθός. Επίσης για $n=0$, $n=2$ ∣ 0 ή 1 δυνατά ζευγη

n>2 • Έστω ότι ο ΟΡΘΟΣ φτιάχνει ζεύγος $A(x_i, x_{2n}) : i \neq 1$ και $B(x_1, x_j) : j \neq 2n$ (προφανώς), με $\Sigma(B) > \Sigma(A)$. αφού $x_i > x_1$ και $x_{2n} > x_j \Rightarrow x_i + x_{2n} > x_1 + x_j$.

Έστω $\Gamma(x_1, x_{2n})$ και $\Delta(x_i, x_j)$ με ανταλλαγή στοιχείων.

Όμως, $x_i + x_{2n} > x_1 + x_{2n}$ και $x_i + x_{2n} > x_i + x_j$

Αφού $x_i > x_1$ — Αφού $x_{2n} > x_j$

Άρα το συνολικό μέγιστο άθροισμα δεν αυξάνεται μετά την εναλλαγή.

Επομένως ο αλγόριθμός μας είναι ορθός.

□

④ [ΣΥΝΕΧΕΙΑ]

Ο αλγόριθμος διαιρεί το πρόβλημα σε 2 subproblems, κάτι που μας δίνει αναδρομικό δέντρο ύψους logn.

Σε κάθε επίπεδο θα γίνουν στην χειρότερη n^2 λειτουργίες (στην περίπτωση $\boxed{ii}$)

Επομένως έχουμε: $T(n) = 2T(n/2) + n^2$ | $a=2, b=2, f(n)=n^2$

Άρα: $n^{\log_b a} = n^1 = n < f(n) \Rightarrow \boxed{O(n^2)}$

Πρακτικά ο αλγόριθμός μας είναι στο $\Theta(n \log n)$.

④ Αλγόριθμος #2 Μία καλύτερη λύση

Ανάλογα με τον αλγόριθμο #1, θα επιστρέφουμε σύνολα, με την διαφορά ότι εδώ θα περιέχουν ένα ζεύγος $(x, v) : v \geq 1$ όπου x ένα στοιχείο του αρχικού πίνακα.

! Χωρίς loss of generality, ας υποθέσουμε ότι τα σύνολα έχουν μοναδικά x και το v το προσθέτουν στην ένωση $\cup$ (όπως στην SQL). π.χ. $\{(a,3)\} \cup \{(a,2),(b,5)\} \Rightarrow$
και $S_1 = S_2$ αν $x_{S_1} = x_{S_2}$. $\Rightarrow \{(a,5),(b,5)\}$

