

NKD

Zadatak možemo riješiti direktnom simulacijom tako da za svaki otvoreni dokument provjerimo nalazi li se u listi ili ga je potrebno dodati. Zatim premjestimo dokument na početak liste i po potrebi skratimo listu.

Drugi pristup rješavanju zadatka je konstrukcija samo konačne liste nakon otvaranja svih dokumenata. Prvi dokument u listi bit će onaj koji je zadnji otvoren. Drugi dokument bit će onaj koji je predzadnji otvoren itd. Prolazimo niz dokumenata od zadnjeg prema prvom i odabiremo dokumente koji još nisu u listi sve dok ne popunimo kapacitet liste ili ne prođemo cijeli niz dokumenata.

Rješenje u C++ implementira prvi pristup, dok rješenje u Pascalu implementira drugi.

NIZOVI

Broj X se nalazi u oba niza ako pri dijeljenju s $(A+B)$ daje ostatak 0 ili A , i ako pri dijeljenju s $(C+D)$ daje ostatak 0 ili C . Međutim rješenje koje to provjerava redom za sve cijele brojeve je presporo i dobiva 50% bodova. Vremenska složenost tog algoritma jest $O((A+B) \cdot (C+D))$.

Da bismo dobili sve bodove potrebno je gornju provjeru raditi samo za brojeve u jednom od nizova. Generiramo redom brojeve u prvom nizu i provjeravamo nalazi li se u drugom nizu (gledanjem ostatka pri djeljenju s $C+D$). Vremenska složenost algoritma jest $O(A+B+C+D)$.

Rješenje koje koristi 32-bitne brojeve s predznakom dobiva 90% bodova. Za sve bodove potrebno je koristiti 32-bitne brojeve bez predznaka ili 64-bitne brojeve.

OCR

Dovoljno je slijediti uputu iz zadatka. Obilazimo polja redom po stupcima, a unutar stupca po recima i pokušavamo svaki broj nacrtati na trenutnu poziciju. Ako sva crna polja odgovaraju, tada ispisujemo broj.

Brojeve ne možemo stavljati u bilo kojem poretku jer na neke pozicije možemo upisati više brojeva (npr. 0 se može napisati preko broja 8, 5 preko broja 6, itd.). Jedno moguće rješenje je sortirati brojeve silazno prema broju crnih polja. Na taj način ćemo uvijek staviti onaj broj koji stvarno piše.

Alternativno rješenje (implementirano u C++ i u Pascalu) traži u tablici slovo 'X' i rekurzivnom pretragom ga izbriše s tablice, zapisujući pritom svaki pomak u string. Za pojedinu znamenku, rekurzivni algoritam uvijek će vratiti isti string.

Dobiveni string uspoređuje se sa stringovima koji su dobiveni istim postupkom na tablici koja sadrži redom znamenke od 0 do 9, kako bismo znali kojoj znamenci odgovara pojedini string. Za detalje pogledajte priložene izvorne kodove.