Data Structures and Algorithms

István András Seres @Eötvös Loránd University

2019 March 5, Tuesday 12:30-14:00, D-0.221

1 Warm-up and recap

True or false? Justify your answer!

- $\sqrt{n} = \Omega(\log(n))$
- $n! = \mathcal{O}(n^{2019})$
- Insertion-sort performs like quicksort on an almost sorted sequence.
- $3n^2 + \frac{1}{n} + 4 = \Theta(n^2)$
- $n\log(n) + n = \mathcal{O}(n \log n)$

2 Majdnem rendezett tömb

Adott különböző számoknak egy növekvően rendezett $A[1 : n]$ tömbje. A tömb elemeit valaki megkeveri, de csak annyira, hogy minden egyes elem új helye az eredetitől legfeljebb k távolságra esik. Adjunk hatékony algoritmust az eredeti állapot helyreállítására!

3 Legközelebbi pontpár

Adott egy $\mathscr{P} = \{p_1, p_2, \ldots, p_n\}$ ponthalmaz a síkon. Szokásos módon, a $p_i = (x_i, y_i)$ és $p_j = (x_j, y_j)$ pontok távolsága

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Adjunk hatékony algoritmust a két legközelebbi pont meghatározására (ha több ilyen pár is van, akkor megelégszünk az egyikkel)!

4 Bináris számok rendezése

Adott d jegyű bináris számok egy $A[1 : n]$ tömbje. Tervezzünk $O(nd)$ költségű algoritmust a tömb rendezésére!

5 Többségi elem

Ha egy $A[1:n]$ tömbben van olyan elem, amely több, mint $n/2$ példányban fordul elő, akkor azt a tömb többségi elemének nevezzük (nyilvánvaló, hogy legfeljebb egy többségi elem lehet egy tömbben). A tömb elemei nem feltétlenül egy rendezett halmazból valók, ezért csak azt tudjuk ellenőrizni, hogy két elem megegyezik-e, más típusú összehasonlításokat nem tudunk végezni. Adjunk hatékony algoritmust annak eldöntésére, hogy a tömb tartalmaze többségi elemet!

6 The power of randomized algorithms: *Bonus**

Freivald's algorithm is a randomized algorithm to check the correctness of a matrix multiplication. Suppose you have $A, B, C \in M^{n*n}$ and prover claims, that $A \times B = C$. Verifier can check this without computing the costly matrix multiplication. Procedure works as follows:

- Verifier generates a $n \times 1$ random binary vector $\overrightarrow{r}$, i.e. $\overrightarrow{r} \in \{0,1\}^n$
- Verifier computes $\overrightarrow{P} = A \times (B\overrightarrow{r}) - C\overrightarrow{r}$
- Verifier outputs "YES" if $\overrightarrow{P} = (0, 0, \ldots, 0)^T$; "NO", otherwise.

It is quite easy to see, that the algorithm is complete, namely if multiplication is correct then prover can always convince verifier. However with a constant probability a cheating prover could convince verifier. Show that if $A \times B \neq C$, then the probability that the algorithm returns "YES" is less then or equal to $\frac{1}{2}$.