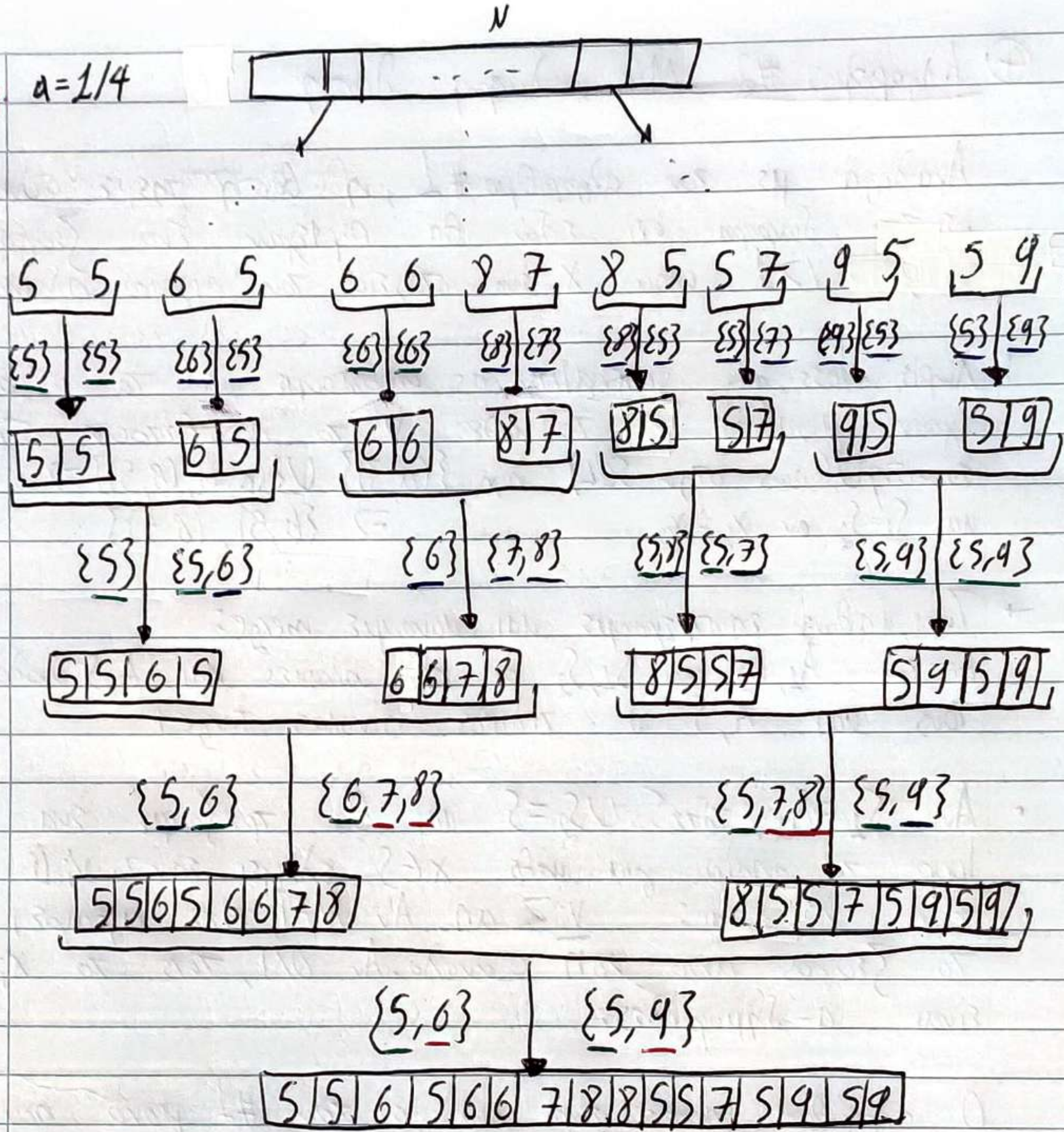
→ Τώρα, καθώς επιστρέφουμε και κάνουμε merge:
Έστω A_1, A_2 και S_1, S_2 οι $n/2$ πίνακες και τα σύνολά τους και A, S ο τελικός πίνακας target.

- Θα κάνουμε $S_1 \cup S_2 = S$ και θα τρέξουμε ένα loop το οποίο για κάθε $x \in S$ ελέγχει εάν το v_i του x_i είναι: $v_i < an$. Αν ΝΑΙ, τότε αφαιρεί το ζεύγος από το σύνολο. Αν ΟΧΙ, τότε το x_i είναι a-δημοφιλές.

Ο αλγόριθμος αυτός είναι ανάλογος του #1 μόνο που δεν χρειάζεται να κάνει parse το $|S| = n$ άλλες n φορές και το loop παραμένει γραμμικό.

$T(n) = 2T(n/2) + n \Rightarrow \boxed{O(n\log_2 n)}$

π.χ. $a = 1/4$



$S = \{5\}$

- Φεύγει από το loop ελέγχου a-κανονικότητος
- Μένει λόγω ισότητας S_1, S_2
- Μένει λόγω loop ελέγχου.

ΤΟ ΠΑΡΑΠΑΝΩ ΙΣΧΥΕΙ ΚΑΙ ΓΙΑ ΤΟΥΣ 2 ΑΛΓΟΡΙΘΜΟΥΣ