D & C

- Je dáno pole $A[1 \dots n]$ obsahující ostře rostoucí posloupnost cisel. Jak zjistit, jestli existuje index $i : A[i] = i$?
- Totéž pro matici $n \times n$, v níž hledáme indexy i, j takové, že $A[i, j] = i + j$.
- Máme n kabelů spojujících opačná křídla budovy. Chceme zjistit, které konce patří k sobě. Umíme do libovolného množství z nich zavést napětí a na druhé straně všechny rychle změřit. Kolikrát musíme jít tam a zpět? Pokud je na druhé straně měřící přístroj, který si po každém přepojení napětí zapametuje všechny výsledky, jak velkou potřebuje paměť, abychom mohli všechny kabely určit bez vracení?
- Mějme graf daný maticí sousednosti. Spočítejte co nejrychleji matici dosažitelnosti ($D[i, j] = 1 <=>$ existuje cesta z v_i do d_j)

RAM

Typy operací : aritmetické, logické a binární, řídící (goto,gotoif,halt)
Adresace: přímá [x], nepřímá [[x]]

- Teorie: kde je vstup a výstup, jak se počítá časová a prostorová složitost, co je složitost operace a co to znamená
- Jak implementovat cykly?
- Implementujte bubble-sort
- Jak implementovat nekonečné pole? A víc takových polí? Jaká technická zjednodušení si můžeme dovolit?
- Jak implementovat volání funkce? Jak implemetovat rekurzi?
- Implementujte (nějaký hloupý) quick-sort.

Domácí úkol

První část

Pomocí metody rozděl a panuj je možné najít medián v lineárním čase. Klíčovým krokem je vyhledávání medianu v pěticích čísel. Je sice jedno jaký postup použijeme, ale proč to neudělat optimálně. Chceme navrhnout algoritmus, který pro čísla $x_1, \ldots, x_5$ na vstupu najde medián optimálně, tzn. s minimálním počtem porovnání v nejhorším případě.

Pokud to umíte, ukažte jak takový algoritmus vypadá a dokažte, že je optimální. K tomu se budou hodit nějaké dobré odhady.

NEBO navrhněte algoritmus, který takový optimální algoritmus najde. Jak hledat algoritmus algoritmem? Můžeme si algoritmus představit jako strom, operací "porovnej[i,j]" (s dvěma syny) a "median je i" (list). Optimální algortimus potom odpovídá nejmělčímu stromu, který dává správné výsledky. Stačí doplnit detaily a domlátit se k řešení hrubou silou (složitost nás nezajímá, algoritmus nám stačí najít pouze jednou).

Druhá část

Máme dánu posloupnost čísel $a_1, \ldots, a_n$ (i negativních). Chceme najít úsek s největším součtem. Brute-force přístup vydá jednoduchý algoritmus $\mathcal{O}(n^3)$. My ale už víme, že metodou rozděl a panuj obvykle získáme něco lepšího.

Navrhněte D& C-algoritmus hledající úsek s největším součtem a analyzujte jeho časovou složitost. Hint: Jako vždy to bude zdánlivě hloupý přístup, prostě přesekněte posloupnost napůl a vypořádejte se s tím, co přečnívá.